МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ОБРАЗОВАНИЕ
НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ

СЕТЬ ЦЕНТРОВ ЦИФРОВОГО ОБРАЗОВАНИЯ ДЕТЕЙ «IT-КУБ»

РЕАЛИЗАЦИЯ ДОПОЛНИТЕЛЬНОЙ ОБЩЕОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ ПО ТЕМАТИЧЕСКОМУ НАПРАВЛЕНИЮ

«ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON»

С ИСПОЛЬЗОВАНИЕМ ОБОРУДОВАНИЯ ЦЕНТРА ЦИФРОВОГО ОБРАЗОВАНИЯ ДЕТЕЙ «IT-КУБ»

МОСКВА 2021

С. Г. Григорьев
М. А. Родионов
И. В. Акимова

Реализация дополнительной общеобразовательной программы по тематическому направлению «Программирование на языке Python» с использованием оборудования центра цифрового образования детей «IT- куб»

Методическое пособие

под ред. С. Г. Григорьева

Москва, 2021

Введение

Цель и задачи создания центров цифрового образования детей «IT-куб»

Целями и задачами центров цифрового образования детей «IT-куб» является реализация программ дополнительного образования, проведение мероприятий по тематике современных цифровых технологий и информатики, знакомство детей с технологиями искусственного интеллекта, а также обеспечение просветительской работы по цифровой грамотности и цифровой безопасности.

Задачами центра являются:

- реализация разноуровневых дополнительных общеобразовательных программ для детей;
- разработка и реализация иных программ, в том числе в каникулярный период;
- вовлечение обучающихся и педагогических работников в проектную деятельность;
- организация внеучебной деятельности в каникулярный период;
- разработка и реализация соответствующих образовательных программ, в том числе для лагерей, организованных образовательными организациями в каникулярный период;
- повышение профессионального мастерства педагогических работников центра, реализующих дополнительные общеобразовательные программы.

В настоящее время, в 20-е г. XXI в., наше общество находится на этапе глобальной информатизации и компьютеризации. Поэтому возрастает потребность в специалистах с высоким уровнем владения информационными компетенциями, отвечающих социальному заказу по подготовке квалифицированных кадров в области программирования, а также обладающих высоким интересом к IT-сфере.

Одной из составляющих информационной компетентности является владение языком программирования. Встаёт вопрос о выборе языка программирования, который отвечает современным требованиям к написанию программ, служит основой для дальнейшего развития и совершенствования программистских компетенций.

Какой язык наиболее отвечает современному этапу развития процесса информатизации общества?

Для определения «популярности» языка программирования существует несколько рейтингов. Опишем кратко основные из них. Рейтинг TIOBE Index представляет собой анализ результатов поисковых запросов, содержащих название языка. В результате на первые позиции выходят те языки, названия которых чаще всего встречаются в поисковых запросах таких систем, как Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo!, Bing, Amazon. Такой расчёт производится ежемесячно: так, по результатам рейтинга за август 2021 г. наиболее популярным языком является C. На второй и третьей позициях находятся Python и Java соответственно (https://www.tiobe.com/tiobe-index/).

Язык программирования Python был представлен в 1990 г. Гвидо ван Россумом. В основе лежал язык ABC, который разрабатывался в центре математики и информатики в Нидерландах. Изначально в языке не была реализована концепция объектно-ориентированного программирования (ООП). В феврале 1991 г. был опубликован исходный текст языка. В него уже были заложены принципы ООП. Версия Python 2.0 была выпущена в 2000 г. В 2008 г. вышла версия Python 3.0, которая не полностью поддерживает вторую версию языка. Версия Python 3.8 вышла 14 октября 2019 г.

Целью представленной здесь дополнительной общеобразовательной программы по тематическому направлению «Программирование на языке Python» является изучение

основ программирования на языке Python, основных приёмов написания программ на современном языке программирования, развитие алгоритмического мышления учащихся, творческих способностей, аналитических и логических компетенций.

Важно!

Программа рассчитана на учащихся в возрасте от 12 до 15 лет, не требует предварительных знаний и входного тестирования.
Занятия проводятся в группах до 12 человек, продолжительность занятия 45 минут, общая продолжительность программы 144 часа.

Нормативная база

Конституция Российской Федерации (принята всенародным голосованием 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020) — URL: http://www.consultant.ru/document/cons_doc_LAW_28399/ (дата обращения: 10.03.2021).

Федеральный закон от 29.12.2012 № 273-ФЗ (ред. от 31.07.2020) «Об образовании в Российской Федерации» (с изм. и доп., вступ. в силу с 01.09.2020) — URL: http://www.consultant.ru/document/cons_doc_LAW_140174 (дата обращения: 28.09.2020).

Паспорт национального проекта «Образование» (утв. президиумом Совета при Президенте РФ по стратегическому развитию и национальным проектам, протокол от 24.12.2018 № 16) — URL: http://do.sev.gov.ru/images/document/Pasport_naciona_proekta_Jbrazovanie_compressed.pdf (дата обращения: 10.03.2021).

Государственная программа Российской Федерации «Развитие образования» (утв. постановлением Правительства РФ от 26.12.2017 № 1642 (ред. от 15.03.2021) «Об утверждении государственной программы Российской Федерации “Развитие образования”» — URL: http://www.consultant.ru/document/cons_doc_LAW_286474 (дата обращения: 12.05.2021).

Стратегия развития воспитания в Российской Федерации на период до 2025 года (утв. распоряжением Правительства РФ от 29.05.2015 № 996-р «Об утверждении Стратегии развития воспитания в Российской Федерации на период до 2025 года») — URL: http://www.consultant.ru/document/cons_doc_LAW_180402/ (дата обращения: 10.03.2021).

Профессиональный стандарт «Педагог (педагогическая деятельность в сфере дошкольного, начального общего, основного общего, среднего общего образования) (воспитатель, учитель)» (ред. от 16.06.2019 г.) (приказ Министерства труда и социальной защиты РФ от 18.10.2013 № 544н, с изм., внесёнными приказом Министерства труда и соцзащиты РФ от 25.12.2014 № 1115н и от 05.04.2016 № 422н) — URL: https://www.garant.ru/products/ipo/prime/doc/70435556/ (дата обращения: 10.03.2021).

Профессиональный стандарт «Педагог дополнительного образования детей и взрослых» (приказ Министерства труда и социальной защиты РФ от 05.05.2018 № 298н «Об утверждении профессионального стандарта «Педагог дополнительного образования детей и взрослых») — URL: https://profstandart.rosmintrud.ru/obshchiy-informatsionnyy-blok/natsionalnyy-reestr-professionalnykh-standartov/reestr-professionalnykh-standartov/index.php?ELEMENT_ID=48583 (дата обращения: 10.03.2021).

Федеральный государственный образовательный стандарт основного общего образования (утв. приказом Министерства образования и науки Российской Федерации от 17.12.2010 № 1897) (ред. от 21.12.2020) — URL: https://fgos.ru (дата обращения: 10.03.2021).

Федеральный государственный образовательный стандарт среднего общего образования (утв. приказом Министерства образования и науки Российской Федерации от 17.05.2012 № 413) (ред. от 11.12.2020) — URL: https://fgos.ru (дата обращения: 10.03.2021).

Методические рекомендации по созданию и функционированию детских технопарков «Кванториум» на базе общеобразовательных организаций (утв. распоряжением Министерства просвещения Российской Федерации от 12.01.2021 № Р-4) — URL: http://www.consultant.ru/document/cons_doc_LAW_374695/ (дата обращения: 10.03.2021).

Методические рекомендации по созданию и функционированию центров цифрового образования «IT-куб» (утв. распоряжением Министерства просвещения Российской Федерации от 12.01.2021 № Р-5) — URL: http://www.consultant.ru/document/cons_doc_LAW_374572/ (дата обращения: 10.03.2021).

Методические рекомендации по созданию и функционированию в общеобразовательных организациях, расположенных в сельской местности и малых городах, центров образования естественнонаучной и технологической направленностей («Точка роста») (утверждены распоряжением Министерства просвещения Российской Федерации от 12.01.2021 № Р-6) — URL: http://www.consultant.ru/document/cons_doc_LAW_374694/ (дата обращения: 10.03.2021).

Основные понятия и термины

Справочник

«IT-куб» — центр образования детей по программам, направленным на ускоренное освоение актуальных и востребованных знаний, навыков и компетенций в сфере информационных технологий.

Универсальные учебные действия (УУД) — совокупность способов действий обучающегося, которая обеспечивает его способность к самостоятельному усвоению новых знаний, т. е. способность субъекта к саморазвитию и самосовершенствованию путём сознательного и активного присвоения нового социального опыта.

Язык программирования — формальный язык, представляющий собой набор формальных правил, по которым пишут компьютерные программы.

Python — язык программирования высокого уровня, применяемый для разработки самостоятельных программ, а также для создания прикладных сценариев в самых разных областях применения.

Оператор — конструкция языка, определяющая команду (набор команд) языка программирования, задающая выполнение действий.

Условный оператор — оператор, который используется для выбора выполнения той или иной последовательности действий в зависимости от истинности или ложности некоторого условия.

Оператор цикла — оператор, который выполняет одну и ту же последовательность действий несколько раз; количество повторений либо задано, либо зависит от истинности или ложности некоторого условия.

Список — упорядоченная изменяемая последовательность элементов различного типа.

Кортеж — упорядоченная неизменяемая последовательность элементов различного типа.

Справочник

Вспомогательный алгоритм — алгоритм, выполняющий некоторую законченную подзадачу, как правило, создаётся для многократного выполнения; в основном алгоритме вызывается по имени. В языке Python может реализовываться в виде функции.

Подходы к структурированию материалов

Содержание обучения по данной программе может быть представлено следующими разделами.

1. Знакомство со средой программирования Python. Переменные.
2. Первые программы на языке Python, основные операторы.
3. Условный оператор if.
4. Циклы в языке Python.
5. Списки в языке Python.
6. Работа со строками в Python.
7. Работа с функциями в Python.
8. Кортежи в языке Python.

Для каждого раздела подготовлены лабораторные работы, включающие: необходимый теоретический материал с примерами, практическую часть с описанием хода работы, указаниями и по выполнению и контрольными вопросами. Также имеются дидактические материалы общей направленности, которые можно использовать при подготовке преподавателей и учащихся к занятиям, при выполнении лабораторных работ.

Материально-техническая база центров цифрового образования детей «IT-куб»

Для организации работы центра «IT-куб» в распоряжении «Об утверждении методических рекомендаций по созданию и функционированию центров цифрового образования “IT-куб” от 12.02.2021 рекомендуется следующее оборудование лаборатории.

Ноутбук тип 1	Форм-фактор: ноутбук. Жесткая неотключаемая клавиатура. Русская раскладка клавиатуры. Диагональ экрана: не менее 15,6 дюйма. Разрешение экрана: не менее 1920 × 1080 пикселей. Количество ядер процессора: не менее 4. Количество потоков: не менее 8. Базовая тактовая частота процессора: не менее 1 ГГц. Максимальная тактовая частота процессора: не менее 2,5 ГГц. Кэш-память процессора: не менее 6 Мбайт. Объём установленной оперативной памяти: не менее 8 Гбайт. Объём поддерживаемой оперативной памяти (для возможности расширения): не менее 24 Гбайт. Объём накопителя SSD: не менее 240 Гбайт. Время автономной работы от батареи: не менее 6 часов. Вес ноутбука с установленным аккумулятором: не более 1,8 кг. Внешний интерфейс USB стандарта не ниже 3.0: не менее трёх свободных. Внешний интерфейс LAN (использование переходников не предусмотрено). Наличие модулей и интерфейсов (использование переходников не предусмотрено): VGA, HDMI. Беспроводная связь Wi-Fi: наличие с поддержкой стандарта IEEE 802.11n или современнее. Веб-камера. Манипулятор «мышь». Предустановленная операционная система с графическим пользовательским интерфейсом, обеспечивающая работу распространённых образовательных и общесистемных приложений
Веб-камера	Микрофон: наличие, автоматическая фокусировка: наличие
МФУ (принтер, сканер, копир)	Набор функций: принтер/сканер/копир. СНПЧ в составе устройства или СНПЧ, совместимая с МФУ в комплекте поставки. Печать цветных изображений. Максимальный формат печати: А3, с максимальным разрешением печати не хуже 4800 × 1200 dpi. Скорость печати: не менее 15 с./мин. Функция автоматической двусторонней печати. Функция печати без полей. Функция беспроводного подключения, как минимум WiFi и AirPrint. Дисплей для отображения информации. Поддержка ОС Windows, macOS, iOS, Android. Интерфейсы подключения USB, RJ45

Продолжение

Рабочее место обучающегося в составе	
Ноутбук тип 2	Форм-фактор: ноутбук. Жесткая неотключаемая клавиатура. Русская раскладка клавиатуры. Диагональ экрана: не менее 15,6 дюйма. Разрешение экрана: не менее 1920 × 1080 пикселей. Количество ядер процессора: не менее 4. Количество потоков: не менее 8. Базовая тактовая частота процессора: не менее 1 ГГц. Максимальная тактовая частота процессора: не менее 2,5 ГГц. Кэш-память процессора: не менее 6 Мбайт. Объём установленной оперативной памяти: не менее 8 Гбайт. Объём поддерживаемой оперативной памяти (для возможности расширения): не менее 24 Гбайт. Объём накопителя SSD: не менее 240 Гбайт. Время автономной работы от батареи: не менее 6 часов. Вес ноутбука с установленным аккумулятором: не более 1,8 кг. Внешний интерфейс USB стандарта не ниже 3.0: не менее трёх свободных. Внешний интерфейс LAN (использование переходников не предусмотрено). Наличие модулей и интерфейсов (использование переходников не предусмотрено): VGA, HDMI. Беспроводная связь Wi-Fi: наличие с поддержкой стандарта IEEE 802.11n или современнее. Веб-камера. Манипулятор «мышь». Предустановленная операционная система с графическим пользовательским интерфейсом, обеспечивающая работу распространённых образовательных и общесистемных приложений
Наушники	Тип: полноразмерные
Презентационное оборудование	
Моноблочное интерактивное устройство	Интерактивный моноблочный дисплей, диагональ экрана: не менее 65 дюймов, разрешение экрана: не менее 3840 × 2160 пикселей. Встроенная акустическая система. Количество одновременно распознаваемых касаний сенсорным экраном: не менее 20 касаний. Высота срабатывания сенсора экрана: не более 3 мм от поверхности экрана. Встроенные функции распознавания объектов касания (палец или безбатарейный стилус). Количество поддерживаемых безбатарейных стилусов одновременно: не менее 2 шт. Возможность использования ладони в качестве инструмента стирания либо игнорирования касаний экрана ладонью. Интегрированный датчик освещённости для автоматической коррекции яркости подсветки. Наличие функции графического комментирования поверх произвольного изображения, в том числе от физически подключённого источника видеосигнала. Интегрированные функции вывода изображений с экранов мобильных устройств (на платформе

Продолжение

	распространённых ОС), а также с возможностью интерактивного взаимодействия (управления) с устройством-источником. Интегрированный в пользовательский интерфейс функционал просмотра и работы с файлами основных форматов с USB-накопителей или сетевого сервера. Поддержка встроенными средствами дистанционного управления рабочих параметров устройства через внешние системы. Предустановленная операционная система с графическим пользовательским интерфейсом, обеспечивающая работу распространённых образовательных и общесистемных приложений. Интегрированные средства, обеспечивающие следующий функционал: создание многостраничных уроков с использованием медиаконтента различных форматов, создание надписей и комментариев поверх запущенных приложений; распознавание фигур и рукописного текста (русский, английский языки); наличие инструментов рисования геометрических фигур и линий; встроенные функции: генератор случайных чисел, калькулятор, экранная клавиатура, таймер, редактор математических формул; электронные математические инструменты: циркуль, угольник, линейка, транспортир; режим «белой доски» с возможностью создания заметок, рисования, работы с таблицами и графиками; импорт файлов форматов: PDF, PPT
Напольная мобильная стойка для интерактивных досок или универсальное настенное крепление	Совместимость с моноблочным интерактивным устройством. Максимальный вес, выдерживаемый креплением: не менее 60 кг
Дополнительное оборудование	
Доска магнитно-маркерная настенная	Тип: полимерная, сухостираемая
Флипчарт магнитно-маркерный на треноге	Размер рабочей области: не менее 700 × 1000 мм. Тип опоры: тренога
Комплект кабелей и переходников	Кабели, переходники для подключения и коммутации оборудования. Сетевой удлинитель для подключения оборудования к сети электропитания и др. (по выбору)
Учебная и методическая литература	Для реализации образовательных программ
Комплект комплектующих и расходных материалов	Для реализации образовательных программ
Мебель	
Комплект мебели	Учебная мебель: столы для всех учеников, стулья/кресла для всех учеников, пуфы. Мебель для педагога: стол, стул (кресло). Системы хранения: тумбы, шкафы, стеллажи (по выбору)

Рис. 1. Лаборатория в центре «IT-куб»

В центре «IT-куб» действуют несколько лабораторий (рис. 1), в том числе лаборатория для осуществления направления «Программирование на языке Python».

Лаборатория оборудована ноутбуками Asus, а также оснащена интерактивной доской, маркерной доской, МФУ.

На данном оборудовании могут выполняться лабораторные работы по программе «Программирование на языке Python», проводиться открытые занятия, защита проектов и т. д. С использованием презентационного оборудования преподаватели объясняют новый материал, приводят примеры работы программ и т. д. (рис. 2).

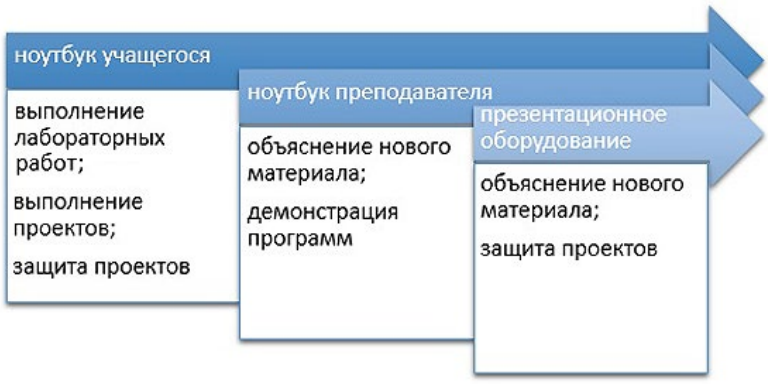


Рис. 2. Использование оборудования в центре «IT-куб» при организации занятий по программе «Программирование на языке Python»

Примерная рабочая программа дополнительного образования для организации работы по тематическому направлению «Программирование на языке Python» деятельности центра цифрового образования детей «IT-куб»

Важно!

Как было сказано ранее, целью программы по тематическому направлению «Программирование на языке Python» является изучение основ программирования на языке Python, основных приёмов написания программ на современном языке программирования, развитие алгоритмического мышления учащихся, творческих способностей, аналитических и логических компетенций.

Планируемые результаты освоения программы

Личностные результаты:

- формирование умения самостоятельной деятельности;
- формирование умения работать в команде;
- формирование коммуникативных навыков;
- формирование навыков анализа и самоанализа;
- формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные результаты:

- формирование понятий «алгоритм», «программа»;
- формирование понятий об основных конструкциях языка программирования Python, таких как оператор ветвления if, операторы цикла while, for, вспомогательные алгоритмы;
- формирование понятий о структурах данных языка программирования Python;
- формирование основных приёмов составления программ на языке программирования Python;
- формирование алгоритмического и логического стилей мышления.

Метапредметные результаты:

- формирование умения ориентироваться в системе знаний;
- формирование умения выбирать наиболее эффективные способы решения задач на компьютере в зависимости от конкретных условий;
- формирование приёмов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы;
- формирование умения распределять время;
- формирование умений успешной самопрезентации.

Тематическое планирование

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов		Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Оборудование
				Теор.	Практ.		
1	*Знакомство со средой программирования на языке Python. Переменные*	Знакомство со средой программирования на языке Python, изучение основных элементов интерфейса, запуск программы. Изучение понятий «переменная», «значение переменной»	Ознакомление со средой программирования на языке Python, изучение основных инструментов среды, изучение понятия «переменная», задание значения переменной	3	3	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
2	*Первые программы на языке Python, основные операторы*	Написание простых программ на языке программирования Python, знакомство с операторами присваивания, ввода/вывода данных, разработка программ, реализующих линейные алгоритмы на языке программирования Python	Ознакомление с основами написания программ на языке программирования Python, работа с операторами присваивания, ввода/вывода данных	3	3	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
3	*Условный оператор* `if`	Формат оператора ветвления `if` на языке программирования Python, разработка программ, реализующих условные алгоритмы	Ознакомление с условным оператором `if` на языке программирования Python	6	6	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов		Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Оборудование
				Теор.	Практ.		
4	*Циклы в языке Python*	Формат оператора цикла с предусловием `while`, оператора цикла с параметром `for` на языке программирования Python, разработка программ, циклические алгоритмы	Ознакомление с операторами цикла `for`, `while` языка программирования Python	5	5	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
5	*Решение задач по изученным темам*	Решение дополнительных задач по темам «Условный оператор `if`», «Циклы в языке Python»	Ознакомление с основными операторами языка программирования Python		10	Самостоятельное решение задач	Компьютер, проектор, интерактивная доска
6	*Контрольная работа*	Решение задач	Проверка полученных навыков по темам «Условный оператор `if`», «Циклы в языке Python»		4	Самостоятельное выполнение контрольных заданий	Компьютер, проектор, интерактивная доска
7	*Списки в языке Python*	Понятие «список» в языке программирования Python, создание списка, различные способы задания списка, вывод элементов списка на экран, основные функции по работе со списками в языке программирования Python	Ознакомление с понятием «список» в языке программирования Python	10	7	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

В содержание

8	*Работа со строками в языке Python*	Понятие «строка» в языке программирования Python, различные способы задания строк, основные функции по работе со строками в языке программирования Python	Ознакомление с понятием «строка» в языке программирования Python	8	5	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
9	*Решение задач по изученным темам*	Решение дополнительных задач по темам «Списки в языкке Python», «Работа со строками в языке Python»	Ознакомление с основными операторами языка программирования Python		10	Самостоятельное решение задач	Компьютер, проектор, интерактивная доска
10	*Контрольная работа*	Решение задач	Проверка полученных навыков по темам «Списки в языке Python», «Работа со строками в Python»		4	Самостоятельное выполнение контрольных заданий	Компьютер, проектор, интерактивная доска
11	*Работа с функциями в Python*	Вспомогательный алгоритм при разработке программ, понятие «функция» в языке программирования Python, описание функции, структура функции, обращение к функции в тексте программы, приёмы написания программ с использованием вспомогательных алгоритмов	Ознакомление с понятием «функция» в языке программирования Python, описание функции, основные приёмы структурного программирования	8	6	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов		Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Оборудование
				Теор.	Практ.		
12	*Кортежи в языке Python*	Понятие «кортеж» в языке программирования Python, создание кортежа, основные функции по работе с кортежами в языке программирования Python	Ознакомление с понятием «кортеж» в языке программирования Python	6	6	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
13	*Индивидуальное задание*	Разработка индивидуального или группового проекта на языке программирования Python	Создание проекта на языке программирования Python		22	Самостоятельная индивидуальная или групповая проектная деятельность	Компьютер, проектор, интерактивная доска
14	*Итоговые занятия*	Защита индивидуальных или групповых проектов, подведение итогов курса	Защита проекта		4	Самостоятельная индивидуальная или групповая проектная деятельность	Компьютер, проектор, интерактивная доска
	Итого:			144			

В содержание

Содержание и форма организации учебных занятий

Планы учебных занятий

1. Знакомство со средой программирования на языке Python. Переменные

Рекомендуемое количество часов на данную тему — 6.

Планируемые результаты:

предметные: получение навыков работы в среде программирования на языке Python, изучение основных инструментов среды, изучение понятия «переменная», задание значения переменной;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работа 1 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

2. Первые программы на языке Python, основные операторы

Рекомендуемое количество часов на данную тему — 6.

Планируемые результаты:

предметные: получение навыков создания первых программ в среде программирования на языке Python, изучение основных операторов языка Python, ввода/вывода данных, встроенных функций;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 2.1, 2.2 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

3. Условный оператор if

Рекомендуемое количество часов на данную тему — 12.

Планируемые результаты:

предметные: получение навыков использования условного оператора if в среде программирования на языке Python, разработка программ, реализующих разветвляющиеся алгоритмы;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 3.1, 3.2 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

4. Циклы в языке Python

Рекомендуемое количество часов на данную тему — 10.

Планируемые результаты:

предметные: получение навыков использования операторов цикла в среде программирования на языке Python, разработка программ, реализующих циклические алгоритмы;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 4.1, 4.2 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

5. Списки в языке Python

Рекомендуемое количество часов на данную тему — 17.

Планируемые результаты:

предметные: получение навыков использования списков в среде программирования на языке Python, разработка программ, реализующих работу со структурами данных;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 5.1, 5.2, 5.3 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

6. Работа со строками в языке Python

Рекомендуемое количество часов на данную тему — 13.

Планируемые результаты:

предметные: получение навыков использования строк в среде программирования на языке Python, разработка программ, реализующих работу со строковыми данными;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 6.1, 6.2 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

7. Работа с функциями в Python

Рекомендуемое количество часов на данную тему — 14.

Планируемые результаты:

предметные: получение навыков использования функций в среде программирования на языке Python, разработка программ, реализующих работу со вспомогательными алгоритмами;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работы 7.1, 7.2 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

8. Кортежи в языке Python

Рекомендуемое количество часов на данную тему — 12.

Планируемые результаты:

предметные: получение навыков использования кортежей в среде программирования на языке Python, разработка программ, реализующих работу со структурами данных;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: работа 8 — изучение теоретического материала лабораторной работы, выполнение лабораторной работы.

Дидактические (справочные) материалы

Работа со средой программирования на языке Python

Важно!

В данном пособии мы опираемся на Python версии 3.x. Если в вашей операционной системе уже установлен Python версии 2.x, то вы можете установить эти две версии одновременно в одной операционной системе и по выбору использовать ту или иную версию в зависимости от потребностей.
Далее рассматривается установка Python версии 3.x в операционных системах семейства Windows.

Для установки нужно перейти в браузере на официальную страницу Python: http://www.python.org/download/ — и загрузить последнюю версию Python 3.x. Установка производится стандартным способом: так же, как и любых других программ Windows.

Для того чтобы работать с Python из командной строки Windows, необходимо задать значение переменной PATH. Это можно сделать при установке Python, выбрав пункт *Add Python 3.x to PATH* (рис. 3).

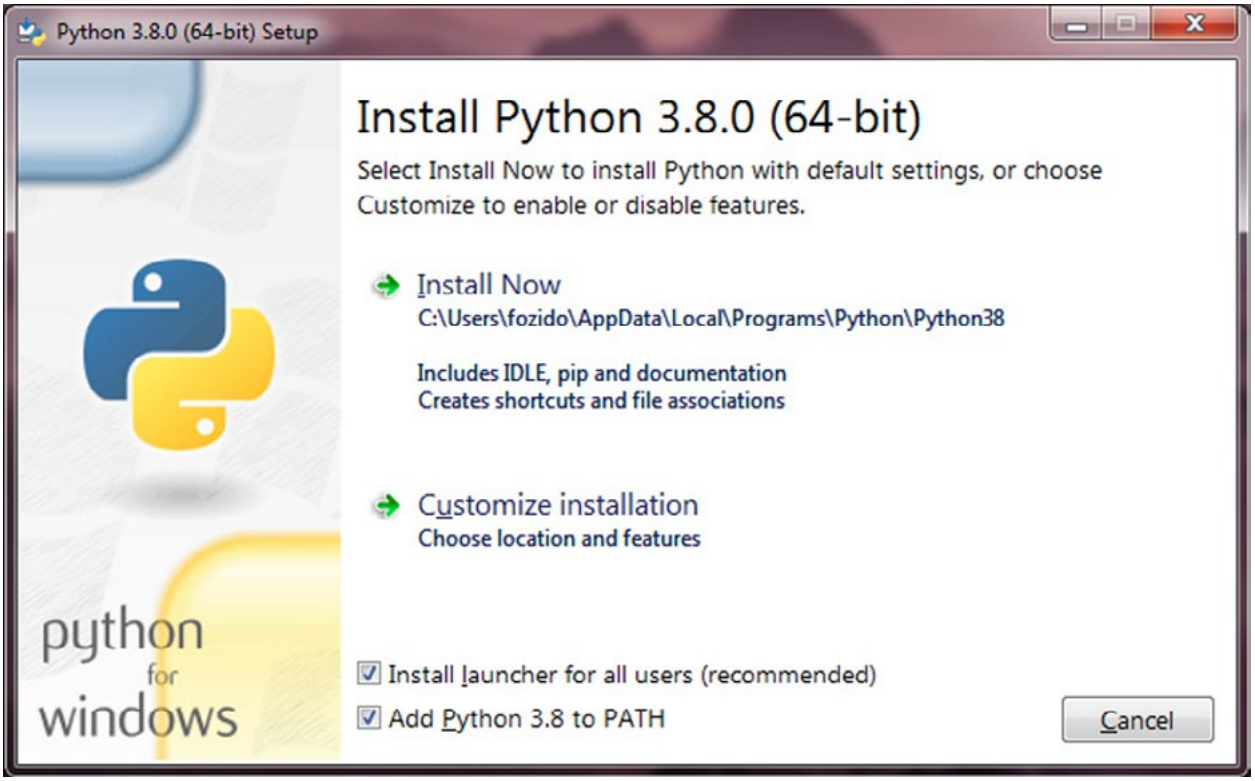


Рис. 3. Окно установки Python 3.x для Windows

Для настройки переменной PATH сначала можно в контекстном меню значка компьютера выбрать пункт *Свойства* или нажать кнопку *Пуск*, далее выбрать команду *Панель управления — Система и безопасность — Система*. Потом нажать кнопку слева *Дополнительные параметры системы*, затем выбрать вкладку *Дополнительно*. Внизу нажать кнопку *Переменные среды*, в разделе *Системные переменные* найти переменную PATH, выбрать её и нажать кнопку *Редактировать*. (В разных версиях Windows набор команд может немного различаться.)

Далее в конце строки в поле *Значение переменной* набрать: *C:\Python3x*, где *x* — это номер версии Python. В показанном на рисунке 3 примере — версия Python 3.8, значит, нужно набрать: *C:\Python38*.

Если ранее значение переменной PATH было *%SystemRoot%\system32;*, теперь оно примет вид *%SystemRoot%\system32;C:\Python3x*. Если все шаги установки выполнены правильно, то можно запускать интерпретатор из командной строки в Windows. Чтобы открыть консольное окно с командной строкой в Windows, нажмите кнопку *Пуск* и выберите *Выполнить*. В появившемся диалоговом окне наберите cmd и нажмите клавишу *Enter*. Далее набираем слово python и проверяем, нет ли ошибок (рис. 4).

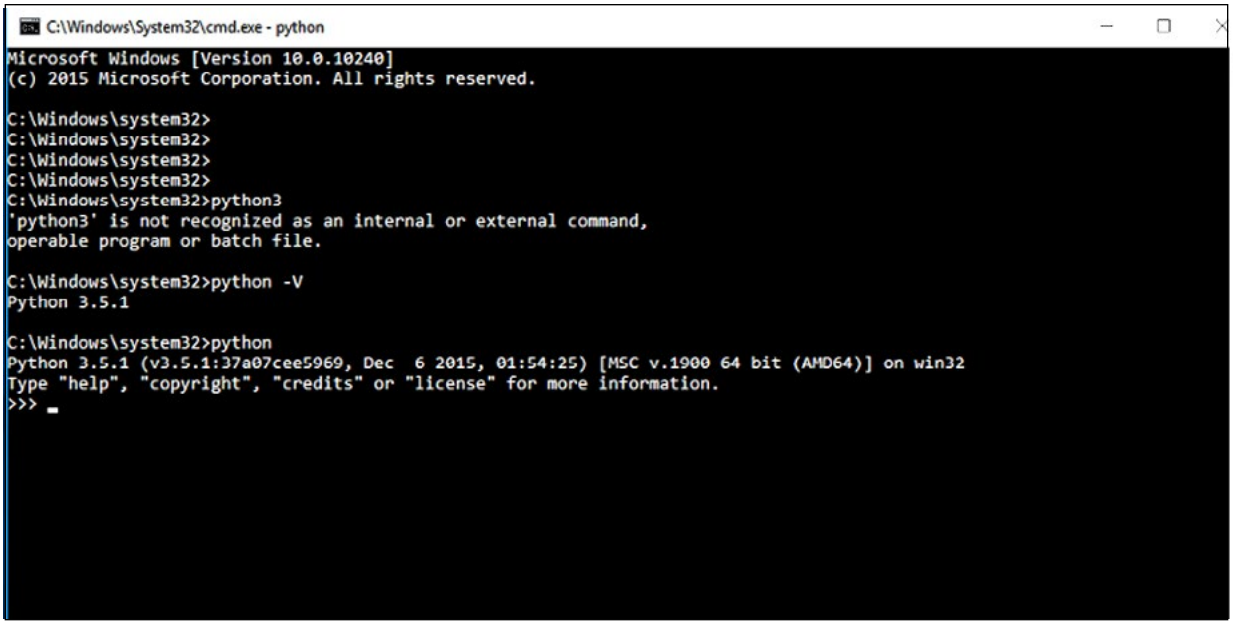


Рис. 4. Работа Python 3.x в командной строке Windows

Альтернативный способ: использовать IDLE — интегрированную среду разработки и обучения на языке Python, которая устанавливается вместе с Python. Для этого выбрать: *Пуск — Программы — Python 3.x — IDLE (Python 3.x)* (рис. 5).

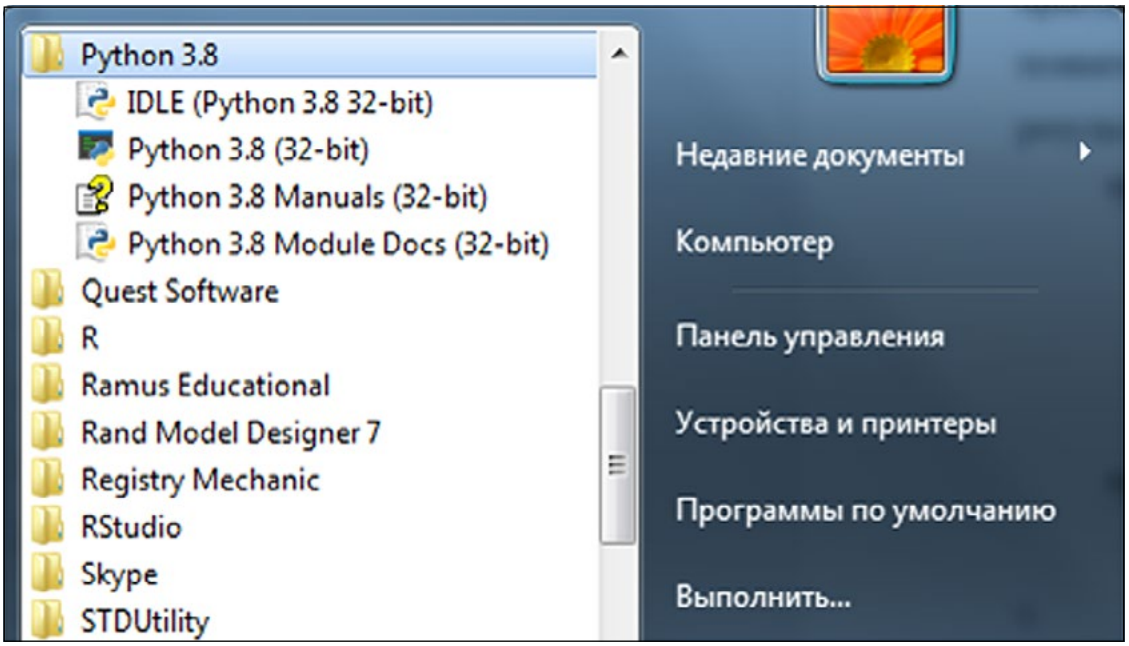


Рис. 5. Запуск Python 3.x

Как только будет запущен Python 3.x, на экране появится приглашение к вводу — символ >>> в начале строки, где вы можете что-то набирать. Это называется командной строкой интерпретатора Python 3.x (рис. 6). Попробуем ввести в окне приглашения print('Привет') и нажать клавишу *Enter*. В результате должно появиться слово «Привет» (рис. 7). Как видим, синтаксис языка подсвечивается, результат работы программы выделяется синим цветом.

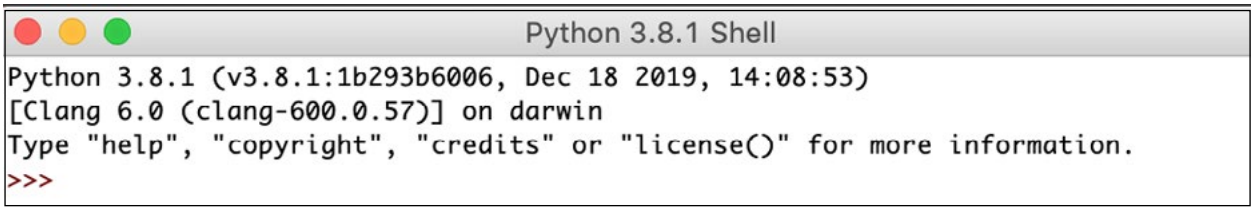


Рис. 6. Окно IDLE (Python GUI)

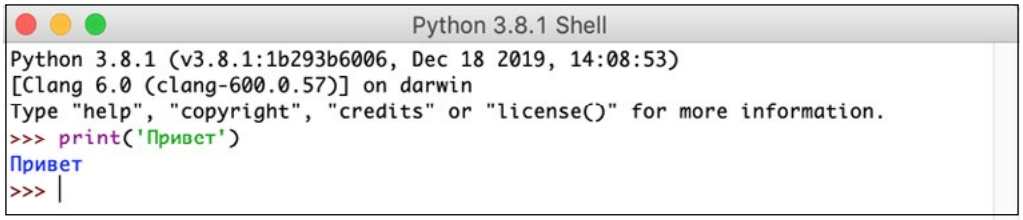


Рис. 7. Окно IDLE (Python GUI)

Стоит заметить, что Python 3.x выдаёт результат работы немедленно. Но не всегда удобно с точки зрения работы с разными программами набирать их в командной строке интерпретатора. Очень важной составляющей является сохранение программы в файле, чтобы потом можно было запускать её неограниченное количество раз.

На практике в современном программировании используются различные редакторы кода. Стоит очень ответственно подойти к выбору редактора. Хороший редактор поможет вам легко писать программы на Python 3.x, делать это понятно, быстро, комфортно, поможет быстрее реализовать ваши идеи.

В настоящее время в кругу разработчиков чётко определились основные требования к редактору кода, такие как: умный поиск, подсветка синтаксиса, обработчик ошибок, поддержка баз данных и т. д.

Справочник

IDE (интегрированная среда разработки) — программа, предназначенная для разработки программного обеспечения. Стандартные инструменты IDE:

- текстовый редактор для обработки кода;
- транслятор;
- средства сборки;
- средства отладки.

IDE, поддерживающие множество языков программирования и содержащие другие специализированные дополнительные функции, занимают внушительное место в памяти и, как правило, работают медленно.

Можно сформулировать требования к среде разработки программ на Python 3.x, которые упрощают разработку:

- возможность сохранить работу и открыть её в том же состоянии;
- запуск кода в среде программирования;
- возможность отладки;
- подсветка синтаксиса — для удобства визуального восприятия программы;
- автоматическое форматирование кода, например, распознавание двоеточия в конце оператора `for` или `while`, в результате чего дальнейший текст будет автоматически набираться на следующей строке, и др.

Выделяют два вида сред разработки программ на языке Python:

1) среды разработки и редакторы кода с поддержкой языка Python;

2) специализированные редакторы кода и IDE для Python.

К первому виду можно отнести такие продукты, как Sublime Text, ATOM, GNU Emacs, Vim, Eclipse, Visual Studio Code.

Sublime Text (рис. 8) — редактор кода, поддерживает многие языки программирования, в том числе и Python. Работает на всех современных платформах, в том числе в Windows. Имеет пользовательские расширения (пакеты).

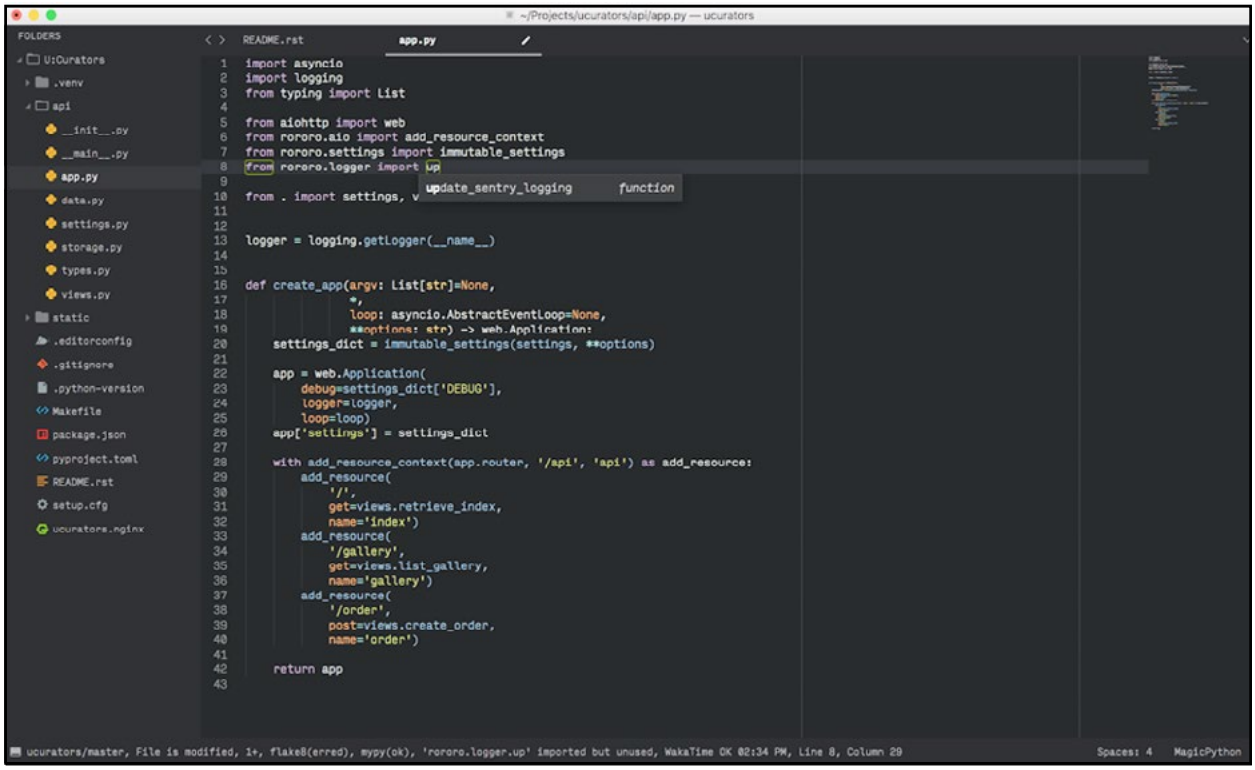


Рис. 8. Интерфейс Sublime Text

В таблице 1 отмечены основные достоинства и недостатки Sublime Text.

Таблица 1

Достоинства и недостатки редактора кода Sublime Text

Достоинства	Недостатки
Быстрая навигация, API плагинов на Python, одновременное редактирование, высокая настраиваемость. Поддерживает много языков программирования. Считается достаточно лёгким и быстрым редактором кода	Пакеты Sublime Text написаны на Python, и для установки редактора часто требуется выполнить скрипты Python непосредственно в Sublime Text. Программа не является бесплатной, есть период пробной версии. Установка расширений требует дополнительных усилий. В редакторе отсутствует прямая поддержка для выполнения или отладки кода из редактора

Atom (рис. 9) — бесплатный текстовый редактор с открытым исходным кодом для Windows с поддержкой плагинов, написанных на JavaScript. Редактор основан на Electron (ранее известный как Atom Shell) — фреймворке кроссплатформенной разработки с использованием Chromium и io.js. Язык Python поддерживается расширением, которое можно установить при запуске Atom.

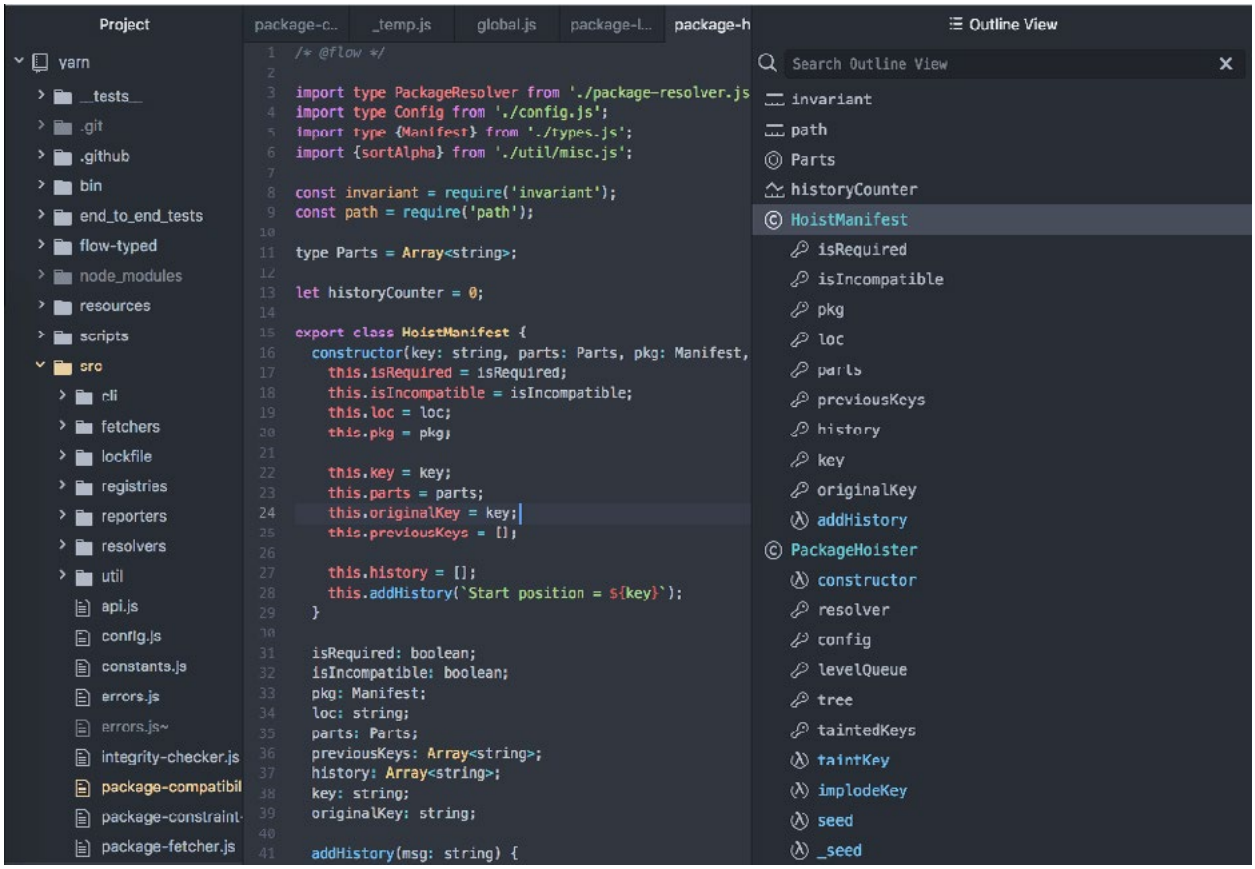


Рис. 9. Интерфейс Atom

В таблице 2 отмечены основные достоинства и недостатки Atom.

Таблица 2

Достоинства и недостатки редактора кода Atom

Достоинства	Недостатки
Достаточно лёгкий и быстрый редактор. Поддерживает много языков программирования, каждый язык работает при наличии предварительно установленного компилятора/интерпретатора. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, дополнительно подключаемые плагины	Редактор построен на Electron, это означает, что он работает как процесс JavaScript, а не как отдельное приложение

Visual Studio Code (VS-Code) (рис. 10) — полнофункциональный редактор кода с открытым исходным кодом, доступный для различных платформ, в том числе и для Windows. Достаточно универсальный, быстрый, масштабируемый. Как и редактор кода Atom, VS-Code построен на Electron.

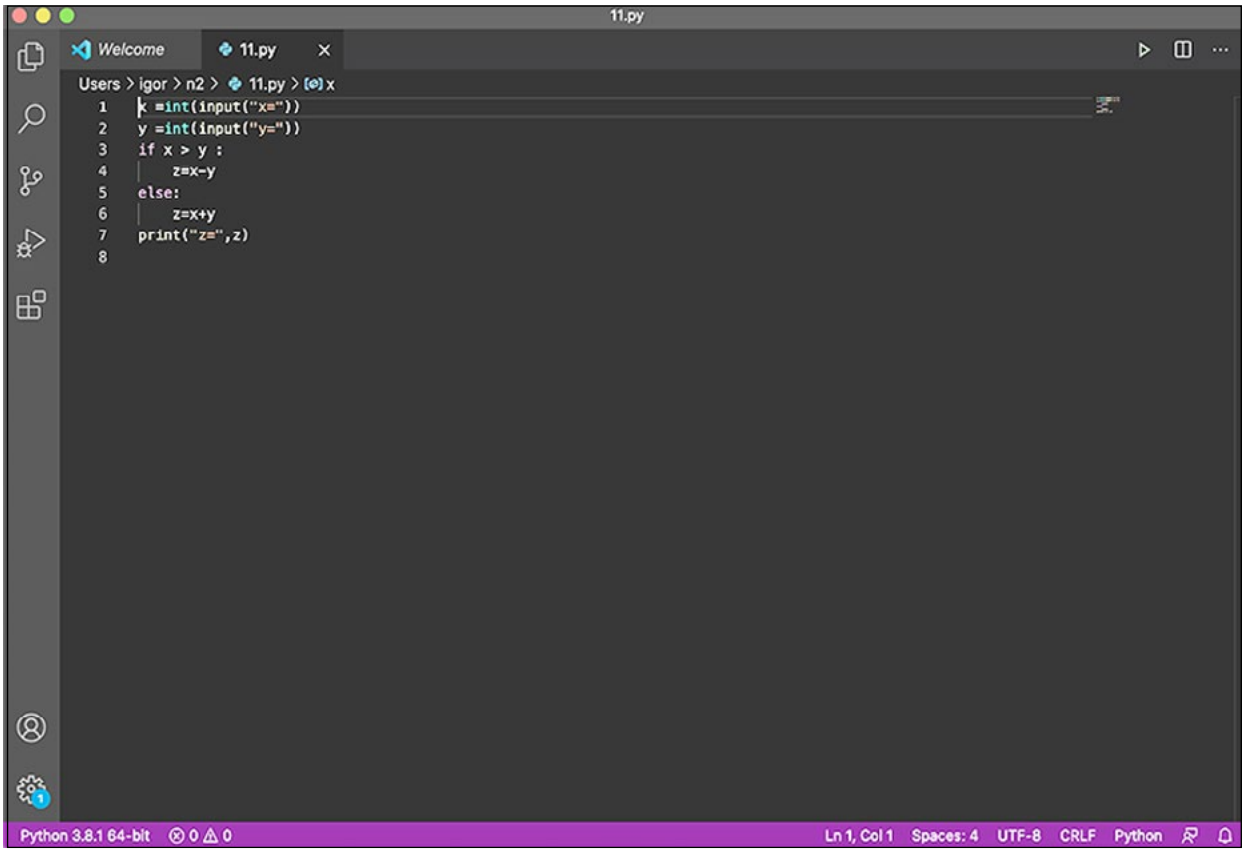


Рис. 10. Интерфейс Visual Studio Code

В таблице 3 отмечены основные достоинства и недостатки Visual Studio Code.

Таблица 3

Достоинства и недостатки редактора кода Visual Studio Code

Достоинства	Недостатки
Достаточно простой и лёгкий редактор кода для кроссплатформенной разработки веб-приложений. Содержит отладчик, инструменты для работы с Git, подсветку синтаксиса, дополнительно подключаемые плагины, средства для рефакторинга. В программе хорошо представлены возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Доступен для Windows, macOS и Linux	Редактор построен на Electron, это означает, что он работает как процесс JavaScript, а не как отдельное приложение

Второй вид программ — это специализированные редакторы и IDE для Python 3.x.

IDE PyCharm компании Jetbrains (рис. 11) — одна из лучших полнофункциональных выделенных IDE для Python.

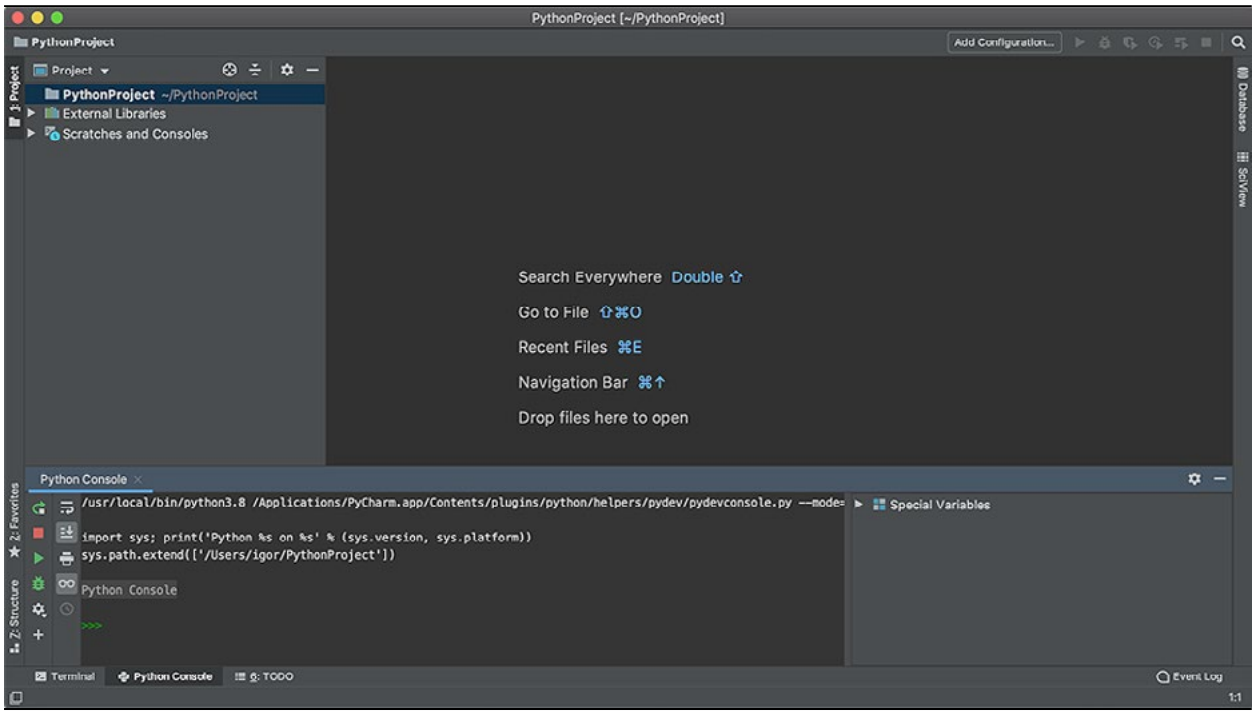


Рис. 11. Интерфейс IDE PyCharm

В таблице 4 отмечены основные достоинства и недостатки IDE PyCharm.

Таблица 4

Достоинства и недостатки IDE PyCharm

Достоинства	Недостатки
Все возможности IDE. Управление версиями и проектами, нативный запуск и написание кода, подсветка синтаксиса, автозаполнение кода Python и т. д. Доступна бесплатная версия с открытым исходным кодом (Community) для Windows, macOS и Linux	Медленная загрузка среды разработки, требует более углублённой настройки для проектов

IDE Spyder (рис. 12) — программа с открытым исходным кодом, создана для разработок, связанных с анализом данных. Поставляется вместе с дистрибутивом диспетчера пакетов Anaconda.

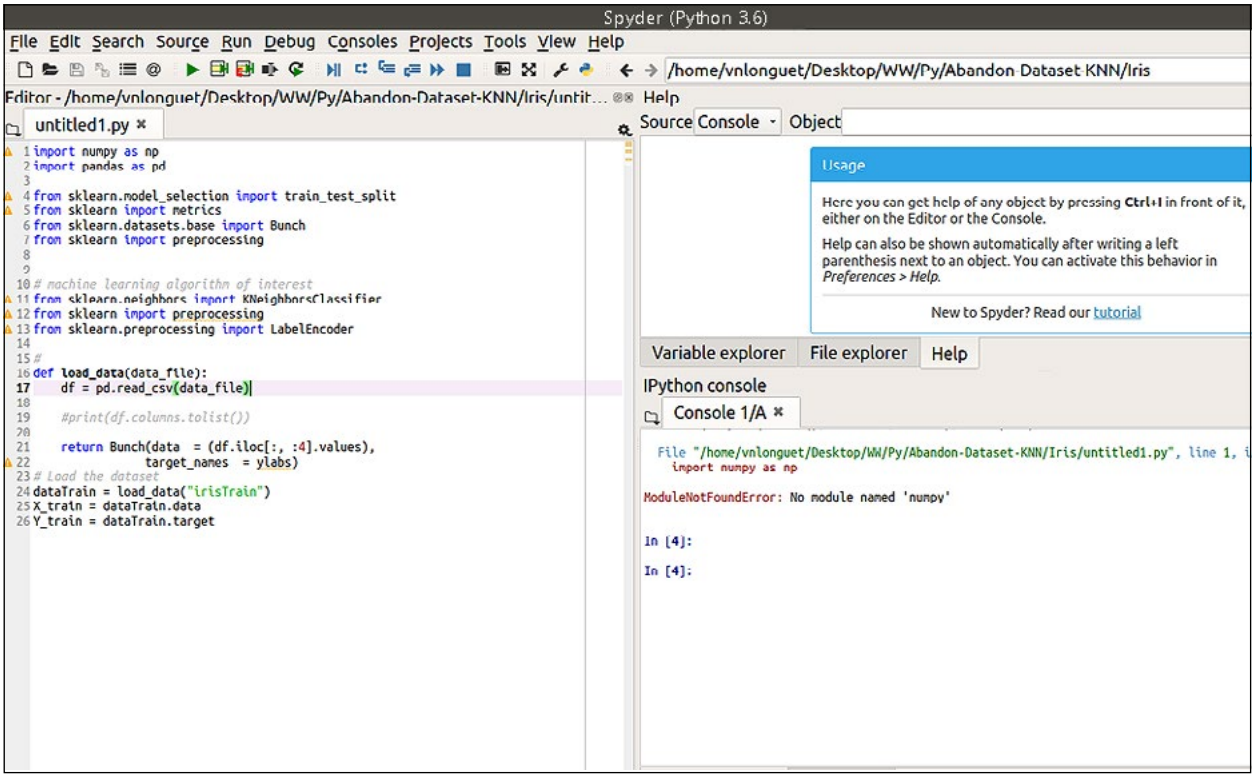


Рис. 12. Интерфейс IDE Spyder

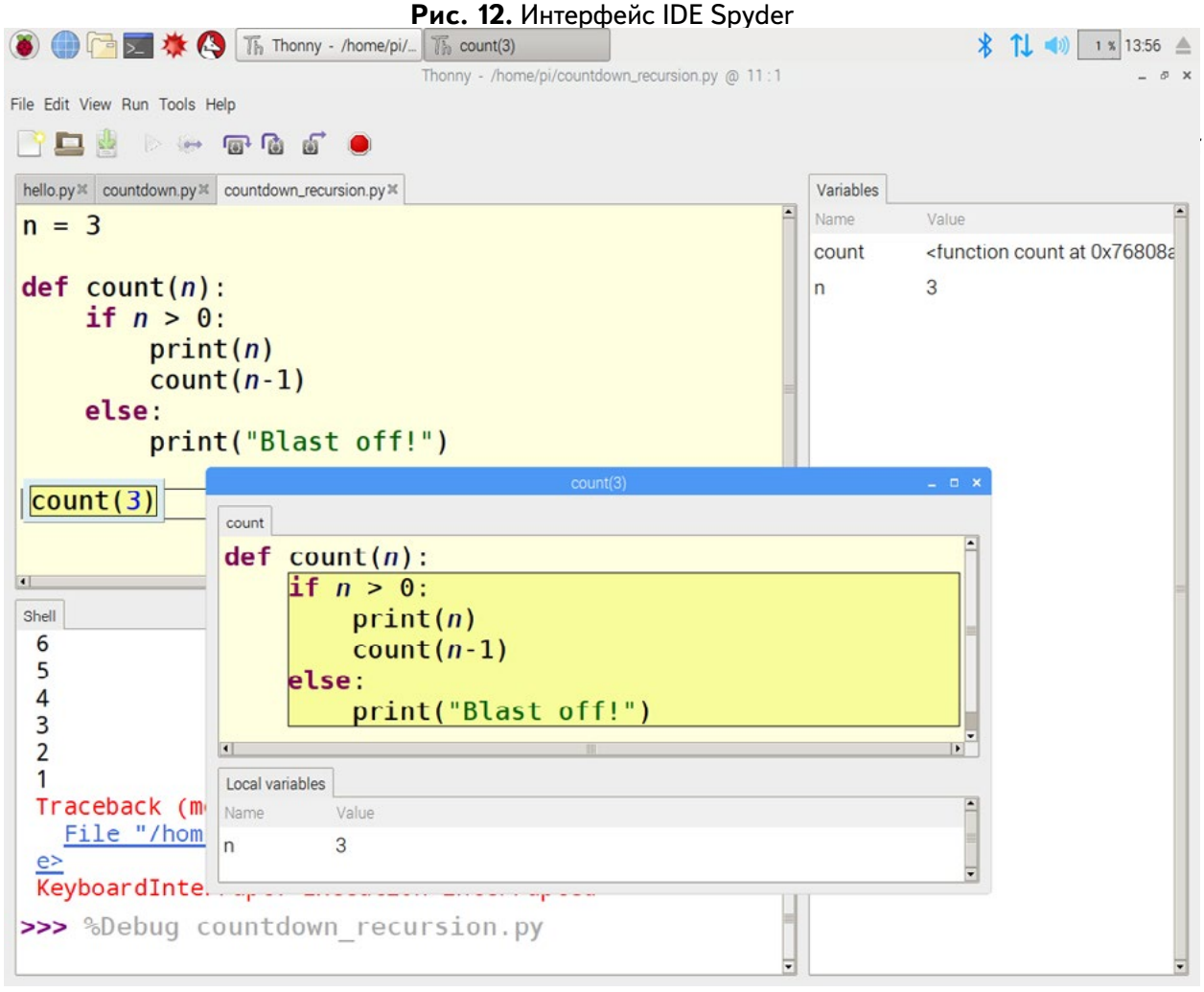


Рис. 13. Интерфейс IDE Thonny

В таблице 6 отмечены основные достоинства и недостатки IDE Thonny.

Таблица 6

Достоинства и недостатки IDE Thonny

Достоинства	Недостатки
Поддержка номеров строк, пошаговое выполнение выражений без точек останова, живые переменные во время отладки, поэтапное вычисление выражений, отдельные окна для выполнения вызовов функций, возможность регистрировать действия пользователя для воспроизведения или анализа процесса программирования	Не поддерживает расширенный функционал для опытных разработчиков, проект достаточно молодой, может содержать ошибки

Основные возможности PyCharm

Рассмотрим более подробно продукт компании JetBrains — IDE PyCharm.

PyCharm — это интегрированная среда разработки (IDE), используемая для написания программ на Python. В ней есть инструменты для анализа кода, графический отладчик, встроенное модульное тестирование и поддержка веб-разработки с современными фреймворками.

Для установки PyCharm сначала нужно установить интерпретатор Python 3.x. Затем на официальном сайте компании JetBrains выбрать продукт PyCharm: https://www.jetbrains.com/pycharm/. Вам будут предложены к загрузке две версии PyCharm: **Community** или **Professional**. Professional является платной версией с полным набором функций и поддержкой фреймворков. Она идеально подходит для профессиональной разработки. Версия Community бесплатная, обладает базовыми возможностями.

Важной особенностью продукта PyCharm является продвинутый редактор кода, который предоставляет поддержку для Python 3.x, JavaScript, TypeScript, CSS, HTML и других популярных языков. Этот редактор помогает быстро определять синтаксис языка, проводить проверку ошибок, вносить корректировки в код программы.

В программе также реализована функция умного поиска, которая может проводить поиск не только в коде программы и событиях, но и в окне инструментов PyCharm. С помощью поиска быстро переходим к методу, тестированию, использованию, реализации и т. д.

IDE PyCharm предлагает обновление кода (безопасное удаление и переименование, метод извлечения, вводные переменная и метод и другие виды рефакторинга).

Встроенные инструменты разработки содержат большую коллекцию инструментов PyCharm, таких как: встроенный терминал, инструменты для работы с базами данных, возможность удалённой разработки с удаленными интерпретаторами, терминал SSH, интеграция с Docker, инструменты контролей версии, профайлер Python, интегрированный отладчик, запуск тестирования и т. д. IDE PyCharm может запустить консоль Python REPL, что даёт массу преимуществ, таких как мгновенная проверка синтаксиса с дополнительными проверками, сопоставлением скобок и кавычек и, конечно, автозаполнением кода.

Важным фактором выбора PyCharm становится и то, что эта среда обеспечивает специальную поддержку библиотек, таких как Anaconda, Numpy, Matplotlib и другие, предоставляя пользователю глубокое понимание кода, интерактивные графики, просмотр массивов и многое другое.

Немаловажным фактором в разработке программ на языке Python является выбор операционной системы. Программный продукт PyCharm можно установить на Windows, macOS и Linux, используя один лицензионный ключ. Данная IDE является полностью кроссплатформенной. Написание кода и интерфейс программы будут выглядеть абсолютно одинаково во всех операционных системах. Пользовательский интерфейс и его настройка предоставляет каждому удобное рабочее пространство с настраиваемыми цветовыми схемами и «горячими клавишами». PyCharm поддерживает дополнительные плагины, интеграции с различными инструментами и фреймворками, редактором обновлений, таким как эмуляция Vim.

IDE PyCharm поддерживает разработку приложений с помощью популярных фреймворков Django, Flask и Pyramid.

Фре́ймворк — программная платформа, которая определяет структуру программной системы и облегчает разработку и объединение разных модулей большой программной системы.

Django — свободный фреймворк с открытым кодом. С его помощью можно добавлять большинство стандартных функций в виде пакета. Это такие функции, как аутентификация, URL-маршрутизация, миграция схемы данных и т. п. Django использует ORM для сопоставления объектов с таблицами баз данных. Один и тот же код работает с разными базами данных. Django работает с такими СУБД, как PostgreSQL, MySQL, SQLite и Oracle.

Pyramid — фреймворк с открытым кодом, обеспечивает максимальные возможности при минимальных затратах времени и ресурсов. Работает как с большими, так и с малыми

приложениями. Ещё отметим такие функции, как генерация URL, масштабируемая конфигурация, гибкая схема аутентификации, понятная техническая документация.

Flask — это микрофреймворк, предоставляется по лицензии BSD. Имеет модульный дизайн, благодаря этому фреймворк можно применять для выполнения разных задач. Следующие функции являются стандартными: встроенный сервер и отладчик, шаблоны Jinja2, поддержка WSGI 1.0, Unicode, подключение к любой ORM.

Но стоит заметить, что все эти расширенные функции включены в профессиональную версию продукта.

IDE PyCharm полностью поддерживает HTML и HTML5, CSS, JavaScript и XML. Всё это реализуется через подключение к IDE плагинов (рис. 14). Поддержка других языков и фреймворков также может быть добавлена через плагины.

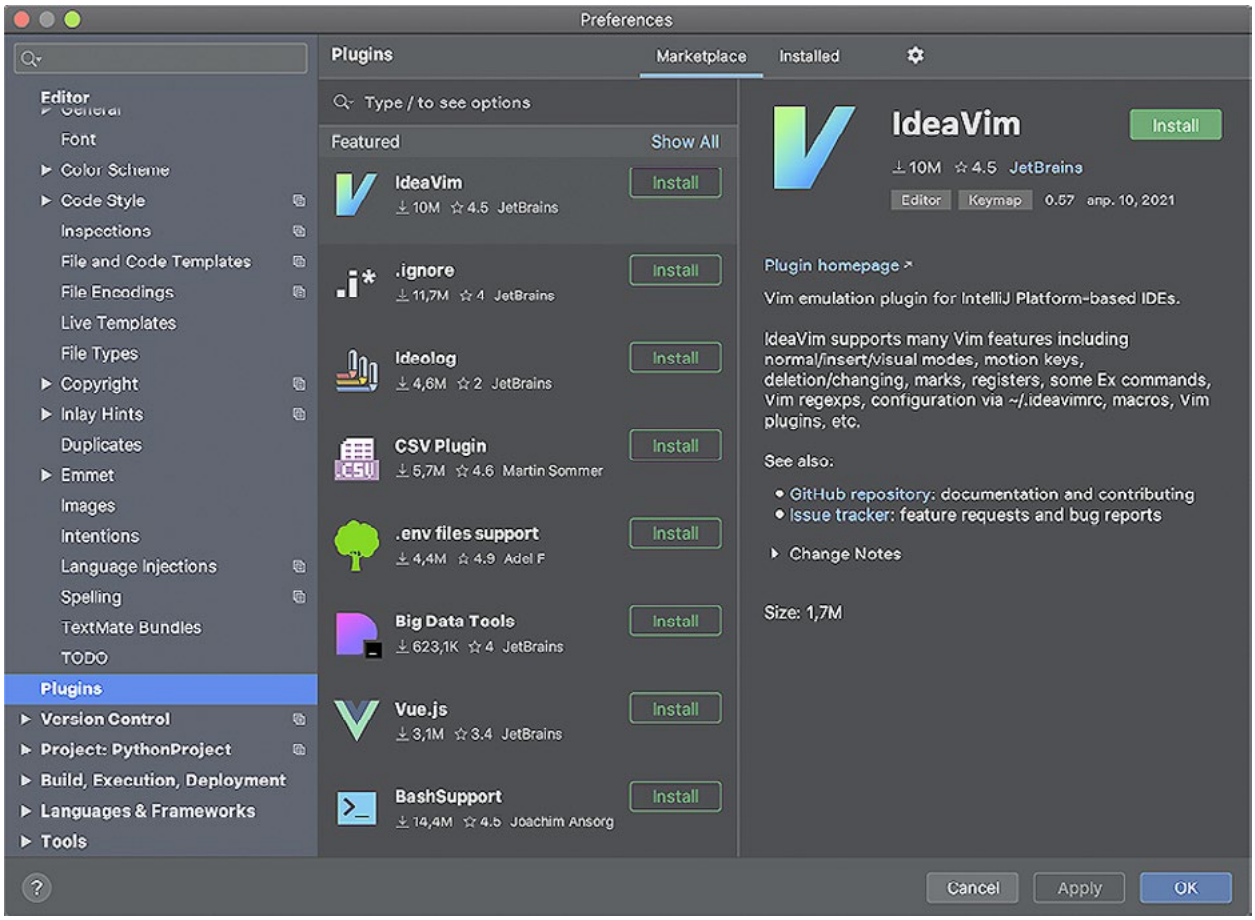


Рис. 14. Добавление плагинов к PyCharm

Одним из больших преимуществ PyCharm является то, что она является кроссплатформенной средой разработки, которая работает в Windows, macOS и Linux.

Далее познакомимся более подробно с интерфейсом программы PyCharm и рассмотрим создание нашего первого проекта. Отметим, что вся работа в PyCharm ведётся в контексте проекта. После запуска программы нас встречает окно приветствия, которое предоставляет возможность выбора существующих проектов на компьютере или создания нового проекта (рис. 15).

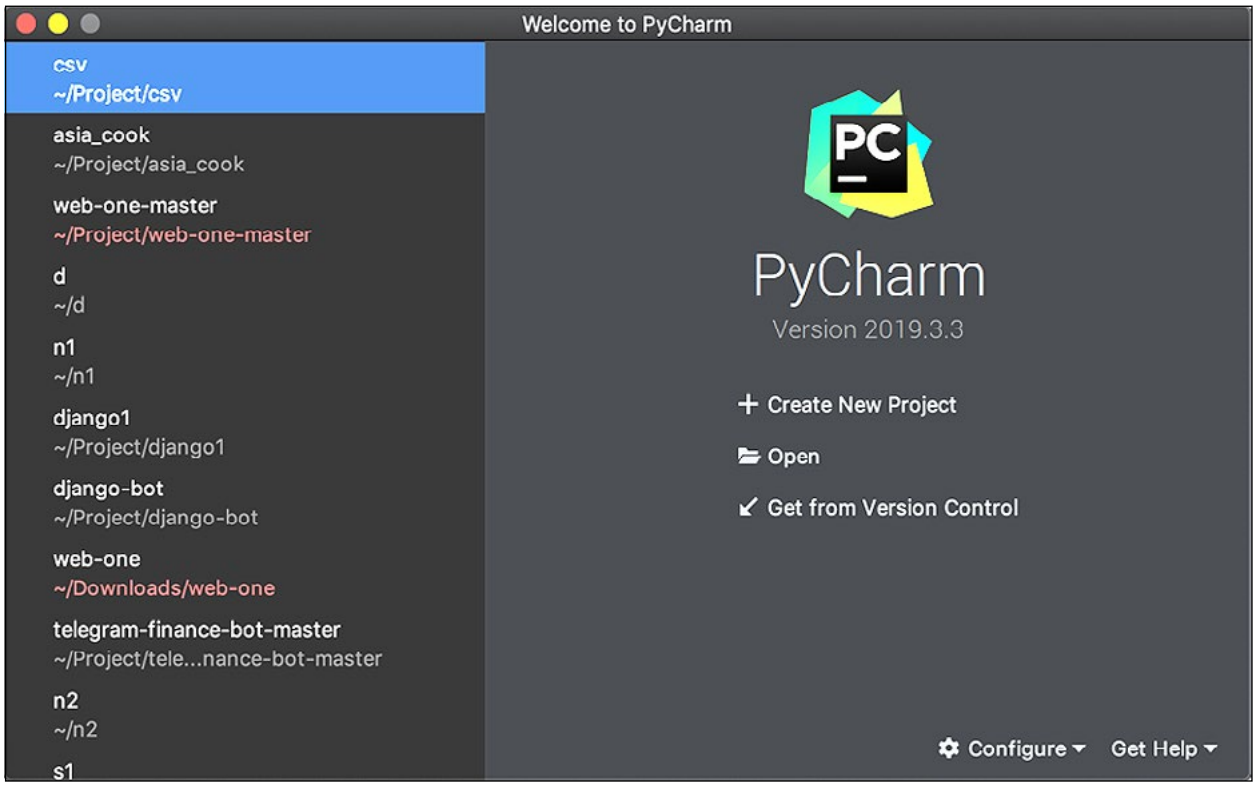


Рис. 15. Запуск IDE PyCharm

Для открытия существующего проекта нужно выбрать пункт *Открыть* (*Open*) или команду *File — Open* (рис. 16). Затем указать каталог на жёстком диске, где хранятся источники для вашего проекта.

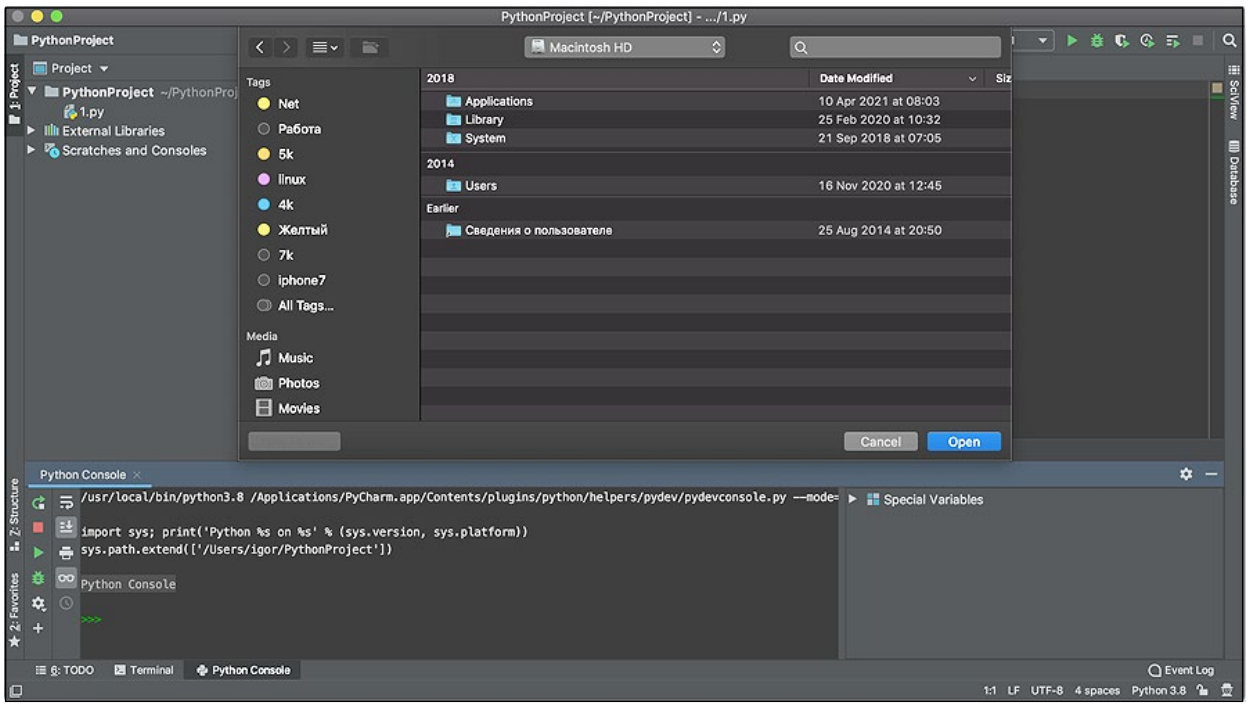


Рис. 16. Создание проекта в PyCharm

Можно выбрать новый проект (*New Project*) и ввести имя проекта в диалоговом окне (рис. 17).

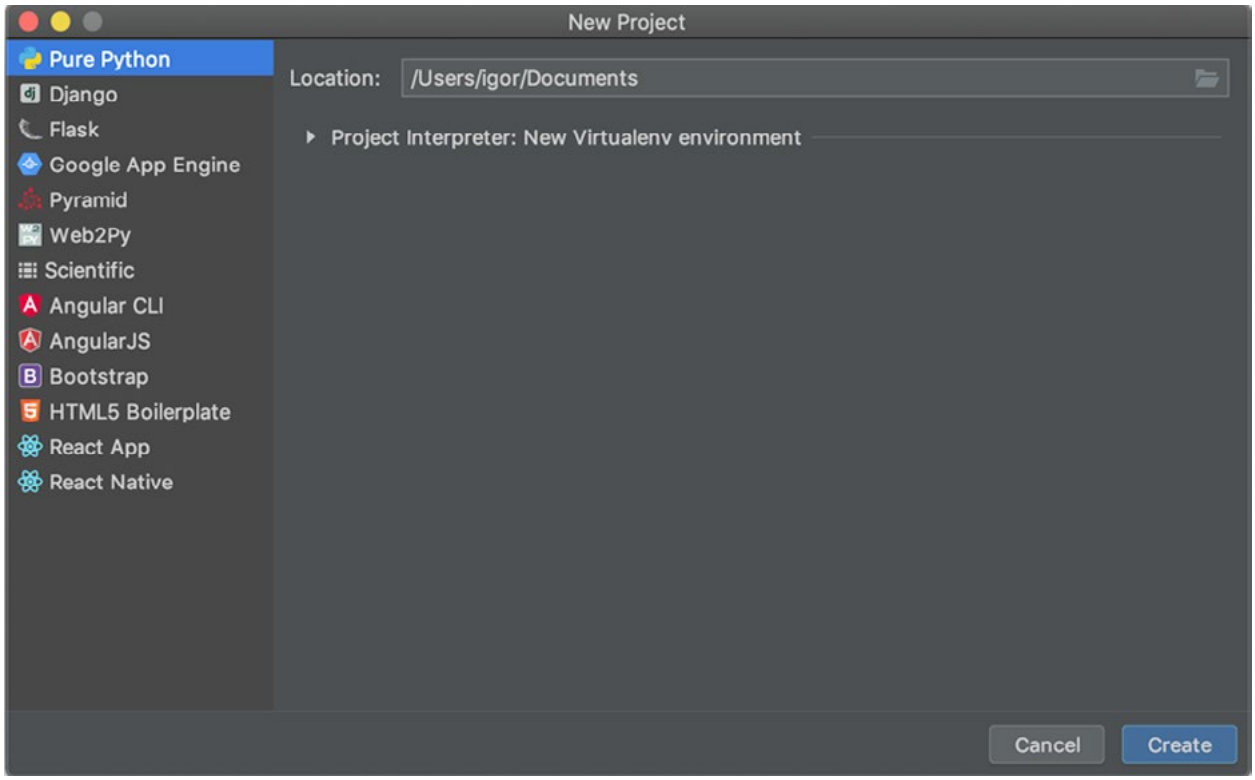


Рис. 17. Создание нового проекта в PyCharm

PyCharm предложит выбрать интерпретатор языка — нужно указать текущую установленную версию Python (рис. 18).

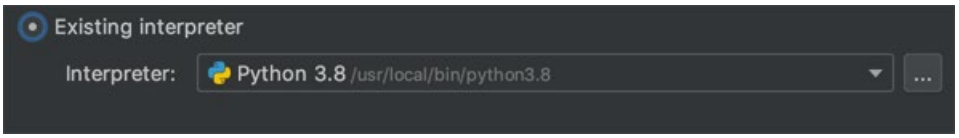


Рис. 18. Выбор интерпретатора языка Python

Рассмотрим интерфейс PyCharm. Главное окно программы разделено на несколько областей (рис. 19).

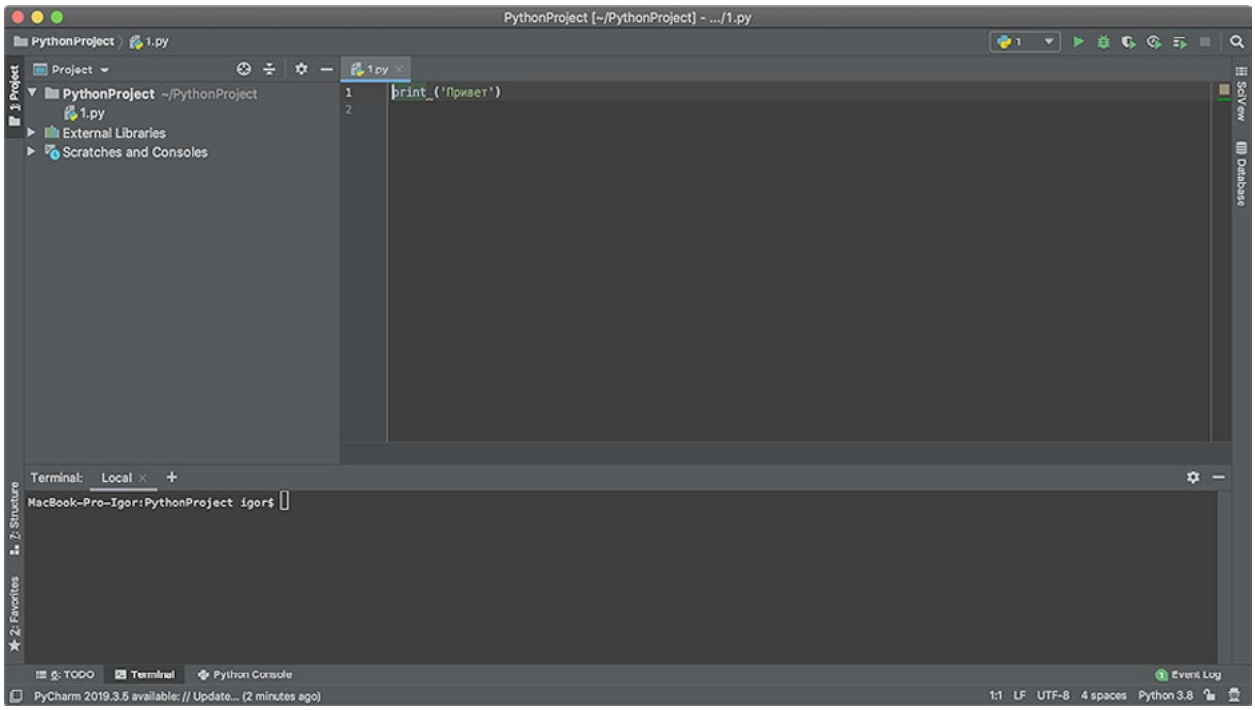


Рис. 19. Главное окно программы PyCharm

Рассмотрим основные элементы пользовательского интерфейса.

1. Панель инструментов проекта (Project Tool Window) — слева. На этой панели отображаются файлы проекта.

2. Редактор PyCharm (PyCharm Editor) — справа. Здесь вводится код программы на Python.

3. Панель навигации (Navigation Bar) — над редактором. Для быстрого запуска и отладки программы.

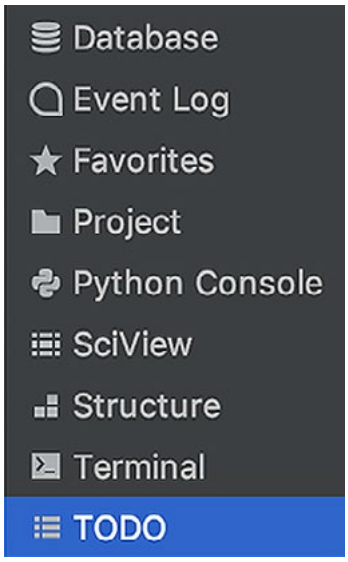


Рис. 20. Список доступных панелей инструментов в PyCharm

4. Левый столбец (Left gutter) — вертикальная полоса слева от редактора. Номера строк, переход по иерархии кода.

5. Правый столбец (Right gutter) — вертикальная полоса справа от редактора. Результаты проверки: ошибки, предупреждения и т. д.

6. Панели инструментов PyCharm (PyCharm Tool Window) — внизу. Доступ к управлению проектами, поиск и навигация по исходному коду и т. д.

7. Строка состояния (Status Bar) —горизонтальная полоса в самом низу. Состояние проекта, предупреждения, информационные сообщения. Слева в строке состояния — кнопка, управляющая показом панелей инструментов. Если навести указатель мыши на эту кнопку, появится список доступных панелей (рис. 20).

Одним из преимуществ PyCharm является гибкая настройка среды IDE, для того чтобы она полностью соответствовала вашим потребностям и была удобна для вас. Меню *File — Settings* либо *Preferences* позволяет просмотреть список доступных параметров настройки (рис. 21).

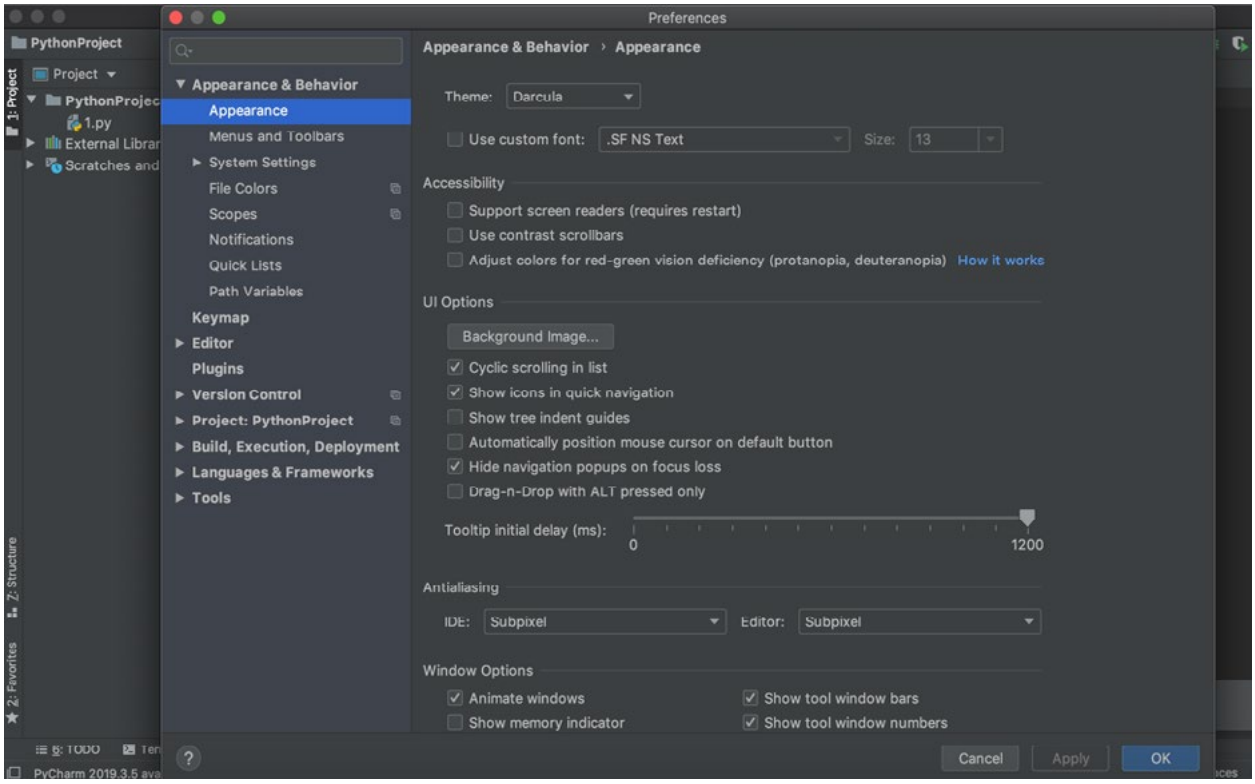


Рис. 21. Окно основных настроек PyCharm

Если в настройках выбрать пункт *Appearance and Behavior — Appearance*, то мы сможем выбрать светлую либо тёмную основную тему IDE. С помощью меню *File — Settings — Editor* можно настроить редактор (рис. 22). Доступно много опций (рис. 23).

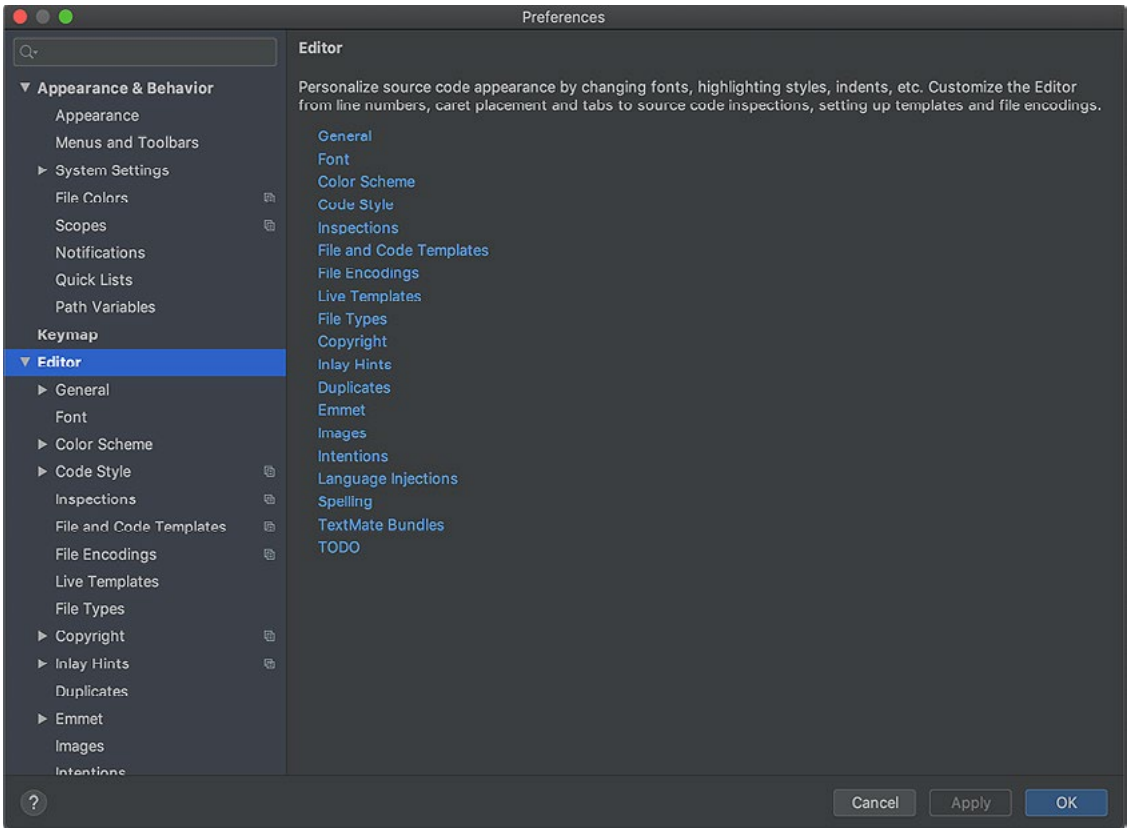


Рис. 22. Настройки редактора PyCharm

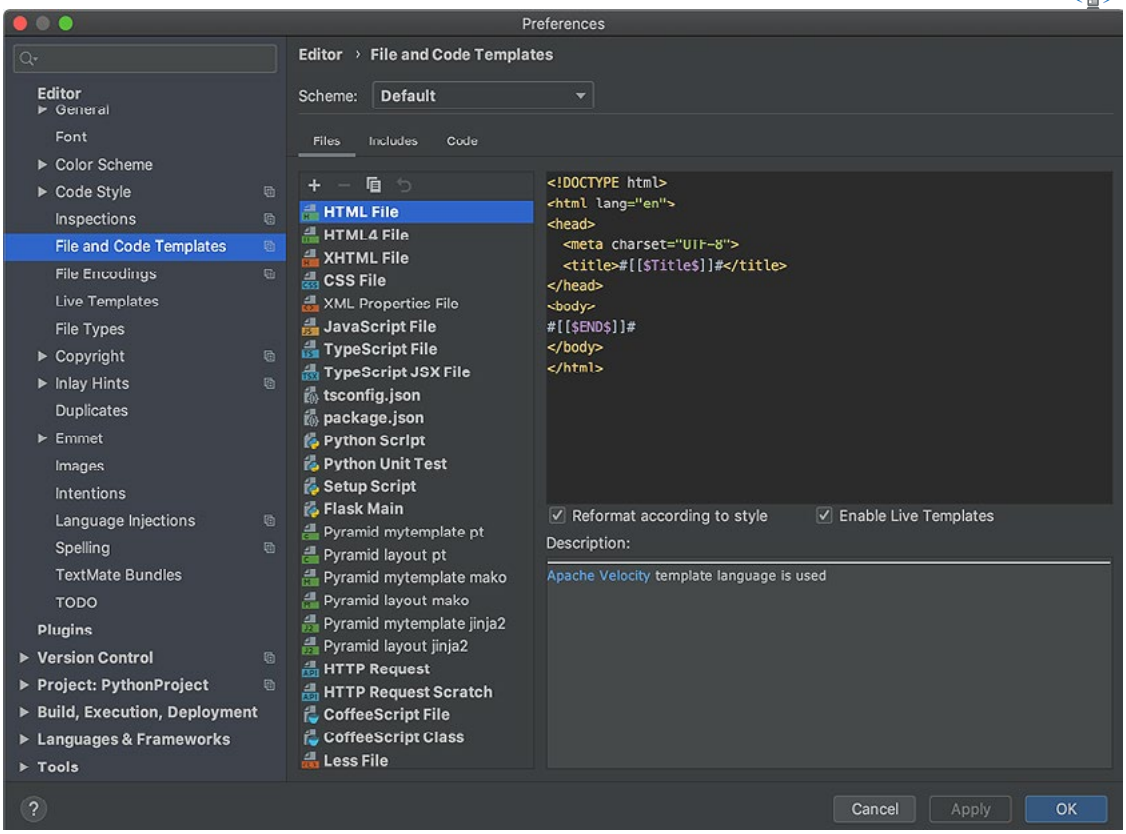


Рис. 23. Настройка шаблона файла HTML в PyCharm

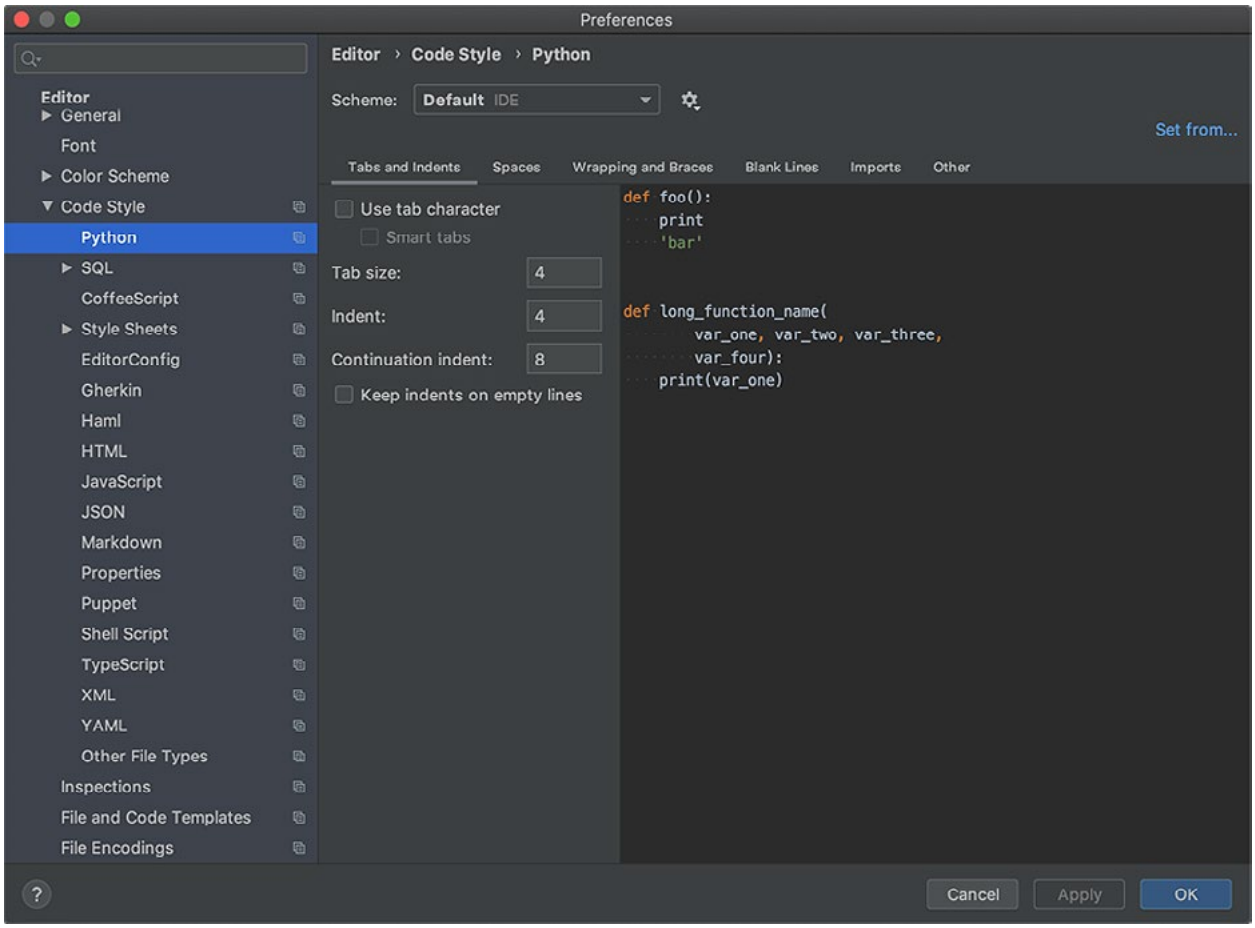


Рис. 24. Настройка стиля кода Python в PyCharm

В качестве одной из удобных функций стоит отметить настройку стиля кода языка. Для этого служит команда *File — Settings — Editor — Code Style*. Стиль кода может быть определён для каждого языка. Вы также можете создать и сохранить свой собственный стиль кода (рис. 24).

IDE PyCharm позволяет использовать «горячие клавиши». Стандартная раскладка клавиатуры выбирается командой *Help — Keymap Reference*. Раскладку всегда можно изменить с помощью команды *File — Settings — Keymap* (рис. 25).

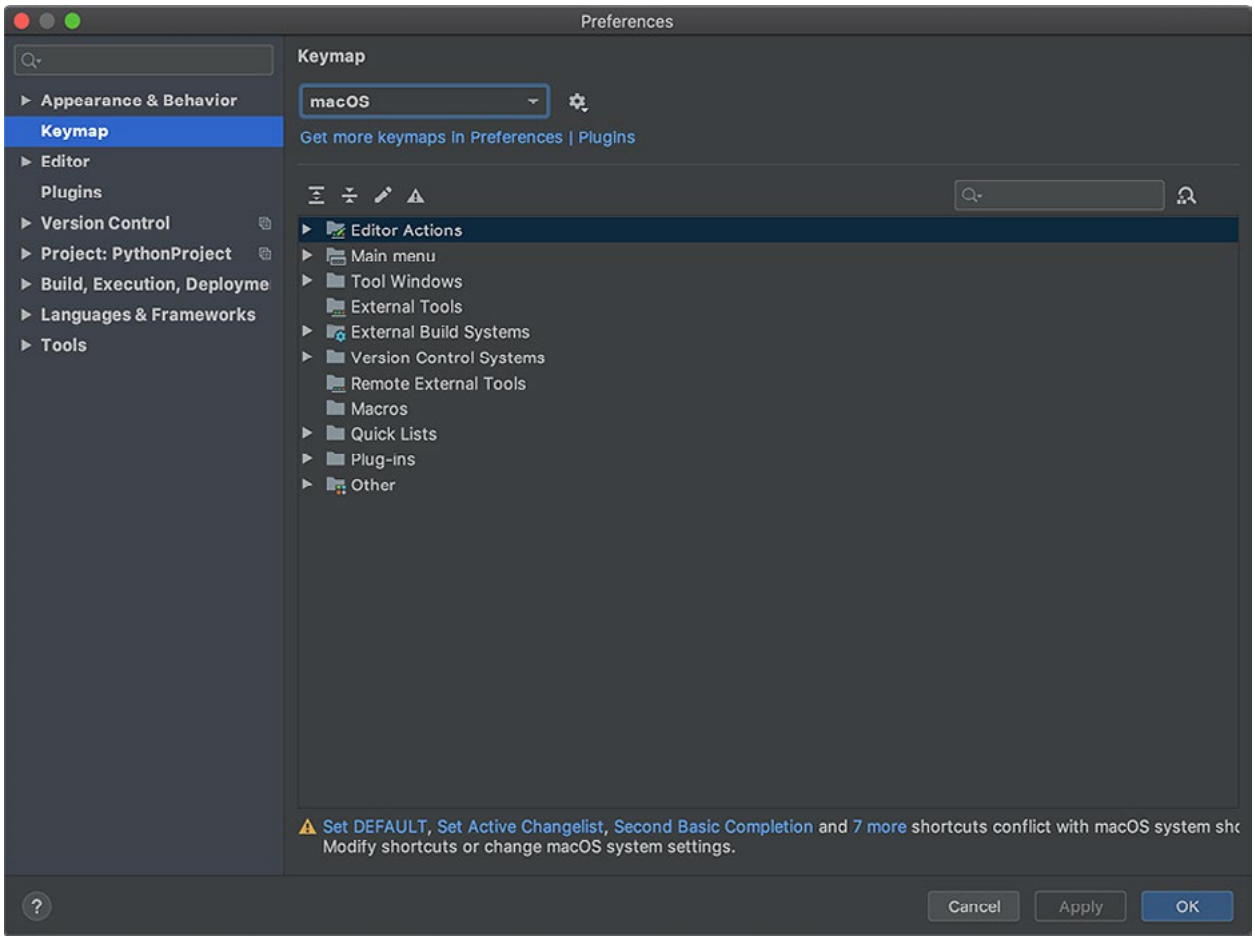


Рис. 25. Настройка раскладки клавиатуры в PyCharm

Следующая функция, которая значительно упрощает написание кода в PyCharm, — это автозаполнение кода (Auto-Completing Code). Автозаполнение помогает значительно экономить время (рис. 26).

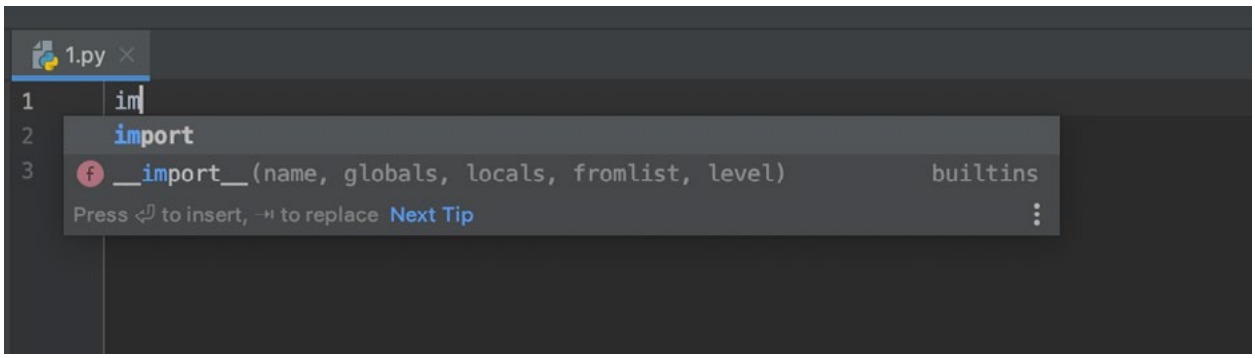


Рис. 26. Функция автозаполнения кода в PyCharm

Помимо функции автозаполнения в PyCharm доступны «умные» всплывающие подсказки и предложения (Intention Actions). Настройка подсказок доступна с помощью команды *File — Settings — Editor — Intentions* (рис. 27).

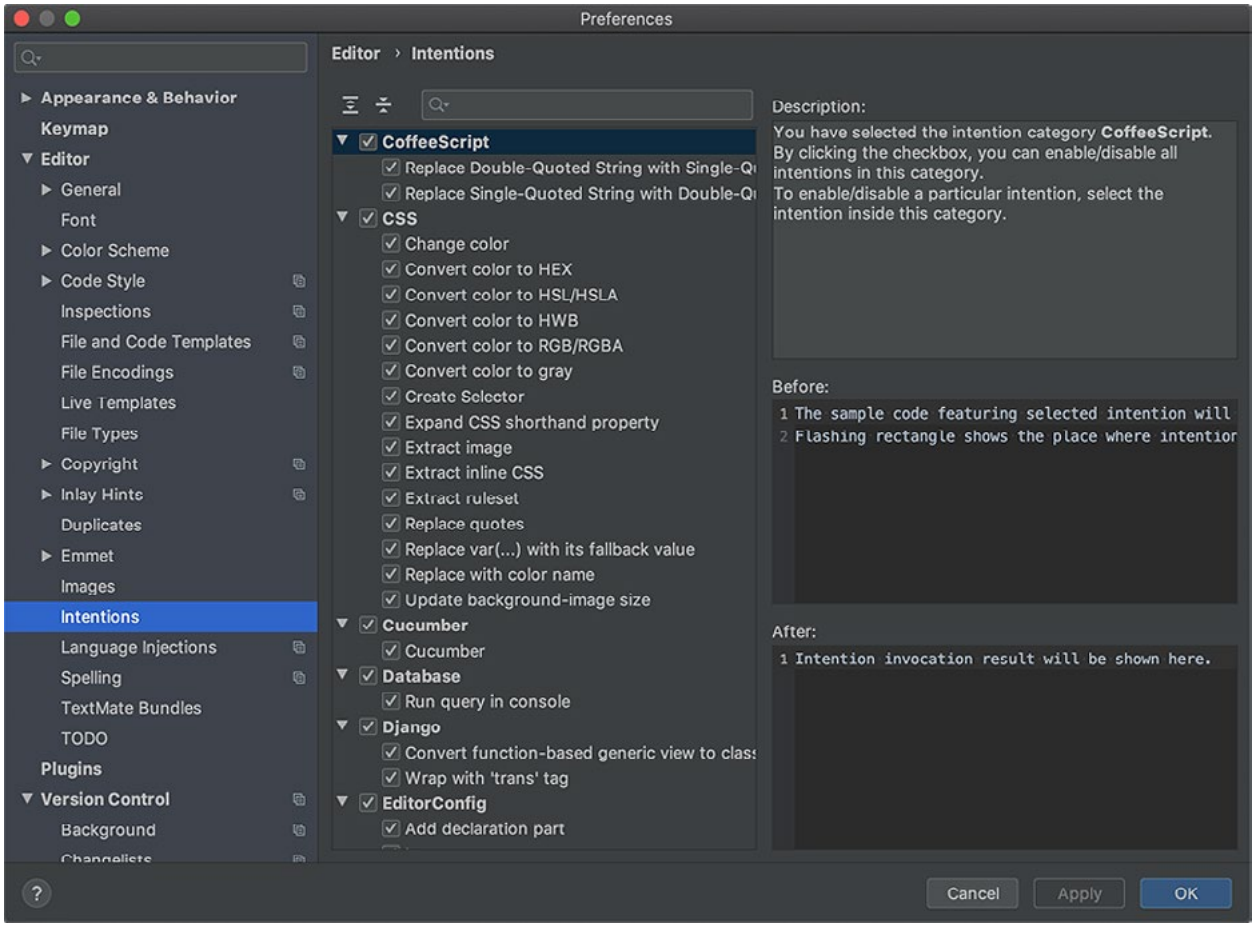


Рис. 27. Настройка подсказок в PyCharm

Программа PyCharm в реальном времени контролирует код пользователя. Она обнаруживает потенциальные ошибки и предлагает исправления. Когда среда IDE обнаруживает, например, бесконечный цикл, появляется оранжевая лампочка. Исправление можно сделать, нажав комбинацию клавиш *Alt + Enter*. Настроить проверку можно с помощью команды *File — Settings — Editor — Inspections*.

Написание кода в PyCharm может быть намного проще и быстрее с помощью генерации кода. Команда *The Code — Generate* (или комбинация клавиш *Alt + Insert*) поможет вам создавать символы и переопределять функции.

Виртуальная среда в Python

Главная задача виртуальной среды в Python состоит в создании *изолированной среды* для каждого проекта Python. Это позволит каждому проекту иметь свои собственные зависимости. Данные виртуальные среды очень легко создать при помощи инструментов `virtualenv` или `pyenv` .

При разработке программ на Python в IDE PyCharm можно использовать `virtualenv` для создания изолированного окружения. Благодаря этому пользователь сможет работать с разными версиями пакетов для разных проектов, что гораздо более практично, чем установка версий пакетов непосредственно в систему. Другим важным достоинством

`virtualenv` является то, что для установки Python-пакетов пользователь не обязан иметь права администратора.

Для установки `virtualenv` введём в командной строке (Linux или macOS) команду `pip install virtualenv`. После установки нужно создать изолированное окружение с помощью команды `virtualenv eproject`. Создаётся каталог *eproject/* для Python-окружения. Любая Python-библиотека, установленная при активированном окружении, будет сохраняться в этом каталоге.

Для активации виртуального окружения необходимо выполнить команду `source eproject /bin/activate`.

Для того чтобы в любой момент деактивировать виртуальное окружение, выполните команду `deactivate`.

При создании нового проекта в IDE PyCharm можно создать виртуальное окружение для вашего проекта (рис. 28).

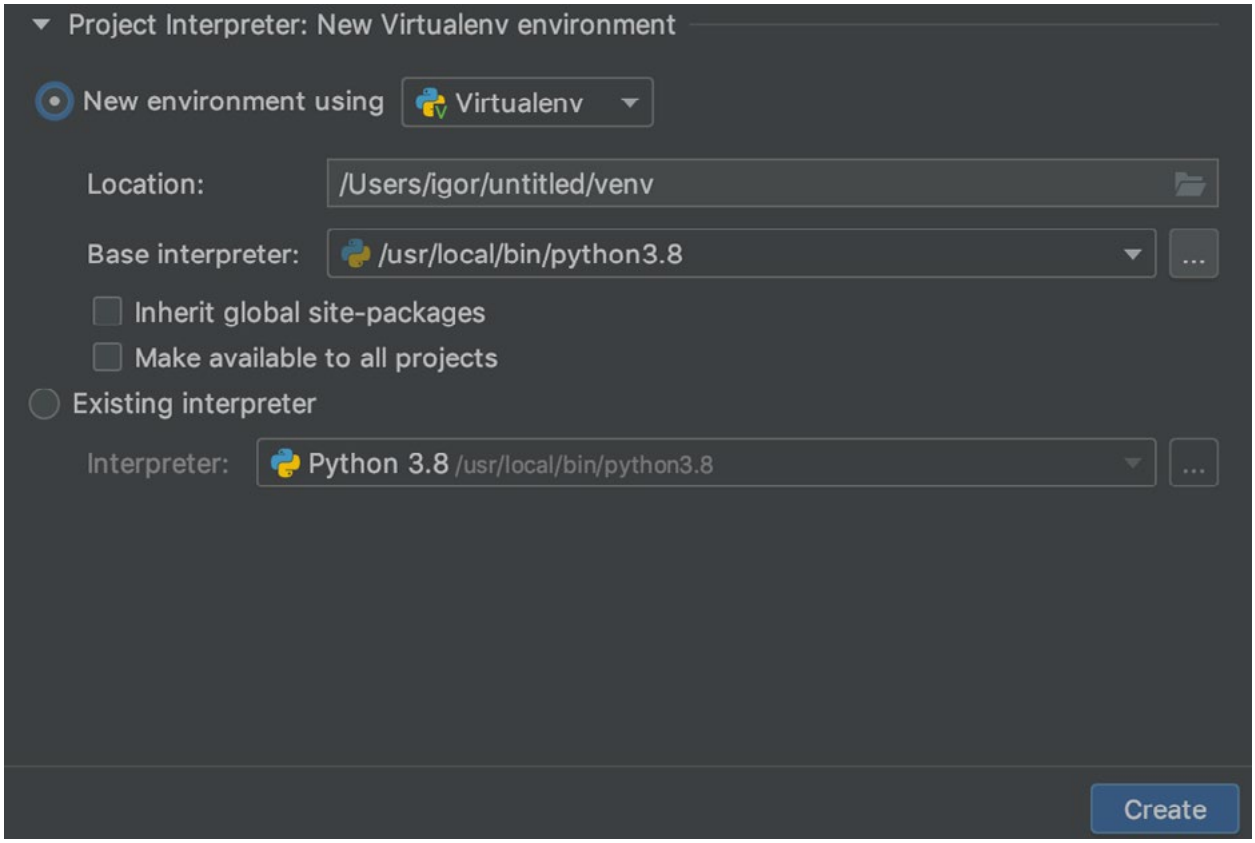


Рис. 28. Выбор новой виртуальной среды проекта в PyCharm

Лабораторные работы

Лабораторная работа 1. Знакомство со средой программирования Python. Переменные

Теоретическая часть

При объяснении учащимся теоретического материала учитель может опираться на информацию из справочных материалов, представленных выше.

Практическая часть

Цель работы: ознакомление со средой программирования Python и запуском первых программ.

Ход работы

1. Проверить, установлена ли на компьютере/ноутбуке среда IDE PyCharm.

2. Ознакомиться с помощью учителя со справочным материалом, посвящённым установке языка Python, установке и первичной настройке IDE PyCharm.

3. Создать первую программу на языке Python по приведённому ниже описанию.

Хорошей традицией при изучении языка программирования является написание первой вашей программы с простым приветствием на экране «Привет, я изучаю Python!».

На жёстком диске создать каталог (папку) с именем *PythonProject*. Запустить среду разработки PyCharm. Выбрать пункт *Открыть* (*Open*) и указать вашу папку *PythonProject*. Откроется рабочее окно вашего проекта (рис. 29).

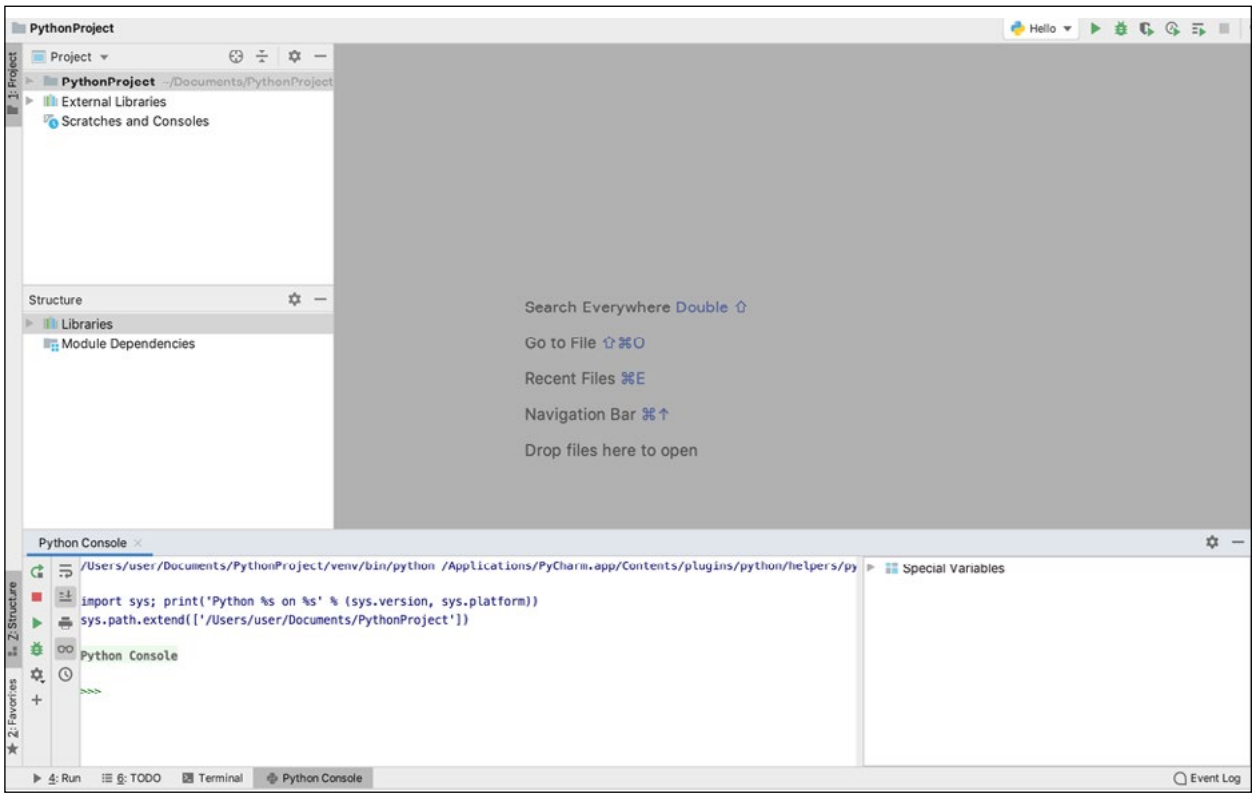


Рис. 29. Окно проекта

Выбрать команду *Файл — Новый — Файл Python* (*File — New — Python File*) (рис. 30).

Задать имя для созданного файла: *Hello* (рис. 31). Заметим, что расширение файла *.py* добавилось автоматически.

В правой области окна проекта откроется область с этим файлом, в которой вы будете набирать код вашей программы. Наберите первую команду: `print` (вывод на экран). Как только в редакторе PyCharm начинается набор, автоматически предлагается автодополнение команды (рис. 32).

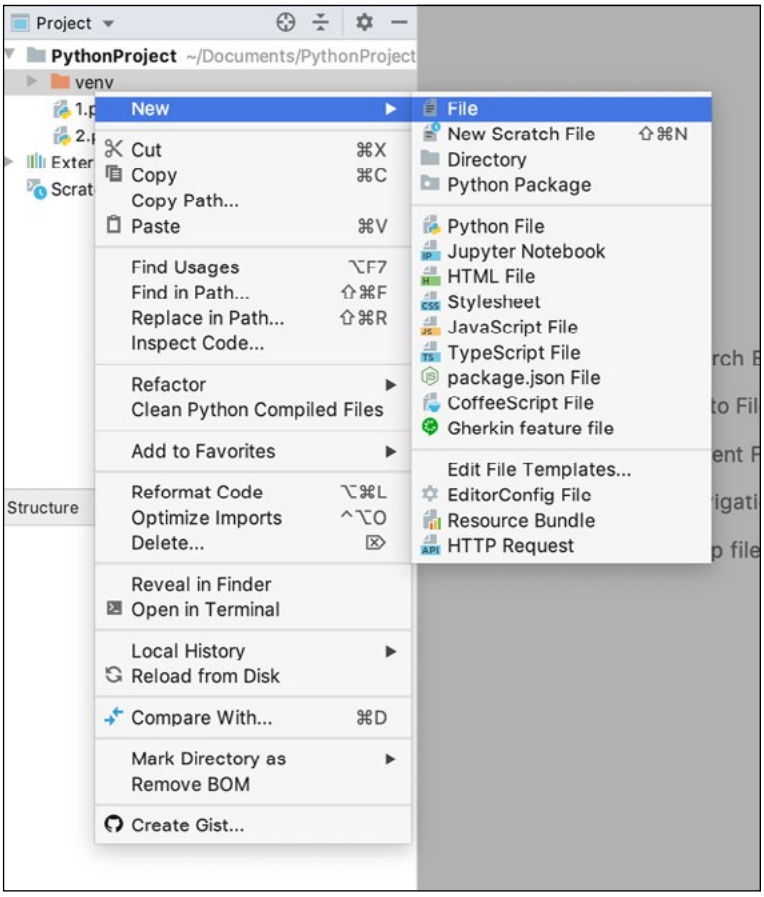


Рис. 30. Создание файла Python

Рис. 31. Ввод имени файла программы

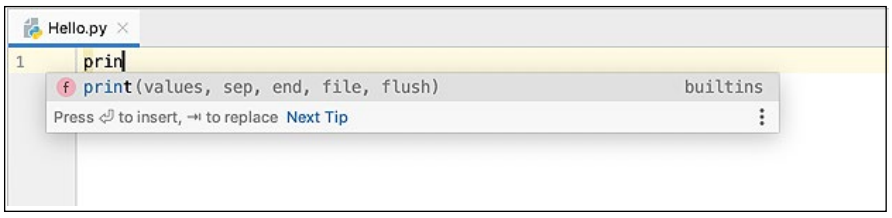


Рис. 32. Пример автодополнения

В скобках в качестве параметра команды `print` в кавычках указать текст, который необходимо вывести в результате работы нашей программы (рис. 33).

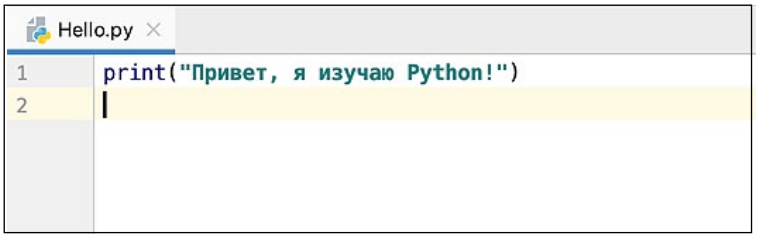


Рис. 33. Ввод команды `print`

Набор кода программы выполнен. Теперь необходимо запустить программу и получить результат. Для запуска кода Python в окне *Конфигурации Запуска/Отладки* (*Run/Debug Configuration*) указать наш интерпретатор: Python нужной нам версии и *script path*: это имя требуемого файла *1.py* (рис. 34, 35). Применить настройки (кнопка *ОК*) и закрыть окно конфигурации.

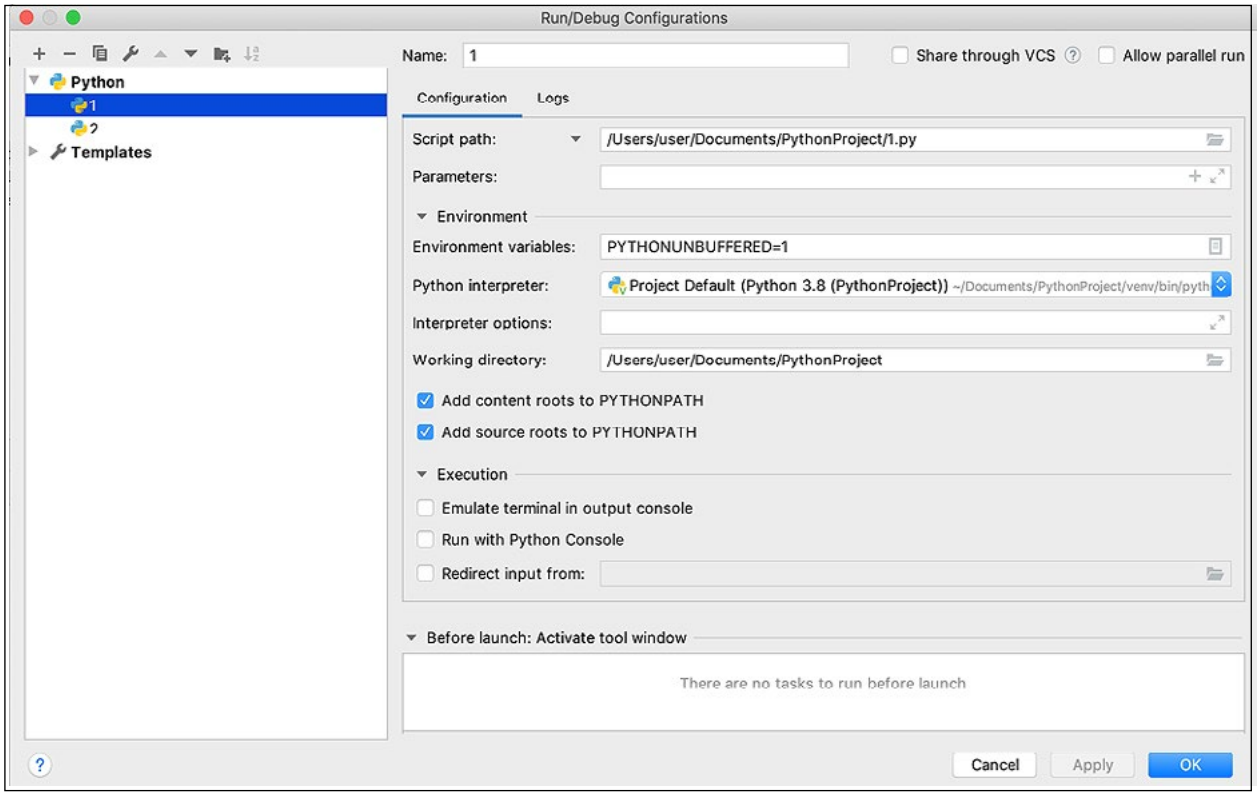


Рис. 34. Окно конфигурации

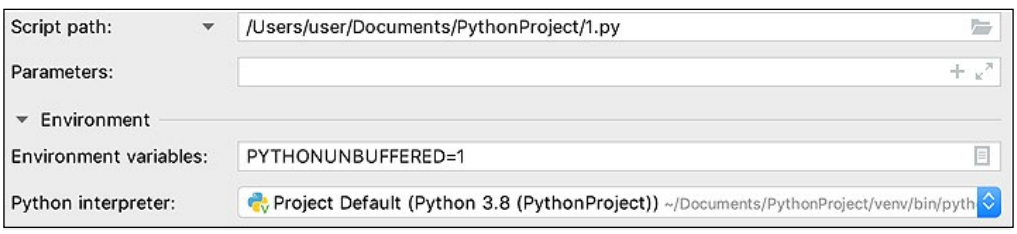


Рис. 35. Окно конфигурации

В правом верхнем углу программы PyCharm расположена панель запуска (рис. 36). Для выполнения кода программы нужно нажать кнопку запуска (*зелёный треугольник*).

Рис. 36. Панель запуска

Результат работы программы появится мгновенно в окне вывода результатов и будет иметь следующий вид (рис. 37).

Рис. 37. Результат работы программы

Далее предлагается рассмотреть следующий механизм, помогающий в написании хорошего кода, — это комментарии. Комментарии пишутся после символа # («решётка») и являются заметками для читающего программу (рис. 38). Так легче понять, что программа делает.

Пример 1

Набрать следующий код:

```
print("Привет, я изучаю Python!") # print -- это функция вывода
```

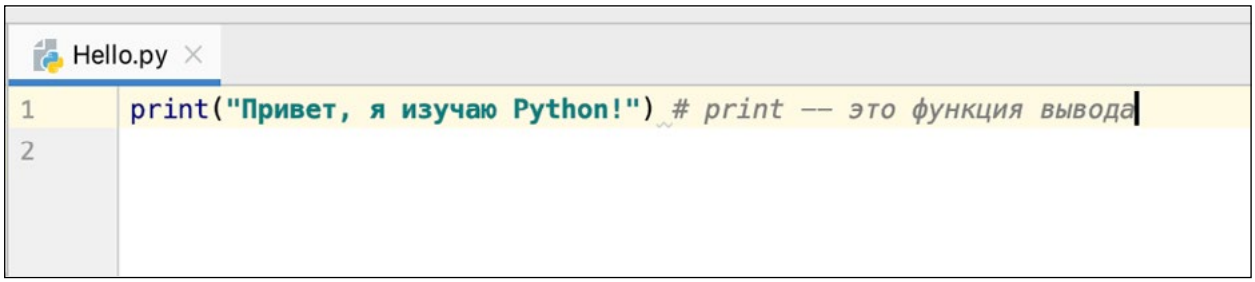


Рис. 38. Вид комментария в программе

Пример 2

Рассмотрим случай, когда в строку вывода требуется добавить какие-либо данные. Для этого в Python применяют метод format(). Набрать следующий код и запустить программу (рис. 39, 40).

```
age=5
print("Привет, я изучаю Python {0} лет!" format(age))
```

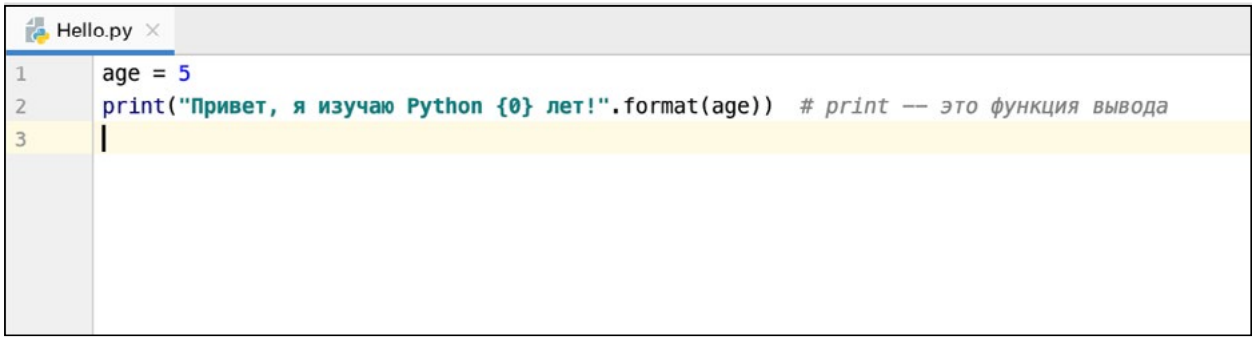


Рис. 39. Вид программы в среде разработки

Рис. 40. Результат работы программы

Метод `format` замещает фигурные скобки и их содержимое соответствующими аргументами. Обозначение `{0}` соответствует переменной `age`, которая является первым аргументом метода `format`. Будет выведено значение переменной `age`. Python начинает отсчёт с нуля, поэтому первому аргументу соответствует 0. (Второму аргументу будет соответствовать 1 и т. д.)

Выводы

В ходе выполнения лабораторной работы вы познакомились со средой программирования Python и запуском первых программ.

Контрольные вопросы

1. С какой средой программирования вы познакомились в ходе выполнения лабораторной работы?
2. Какие действия вы выполняли в ходе лабораторной работы?
3. Какие действия выполняла первая программа, написанная на языке программирования Python?

Лабораторная работа 2.1. Первые программы на языке Python, основные операторы

Теоретическая часть

В первой лабораторной работе были показаны основные аспекты работы со средой для языка Python, вы писали простые команды (операторы).

Далее предлагается рассмотреть структуру программы на языке Python, а также основные правила написания программ.

Любую программу на языке Python можно представить как набор лексем (допустимых символов), записанных в определённом порядке и по определённым правилам. Лексема может представлять собой: литералы, знаки пунктуации, переменные, специальные ключевые слова, комментарии.

Программа на языке Python может содержать достаточное количество комментариев, каждый комментарий начинается с символа `#` «решётка».

Литералы представляют собой значения, заданные в коде программы, например числа (`25`) или строки (`"Привет"`). В языке Python используется динамическая типизация (типы данных определяются автоматически, и их не требуется объявлять в программном коде), но при этом Python является языком со строгой типизацией (вы сможете выполнять над объектом только те операции, которые применимы к его типу).

Если говорить об использовании знаков пунктуации, то стоит отметить, что каждая строка в программе на языке Python не должна заканчиваться точкой с запятой, как, например, в C++, но если есть необходимость записать несколько операторов в одну строчку, то их можно разделять точкой с запятой.

Переменные используются для хранения данных.

Важно!

В языке Python нет специального раздела описания переменных, в котором указывается тип переменной перед её первым использованием. Тип переменной определяется по тому значению, которое ей присваивается.

Есть определённые правила для задания имён переменных (идентификаторов): это последовательность букв и цифр, которая не может начинаться с цифры, но может содержать символ подчёркивания (_). Имена переменных чувствительны к регистру. Имена переменных не могут совпадать с ключевыми словами.

Ключевые слова в языке Python имеют специальное назначение и представляют собой управляющие конструкции языка. Примеры: `and`, `break`, `for` и т. д.

При составлении программ лексемы объединяются в синтаксические конструкции, которые могут вкладываться друг в друга.

Важно!

В результате могут образовываться блочные конструкции, каждый блок кода начинается двоеточием (:), а тело блока выделяется обязательным отступом.

Обычно среда программирования сразу делает отступ для блока после двоеточия. В зависимости от используемой среды программирования блоки могут иметь визуальное выделение.

Важно!

С рекомендациями по составлению программ на языке можно ознакомиться, например, по ссылке:
https://pythonworld.ru/osnovy/sintaksis-yazyka-python.html

Перейдём к рассмотрению основных операторов языка. Оператор — это конструкция языка, определяющая команду (набор команд) языка программирования, задающая выполнение действий.

Представленные в таблице 7 операторы можно разделить на следующие типы: арифметические, операторы сравнения, логические, битовые.

Таблица 7

Некоторые операторы языка Python

Оператор	Описание
+	Сложение
–	Вычитание
*	Умножение
/	Деление
//	Целочисленное деление
%	Остаток от деления
**	Возведение в степень
<	Меньше
>	Больше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Не равно
and	Логическое умножение («и», конъюнкция)
or	Логическое сложение («или», дизъюнкция)
not	Логическое отрицание
|	Побитовое «или»
^	Побитовое исключающее «или»
&	Побитовое «и»
>>	Битовый сдвиг вправо
<<	Битовый сдвиг влево
~	Инверсия битов

Рассмотрим несколько примеров использования данных операторов.

Пример 1

На рисунке 41 показан результат выполнения арифметических операторов с консоли Python. Операторы вводятся в командной строке после приглашения >>> и сразу выполняются интерпретатором после нажатия клавиши ввода.

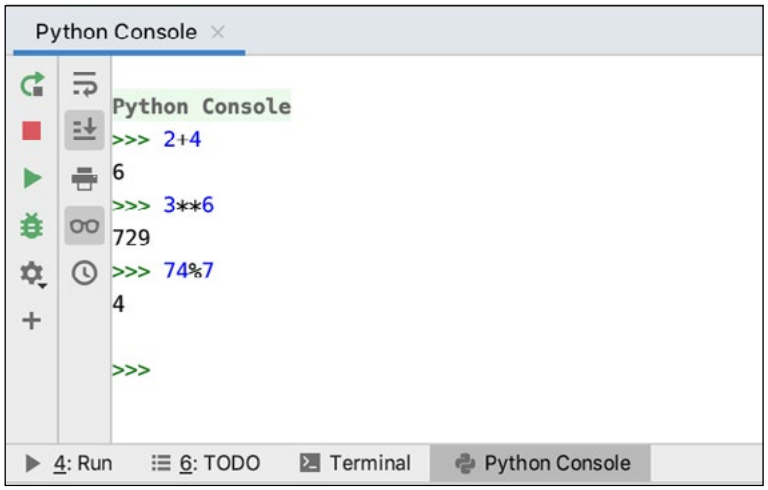


Рис. 41. Результат выполнения арифметических операторов

Один из самых часто используемых операторов — оператор присваивания. Оператор присваивания в языке Python имеет вид:

```
<идентификатор>=<выражение>
```

Важно!

Значения выражения, которое стоит справа от знака «=», присваивается переменной (записывается в эту переменную — в память), расположенной слева от знака «=».

Пример 2

```
a=7
n77=21.9
```

Также в языке Python есть сложные операторы присваивания, они представлены в таблице 8.

Таблица 8

Сложные операторы присваивания

Обозначение	Описание
+=	Значения левой и правой частей оператора присваивания суммируются, результат присваивается переменной в левой части
−=	Вычитается значение правой части из значения переменной в левой части, результат присваивается переменной в левой части

Окончание

Обозначение	Описание
`*=`	Значения левой и правой частей перемножаются, результат присваивается переменной в левой части
`/=`	Значение переменной из левой части делится на значение выражения в правой части, результат присваивается переменной в левой части
`%=`	Находится остаток от деления значения переменной в левой части на значение выражения в правой части, результат присваивается переменной в левой части
`**=`	Выполняется возведение значения переменной в левой части в степень значения выражения в правой части, результат присваивается переменной в левой части
`//=`	Делится нацело значение переменной в левой части на значение выражения в правой части, результат присваивается переменной в левой части

Пример 3

На рисунке 42 показан результат выполнения операторов присваивания с консоли Python.

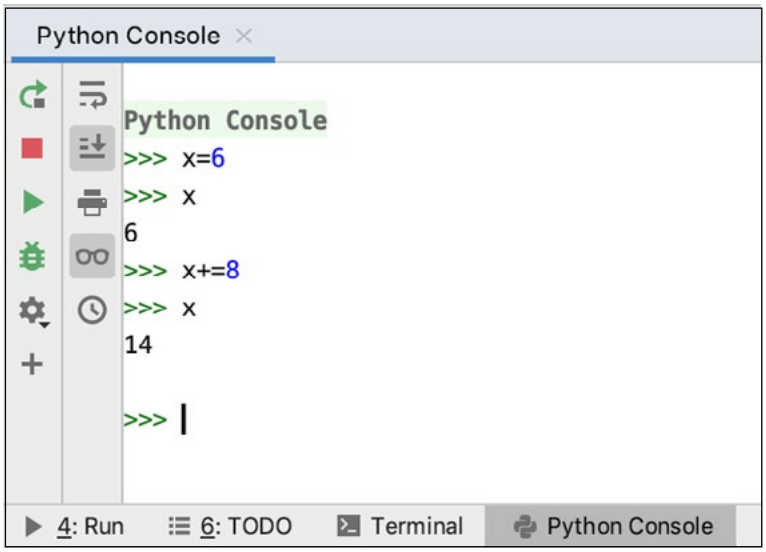


Рис. 42. Результат выполнения операторов присваивания

Часто при решении задач необходимо выполнять ввод данных с клавиатуры и выводить данные на экран. Рассмотрим работу соответствующих операторов в языке Python.

Ввод данных с клавиатуры осуществляется с помощью оператора `print`. Формат оператора:

```
print()
```

Пример 4

```
x=3
y=6
z=-10
print(x,y,z)
```

Результат работы программы представлен на рисунке 43.

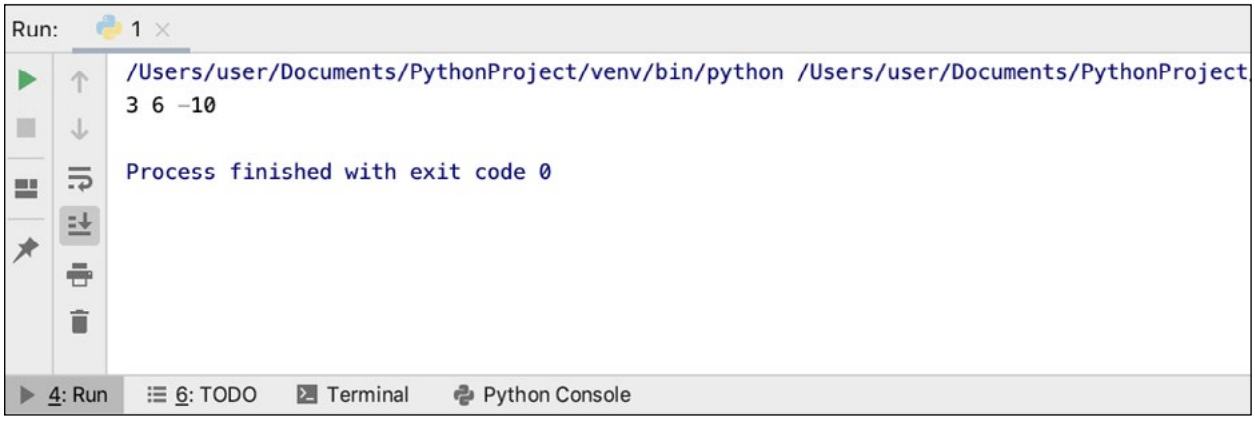


Рис. 43. Результат работы программы

Важно!

После выполнения оператора `print` курсор переходит на новую строку.

Пример 5

```
x=3
y=6
z=-10
print(x)
print(y,z)
```

Результат работы программы представлены на рисунке 44.

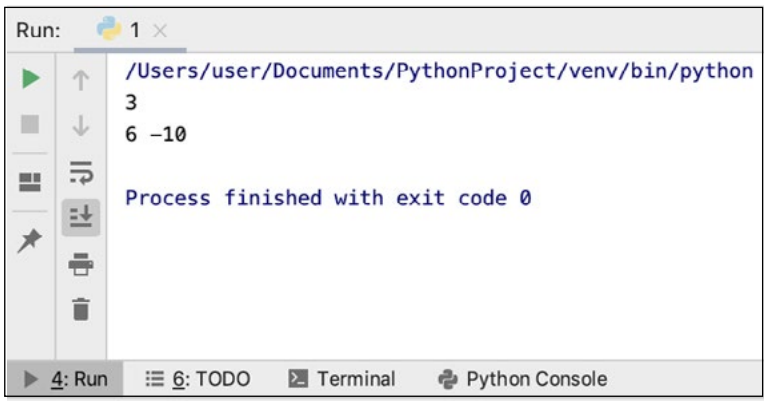


Рис. 44. Результат работы программы

Для вывода в строку необходимо добавить аргумент `end="<разделитель>"`.

Пример 6

```
x=3
y=6
z=-10
print(x)
x+=6
print("x=",x,end="")
print(y,z)
```

Результаты работы программы представлены ниже на рисунке 45.

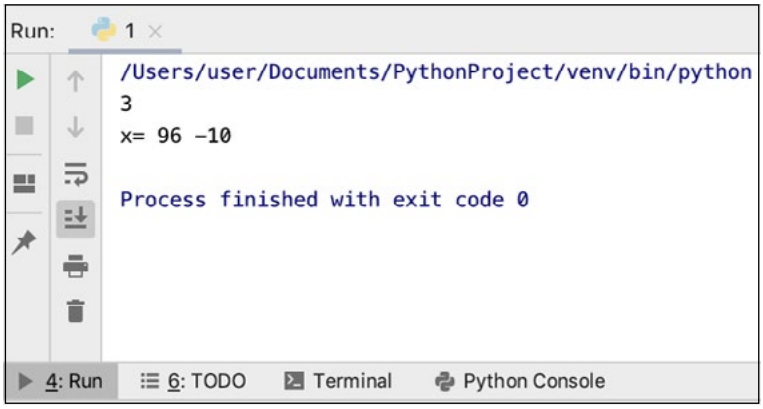


Рис. 45. Результат работы программы

Выводить на экран можно не только значения переменных, но и значения выражений.

Пример 7

```
x=3
y=6
z=-10
print(x)
x+=6
print("x=",x,end="")
print(y+z)
```

Результат работы программы представлен на рисунке 46.

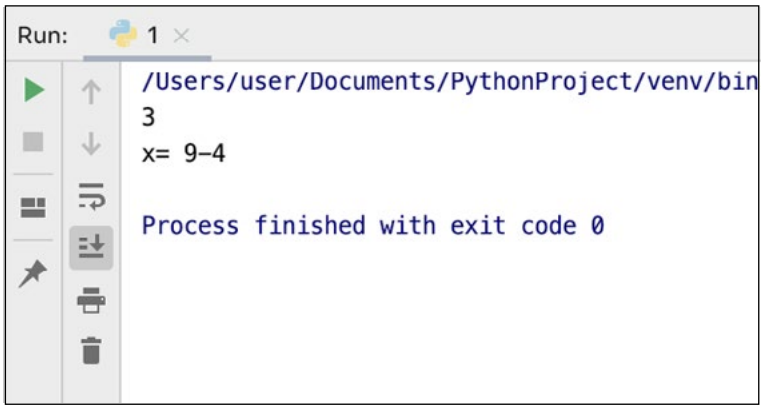


Рис. 46. Результат работы программы

Также в языке Python можно использовать форматированный вывод. Для этого необходимо использовать встроенную функцию format(). Синтаксис функции:

```
<строка>.format(<формат>)
```

Здесь:

`<строка>` представляет собой значение для форматированного вывода;

`<формат>` — спецификации формата Mini-Language.

Пример 8

```
a=12
b=13
c=a+b
print("{}+{}={}".format(a, b, c))
```

Результат работы программы представлен на рисунке 47.

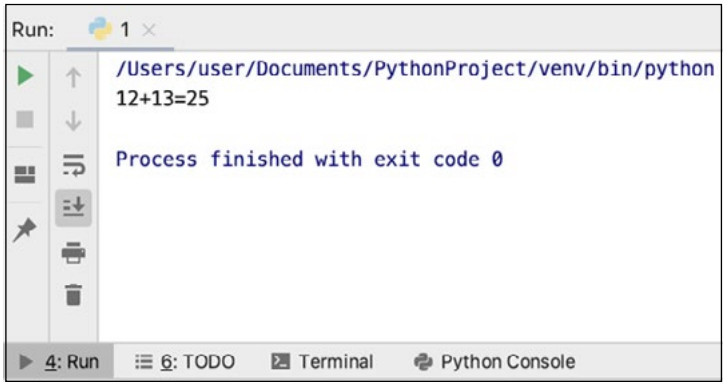


Рис. 47. Результат работы программы

Для ввода данных с клавиатуры используется встроенная функция `input()`.

Важно!

Функция `input()` возвращает в качестве результата строку!

Пример 9

В результате выполнения программы

```
a=input()
c=input()
z=a+c
print(z)
```

на экран выводится не сумма чисел. Выводится строка — сцепление двух строк `a` и `c`.

Результат работы

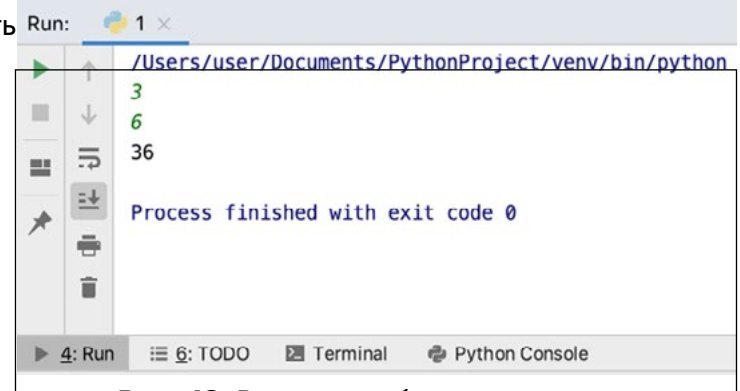


Рис. 48. Результат работы программы

Если необходимо ввести числовые значения, то следует использовать встроенные функции `int` и `float`.

Функция `int` используется для преобразования строки в целое число.

Пример 10

Изменим предыдущий пример:

```
a=int(input())
c=int(input())
z=a+c
print(z)
```

Результат работы — сумма чисел.

Результат работы программы представлен на рисунке 49.

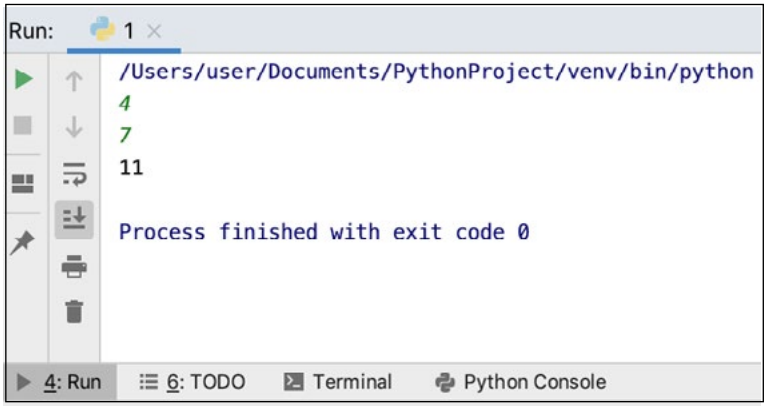


Рис. 49. Результат работы программы

Также можно указывать аргумент функции `input`, в этом случае на экран будет выводиться «пригласительное сообщение».

Пример 11

```
a=int(input("a= "))
c=int(input("c= "))
z=a+c
print(z)
```

Результат работы программы представлен на рисунке 50.

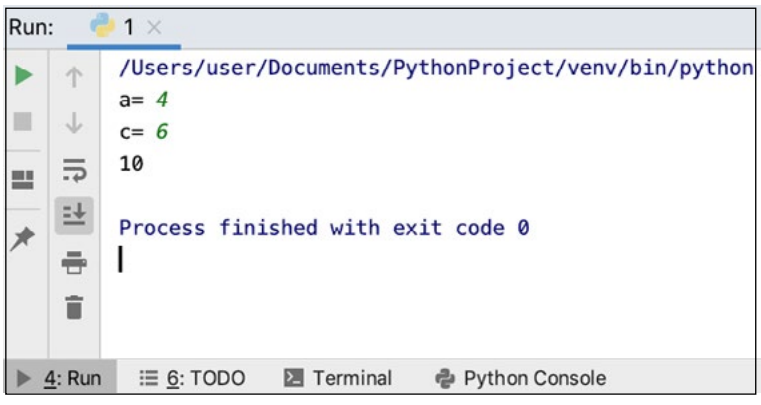


Рис. 50. Результат работы программы

Аналогично работает и функция `float`, но в результате её работы введённое значение преобразовывается в вещественное число.

Пример 12

```
a=float(input("a= "))
x=float(input("x= "))
z=a+x
print(z)
```

Результат работы программы представлен на рисунке 51.

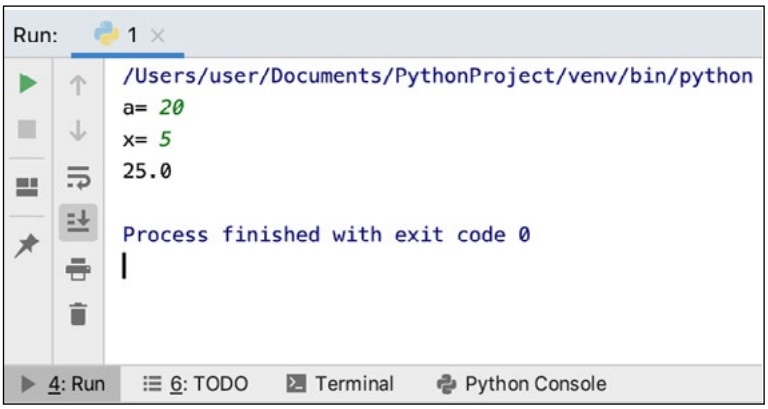


Рис. 51. Результат работы программы

Упомянем работу со встроенными математическими функциями.

Важно!

Все математические функции в языке Python объединены в модуль `math`.

Данный модуль при необходимости нужно импортировать с помощью команды `import`:

```
import math
```

После подключения этого модуля можно использовать всё множество математических функций языка Python.

В составе модуля `math` есть несколько известных математических констант (табл. 9).

Таблица 9

Математические константы языка Python

Обозначение константы	Описание
`math.pi`	Математическая константа π = 3,141592...
`math.e`	Число Эйлера *e* = 2,71828...
`math.inf`	Положительная бесконечность

Также опишем несколько наиболее часто используемых функций, входящих в состав модуля `math` (табл. 10).

Таблица 10

Математические функции языка Python

Обозначение функции	Описание
`math.exp(x)`	Экспонента числа e^x
`math.log(X,[A])`	Логарифм *X* по основанию *A*. Если *A* не указано, вычисляется натуральный логарифм
`math.pow(X,n)`	Возведение в степень X^n
`math.sqrt(X)`	Квадратный корень из *X*
`math.cos(X)`	Косинус *X* (*X* указывается в радианах)
`math.sin(X)`	Синус *X* (*X* указывается в радианах)
`math.tan(X)`	Тангенс *X* (*X* указывается в радианах)
`math.acos(X)`	Арккосинус *X* (*X* указывается в радианах)
`math.asin(X)`	Арксинус *X* (*X* указывается в радианах)
`math.atan(X)`	Арктангенс *X* (*X* указывается в радианах)
`math.degrees(X)`	Конвертирует радианы в градусы
`math.radians(X)`	Конвертирует градусы в радианы

(Квадратные скобки в синтаксисе функций означают необязательные элементы.)
Полный список математических функций можно вывести командой:

```
import math
>>> math
<module 'math' (built-in)>
>>> dir(math)
```

Результат работы программы представлен на рисунке 52.

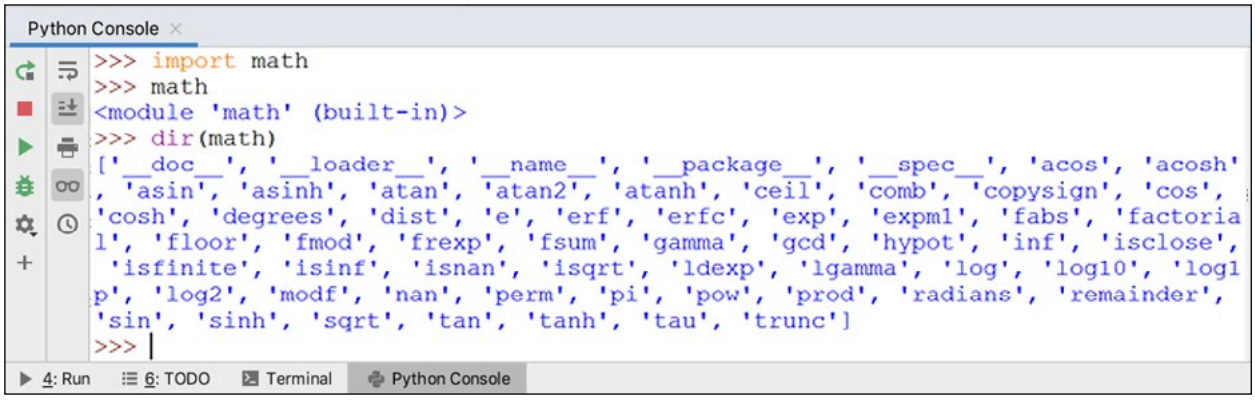


Рис. 52. Вывод списка математических функций языка Python

Важно!

Также в языке Python есть возможность получения так называемых псевдослучайных чисел. Данные числа вычисляются по некоторой математической формуле, могут использоваться несколько алгоритмов (в языке Python используется алгоритм «вихрь Мерсенна»).

Для получения таких чисел необходимо обратиться к модулю `random`.
Имеется возможность получать как целые, так и вещественные случайные числа:
`randint(a,b)` — возвращает случайное целое число из отрезка [*a*; *b*];
`uniform(a,b)` — возвращает случайное вещественное число из отрезка [*a*; *b*].

Пример 13

```
from random import randint
from random import uniform
k=randint(1,100)
z=uniform(1,200)
print(k)
print(z)
```

Результат работы программы представлен на рисунке 53.

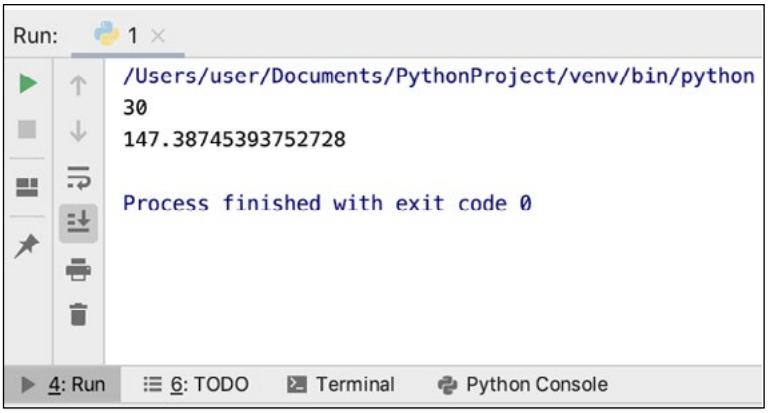


Рис. 53. Результат работы программы

Пример 14

Вычислить выражение $a^2 + b^2$, где *a*, *b* — целые числа, введённые с клавиатуры.

```
a=int(input("a= "))
b=int(input("b= "))
c=a**2+b**2
print("c=", c)
```

Результат работы программы представлен на рисунке 54.

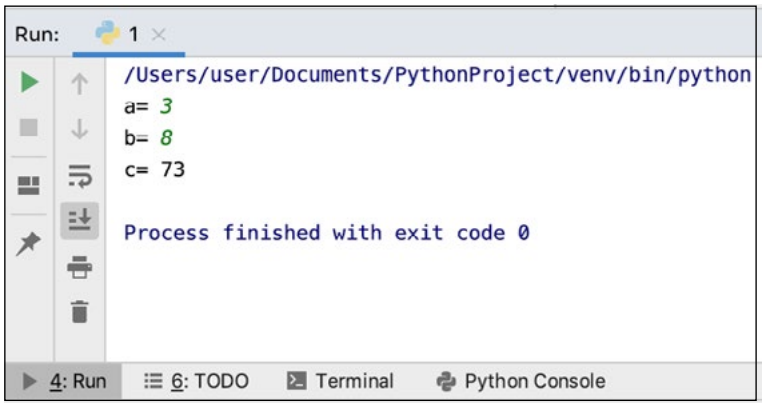


Рис. 54. Результат работы программы

Практическая часть

Цель работы: ознакомление с основами написания программ на языке программирования Python, работа с операторами присваивания, ввода/вывода данных.

Ход работы

Открыть среду разработки PyCorm.

Составить программу нахождения периметра и площади прямоугольника (значения длин сторон вводятся с клавиатуры).

Указание. Примерный вид программы:

```
a=float(input("a= "))
b=float(input("b= "))
print("периметр= ",2*(a+b))
print("площадь= ",a*b)
```

Выводы

В ходе выполнения лабораторной работы вы получили представление о написании простых программ на языке программирования Python.

Контрольные вопросы

1. Какие операторы языка Python вы использовали при выполнении лабораторной работы?
2. Какой оператор используется для ввода целых чисел с клавиатуры; вещественных чисел с клавиатуры?
3. Как вывести данные на экран в языке Python?

Лабораторная работа 2.2. Первые программы на языке Python, основные операторы

Теоретическая часть

Воспользоваться материалами из лабораторной работы 2.1.

Практическая часть

Цель работы: ознакомление с основами написания программ на языке программирования Python, работа с операторами присваивания, ввода/вывода данных.

Ход работы

1. Открыть среду разработки PyCharm.
2. Составить программу вычисления значения функции $y(x) = x^2 - 7x + 8$ для заданного с клавиатуры значения аргумента x.

Указание. Примерный вид программы:

```
x=float(input("x= "))
y=x**2-7*x+8
print("y(x)= ", y)
```

3. Составить программу вычисления расстояния между двумя точками, заданными на плоскости своими координатами.

Указание. Примерный вид программы:

```
x1=float(input("x1= "))
y1=float(input("y1= "))
x2=float(input("x2= "))
y2=float(input("y2= "))
r=((x1-x2)**2+(y1-y2)**2)**(1/2)
print("расстояние= ", r)
```

4. Составить программу вычисления суммы первых n членов арифметической прогрессии по любым двум её членам, номера которых известны.

Выводы

В ходе выполнения лабораторной работы вы получили представление о написании простых программ на языке программирования Python.

Контрольные вопросы

1. Какие основные операторы языка Python вы использовали при выполнении лабораторной работы?
2. Для чего используется оператор `print` при работе в языке Python?
3. Как можно возвести число в квадрат в языке Python?

Лабораторная работа 3.1. Условный оператор `if`

Теоретическая часть

При решении задач важно реализовывать возможность выбора среди альтернативных операций на основе результатов проверки. В императивных языках программирования для этих целей используется оператор ветвления (условный оператор). В языке Python подобный оператор предусматривает возможность сделать выбор как из двух альтернативных ветвей программы, так и из трёх и более.

Общая форма условного оператора:

```
if <условие1>:
    оператор1
    elif <условие2>:
        оператор2
    else:
        оператор3
```

Части `else` и `elif` являются необязательными. После части `if` указывается логическое условие, которое может быть истинным или ложным.

Как видно из описания условного оператора, он может содержать другие операторы, например операторы and (конъюнкция), or (дизъюнкция), `not` — отрицание, равенство `==`, неравенство `!=`. Также в Python можно записывать двойное условие, например `2<=a<=5, -10<v<=9`. В том числе условный оператор может содержать внутри себя другой условный оператор.

Стоит обратить внимание на то, что после логического условия стоит двоеточие, для того чтобы показать, что далее идёт блок выражений. Блок выражений записывается с отступом.

Рассмотрим работу условного оператора более подробно.

В самом простом случае оператор ветвления имеет вид:

```
if <условие>:
    оператор1
```

Это неполная форма условного оператора.

Блок-схема работы данного оператора представлена на рисунке 55.

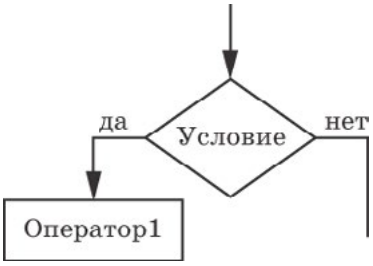


Рис. 55. Блок-схема неполного условного оператора

В случае истинности условия выполняется `оператор1`, а затем осуществляется выход из условного оператора (управление передаётся оператору, следующему за оператором `if`).

Пример 1

```
a=int(input())
if a<0:
    print('Ниже')
```

Результат работы программы представлен на рисунке 56.

Рис. 56. Результат работы программы

В данном примере в качестве условия используется сравнение a<0. Если это условие истинно, то на экран выводится текст «Ниже». Если же условие ложно, то программа ничего не выполняет.

Пример 2

```
a=int(input())
if (a<0) and (a>=-3):
    print('Ниже')
```

Результат работы программы представлен на рисунке 57.

Рис. 57. Результат работы программы

В данном примере условие составное: состоит из двух условий, объединённых операцией `and` (логическое «и»).

Рассмотрим далее более сложный вид оператора ветвления:

```
if <условие>:
    оператор1
else:
    оператор2
```

Обратите внимание на порядок отступов в формате оператора!
Это полная форма условного оператора.
Блок-схема работы данного оператора представлена на рисунке 58.

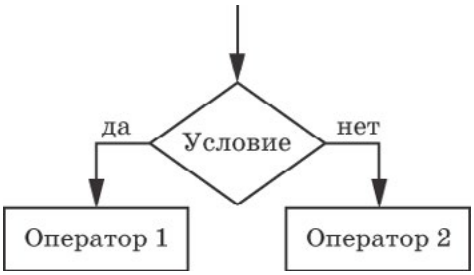


Рис. 58. Блок-схема полной формы условного оператора

Если условие истинно, то выполняется `оператор1`, в противном случае (если условие ложно), выполняется `оператор2`. Далее управление переходит к оператору, который следует за условным оператором.

Приведём примеры работы такой формы условного оператора.

Пример 3

```
a=int(input())
b=int(input())
if a+b>10:
    print('Yes')
else:
    print('No')
```

Результат работы программы представлен на рисунке 59.

Рис. 59. Результат работы программы

В данном примере в случае истинности условия `a+b>10` выполняется оператор `print('Yes')`, в противном случае — `print('No')`.

После условия и после части `else` можно указывать несколько операторов, но тогда все они записываются с отступом! Для сравнения рассмотрим два примера.

Пример 4.1

```
a=10
b=15
c=3
if a+b>10:
    print('Yes')
else:
    print('No')
    c=a+b
print(c)
```

Результат работы программы представлен на рисунке 60.

Рис 60. Результат работы программы

Пример 4.2

```
a=10
b=15
c=3
if a+b>10:
    print('Yes')
else:
    print('No')
c=a+b
print(c)
```

Результат работы программы представлен на рисунке 61.

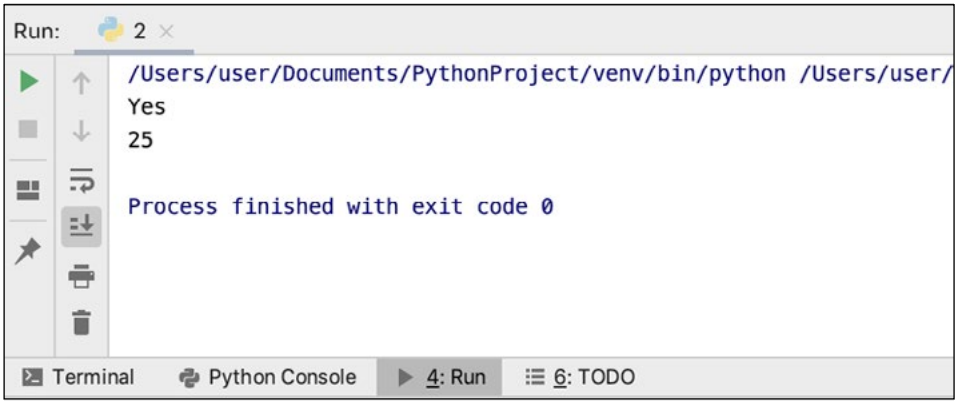


Рис. 61. Результат работы программы

Как видим, результаты работы программ различаются, потому что в примере 4.2 оператор c=a+b выполняется в любом случае, так как стоит вне оператора ветвления.

Третья форма оператора ветвления выглядит следующим образом:

```
if <условие1>:
    оператор1
elif <условие2>:
    оператор2
else:
    оператор3
```

Блок-схема работы данного оператора представлена на рисунке 62.

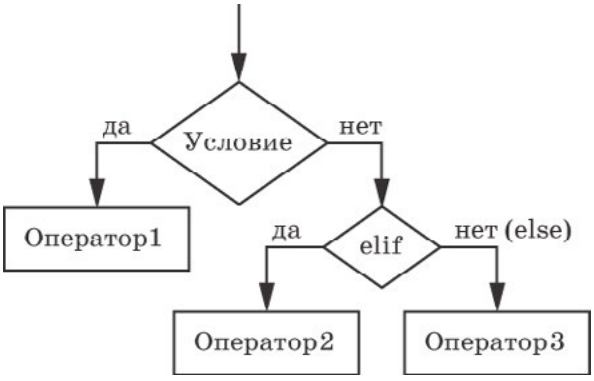


Рис. 62. Блок-схема условного оператора с двумя условиями

При использовании данной формы можно проводить проверку нескольких условий — после `if` и после `elif`. Оператор после else выполняется в том случае, если не выполнилось `условие2` после части `elif`.

Рассмотрим пример использования данной формы оператора `if`.

Пример 5

```
a=10
if a<-5:
    print('Yes')
elif -5<=a<=5:
    print('Maybe')
else:
    print('No')
```

Результат работы программы представлен на рисунке 63.

Рис. 63. Результат работы программы

Пример 6

Написать программу вычисления стоимости покупки с учётом скидки. Скидка в 3% предоставляется в том случае, если стоимость покупки превышает 1000 р., а скидка в 5% — если стоимость выше 3000 р.

```
s=int(input())
if s<1000:
    print(s)
elif 1000<=s<3000:
    print(s*0.97)
else:
    print(s*0.95)
```

Результат работы программы представлен на рисунке 64.

Рис. 64. Результат работы программы

Ещё раз обратимся к синтаксическим правилам языка Python.

В Python отсутствуют фигурные скобки (которые есть в языке C/C++) или разделители `begin/end` (которыми оперирует язык Pascal), окружающие блоки программного кода. Вместо этого принадлежность операторов к вложенному блоку определяется по величине отступов. Также операторы в языке Python обычно не завершаются точкой с запятой; признаком конца оператора служит конец строки с этим оператором.

Все составные операторы в языке Python оформляются одинаково: строка с заголовком завершается двоеточием, далее следуют вложенные операторы (один или более), обычно с отступом относительно заголовка. Эти операции с отступами называются блоком (или иногда набором).

Важно!

Интерпретатор автоматически определяет границы блоков по величине отступов, т. е. по ширине пустого пространства слева от программного кода. Все операторы, смещенные вправо на одинаковое расстояние, принадлежат к одному и тому же блоку кода.

В операторе `if` разделы `elif` и `else` не только являются частями оператора `if`, но и содержат собственные вложенные блоки. На рисунке 65 представлена схема с отступами для оператора `if`.

`if`	
	Блок 1
`elif`	
	Блок 2
`else`	
	Блок 3

Рис. 65. Схема вложенных блоков

Практическая часть

Цель работы: ознакомление с условным оператором `if` языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.

2. Написать программу вычисления оптимального веса: оптимальный вес вычисляется по формуле: рост (см) — 100.

3. Пользователю выдаётся рекомендация о снижении или наборе веса.

Указание. Примерный вид программы:

```
v=int(input("Ваш вес – "))
h=int(input("Ваш рост – "))
ideal_v=h-100
if v==ideal_v:
    print("Ваш вес в норме")
elif v>ideal_v:
    print("Ваш вес выше нормы! Необходимо сбросить вес")
else:
    print("Ваш вес ниже нормы! Необходимо набрать вес")
```

Результат работы программы представлен на рисунке 66.

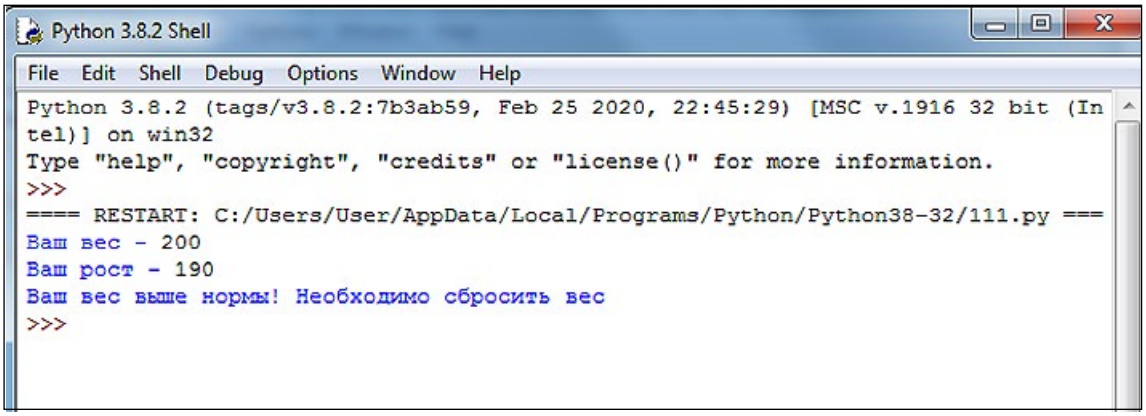


Рис. 66. Результат работы программы

4. Написать программу, определяющую вид треугольника (равносторонний, равнобедренный, разносторонний) по заданным длинам сторон.

5. Написать программу, которая считывает три вещественных числа и заменяет каждое чётное значение его частным от деления на 2, а единицу — числом 2.

6. Определить, является ли введённый пользователем год високосным.

Указание. Примерный вид программы:

```
y = int(input("Введите год "))
if (y%4==0 and y%100!=0) or (y%400==0):
    print("Високосный")
else:
    print("Обычный")
```

Написать программу, проверяющую, является ли введённое число чётным или оно нечётное.

Выводы

В ходе выполнения лабораторной работы вы получили представление о составлении условных алгоритмов с использованием оператора `if-elif-else` языка программирования Python.

Контрольные вопросы

1. Для чего используются условные операторы в программировании?
2. Как выглядит синтаксис условного оператора в языке Python?
3. Какие формы условного оператора можно использовать в языке Python?

Лабораторная работа 3.2. Условный оператор `if`

Теоретическая часть

Воспользоваться материалами из лабораторной работы 3.1.

Практическая часть

Цель работы: ознакомление с условным оператором if языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Среди трёх чисел найти среднее, т. е. которое больше одного числа, но меньше другого.
3. Даны длины трёх сторон одного треугольника и трех стороны другого треугольника. Определить, будут ли эти треугольники равновеликими, т. е. имеют ли они равные площади.
4. Ввести рост человека. Вывести на экран текст «ВЫСОКИЙ», если рост превышает 180 см, и «НЕ ОЧЕНЬ ВЫСОКИЙ» в противном случае.
5. Ввести 3 числа. Вывести их в порядке возрастания. (Пример: 12, 34, 56.)

Выводы

В ходе выполнения лабораторной работы вы получили представление о составлении условных алгоритмов с использованием оператора `if-elif-else` языка программирования Python.

Контрольные вопросы

1. Как выглядит полная форма оператора ветвления в языке Python?
2. Для чего используется часть else в операторе ветвления?
3. Какие основные операторы вы использовали при решении задач из лабораторной работы?

Лабораторная работа 4.1. Циклы в языке Python

Теоретическая часть

Справочник

Цикл в языке программирования представляет собой конструкцию, многократно выполняющую одну и ту же группу операторов. Число повторений (итераций) цикла может быть либо задано заранее, либо зависеть от истинности некоторого условия.

В реальной жизни постоянно применяются циклы, поэтому циклический алгоритм часто используются при решении задач по программированию.

В языке программирования Python может быть реализовано два вида цикла:

1) с предусловием — цикл `while`;
2) с параметром — цикл `for`.

Цикл `while` является часто используемым и универсальным циклом в Python. Полный формат данного цикла:

```
while <условие>:
    <оператор1>
else:
    <оператор2>
```

Часть `else` является необязательной. Блок-схема работы цикла `while` представлена на рисунке 67.

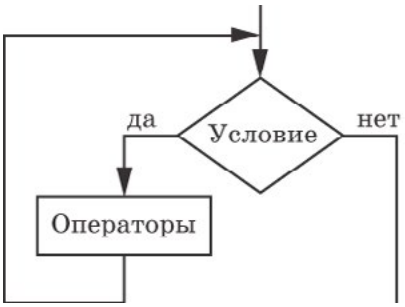


Рис. 67. Блок-схема цикла с предусловием `while`

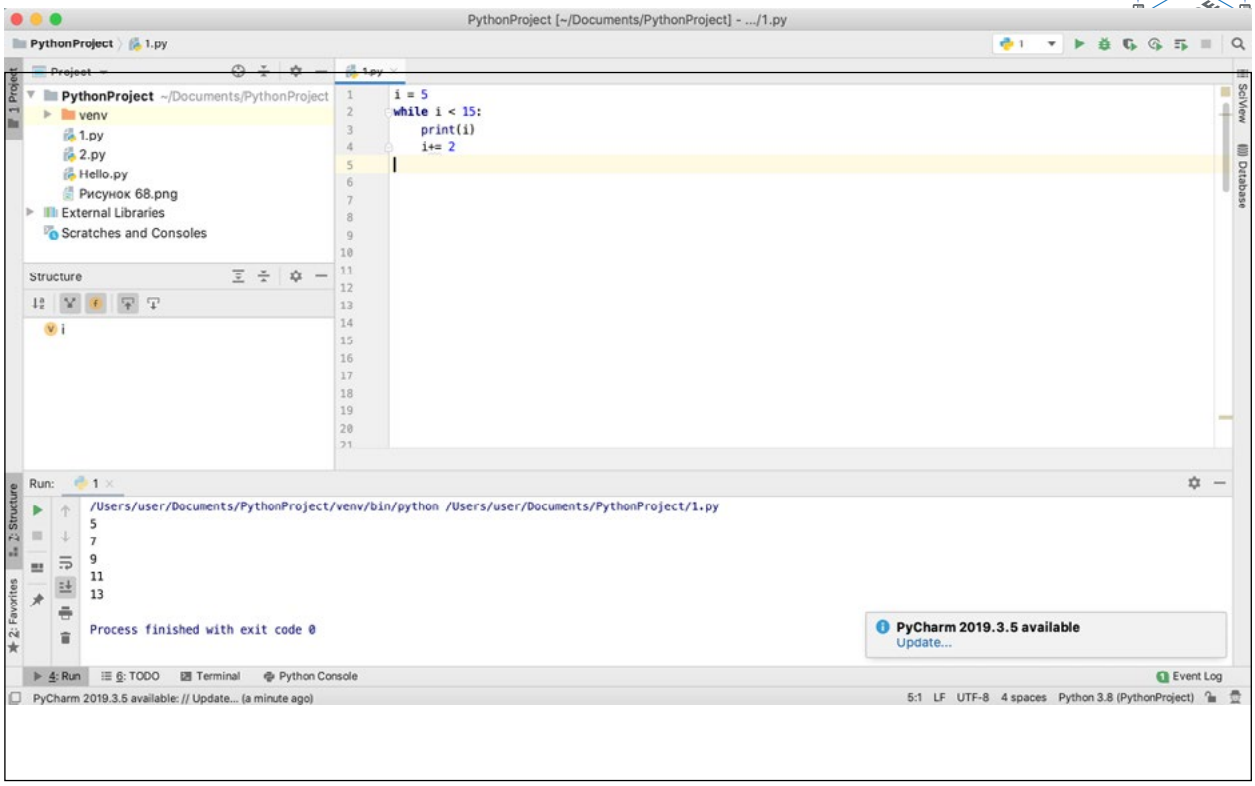
Выполнение цикла `while` начинается с проверки условия. Если оно истинно (не равно `false`), выполняется оператор цикла. Если при первой же проверке выражение в условии равно `false`, цикл не выполнится ни разу. Если условие в цикле `while` никогда не станет ложным, то не будет причин остановки цикла и программа «зациклится». Чтобы этого не произошло, необходимо организовать момент выхода из цикла, т. е. ложность выражения в условии. Так, например, изменяя значение какой-нибудь переменной в теле цикла, можно довести логическое выражение до ложности. Обратите внимание, что операторы тела цикла должны быть записаны с отступом.

Пример 1

```
i=5
while i<15:
    print(i)
    i+=2
```

Интерфейс программы PyCharm с введённой программой и результатом выполнения представлен на рисунке 68.

Рис. 68. Вид программы в среде разработки с результатом выполнения



В данном примере организован перебор значений переменной `i` с шагом 2. Условие работы цикла: `i<15`. В теле цикла происходит изменение (увеличение) переменной `i`, поэтому цикл не будет бесконечным.

Результат работы программы отдельно представлен на рисунке 69.

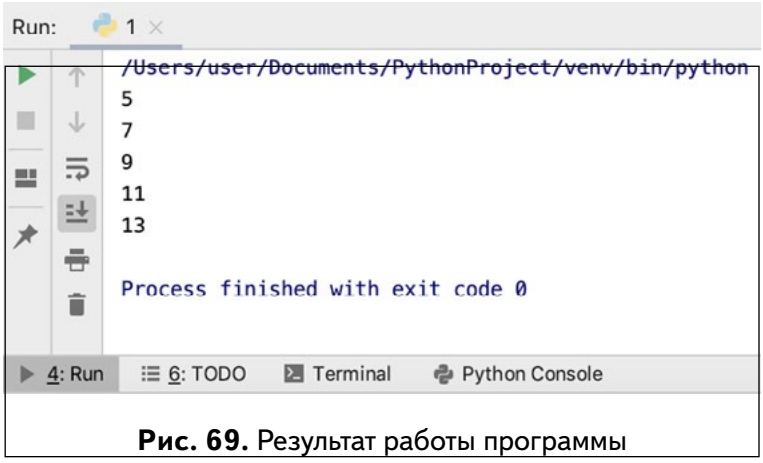


Рис. 69. Результат работы программы

Пример 2

```
a=0
while a<7:
    print("Python")
    a+=1
```

В данном примере цикл выполняется, пока истинно условие `a<7`, значение переменной `a` меняется в теле цикла.

Результат работы программы представлен на рисунке 70.

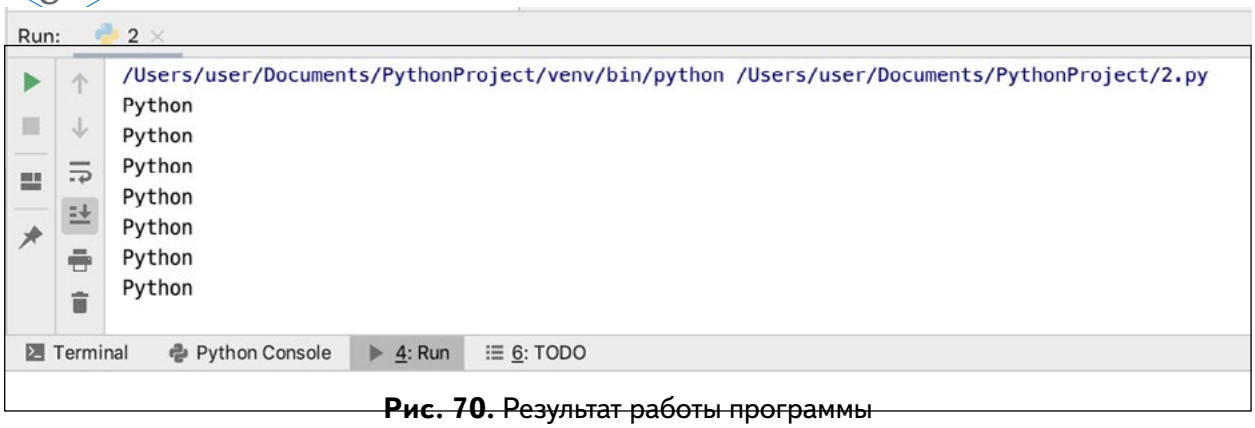


Рис. 70. Результат работы программы

Второй цикл, используемый в языке Python, — цикл с параметром. Синтаксис данного цикла:

```
for <переменная> in <объект>:
    <оператор1>
else:
    <оператор2>
```

Блок-схема работы цикла представлена на рисунке 71.

Рис. 71. Блок-схема цикла с параметром

Этот цикл перебирает заданную последовательность значений любого итерируемого объекта (например, строки или списка) и для каждого значения выполняет тело цикла. Цикл выполняется заданное число раз. Для обращения к текущему элементу последовательности обычно используется переменная цикла, её иногда называют управляющей переменной.

Часто для организации работы цикла с параметром `for` используется функция `range`.

Функция `range()` возвращает последовательность чисел, регулируемую переданными в неё аргументами. Возможны следующие варианты обращения к данной функции:

1) `range(finish)`
2) `range(start, finish)`
3) `range(start, finish, step)`

Здесь `start` — это первый элемент последовательности (включительно), `finish` — последний (не включительно), а `step` — разность между следующим и предыдущим членами последовательности.

Например, `range(5)` возвращает последовательность 0, 1, 2, 3, 4.

Вызов `range(2,8)` возвращает последовательность 2, 3, 4, 5, 6, 7.

Рассмотрим примеры организации работы цикла с параметром.

Пример 3

```
for a in range(10):
    print(a)
```

В данном примере цикл выводит на экран последовательность чисел от 0 до 9 включительно.

Результат работы программы представлен на рисунке 72.

Рис. 72. Результат работы программы

Пример 4

```
for c in range(0,22,3):
    print(c,end=" ")
```

В данном примере на экран выводится последовательность чисел от 0 до 21 с шагом 3 в строку через пробел.

Результат работы программы представлен на рисунке 73.

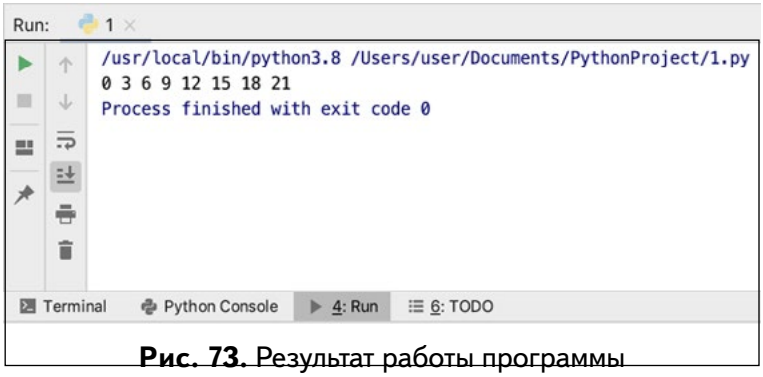


Рис. 73. Результат работы программы

Также, говоря про работу циклов в языке Python, необходимо упомянуть про операторы `continue, break, else`.

Справочник

Оператор `continue` используется для перехода на следующую итерацию цикла, пропуская следующие после `continue` операторы тела цикла.

Пример 5

```
for i in range(10):
    if i==5:
        continue
    print(i*2,end=" ")
```

Результат работы программы представлен на рисунке 74.

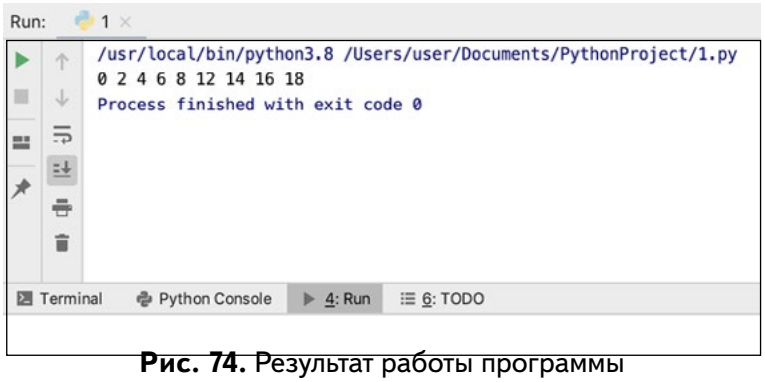


Рис. 74. Результат работы программы

В данном примере при равенстве `i==5` срабатывает оператор `continue`, в результате чего для `i`, равного 5, пропускается оператор `print(i*2,end=" ")`. Поэтому число 10 не выводится на экран.

Справочник

Оператор `break` используется для организации немедленного выхода из цикла. Это означает, что происходит досрочное завершение работы цикла.

Пример 6

```
for i in range(10):
    if i==5:
        break
print(i*2, end=" ")
```

Результат работы программы представлен на рисунке 75.

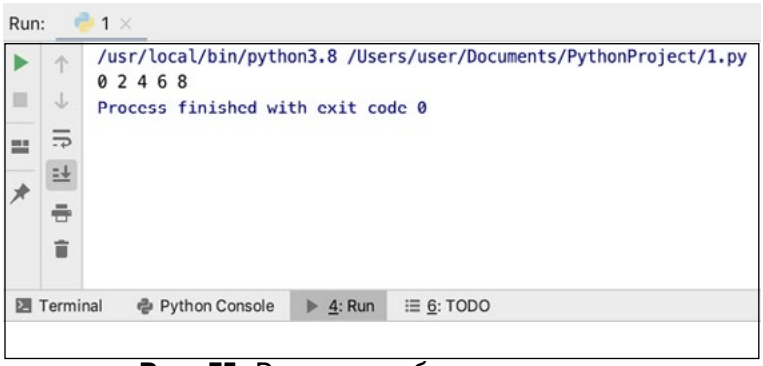


Рис. 75. Результат работы программы

В данном примере при равенстве `i==5` срабатывает оператор `break`, в результате чего происходит завершение работы цикла. Значит, последнее значение `i`, рассмотренное в теле цикла, будет равно 4.

Оператор `else` используется для проверки, был ли произведён выход из цикла посредством оператора `break` или же цикл завершился иным образом.

Пример 7

```
for i in range(10):
    if i==20:
        break
    print(i*2,end=" ")
else:
    print("значение не найдено")
```

Результат работы программы представлен на рисунке 76.

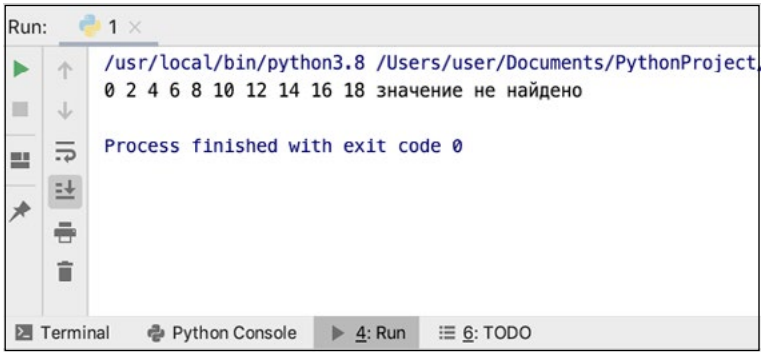


Рис. 76. Результат работы программы

В данном примере после вывода на экран последовательности от 0 до 18 на экран также выводится строка «значение не найдено», так как оператор `break` не сработал.

Также в языке Python возможно использование вложенных циклов, когда есть один внешний цикл и один или несколько вложенных. Стоит отметить, что использование вложенных циклов может замедлить работу программы.

Приведём ещё несколько примеров работы с циклами `for` и `while`.

Пример 8

На тренировке спортсмен ежедневно пробегает некоторую дистанцию, с каждым днём увеличивая её на 10%. Составить программу, определяющую по расстоянию, преодолённому спортсменом в первый день тренировки, длину дистанции на *k*-й день.

```
n= float(input("Введите начальную дистанцию "))
k= int(input("Введите количество дней "))
for i in range(k):
    n+=n*0.1
print("конечная дистанция ", n)
```

Результат работы программы представлен на рисунке 77.

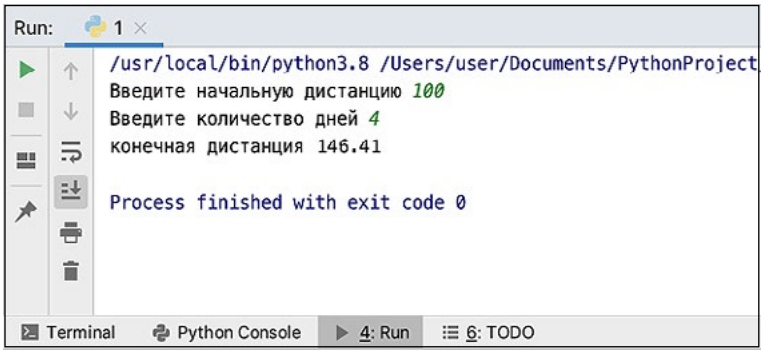


Рис. 77. Результат работы программы

В данной задаче используется цикл `for`, так как известно количество повторов цикла — `k` дней. Внутри цикла идёт увеличение переменной `n`, обозначающей длину дистанции: `n+=n*0.1`.

Пример 9

Перевести введённое пользователем десятичное число в двоичное. Известно, что число меньше 256.

```
dec=int(input("Введите десятичное число "))
v=128
for i in range(1,9):
    if dec>=v:
        print('1',end="")
        dec=dec-v
    else:
        print('0',end="")
    v=v//2
```

Результат работы программы представлен на рисунке 78.

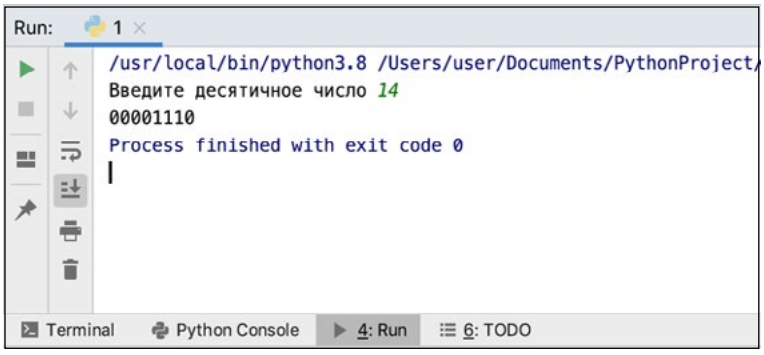


Рис. 78. Результат работы программы

В данном примере в переменной `v` задаётся вес старшего разряда двоичного числа. Так как по условию число должно быть меньше 256, то старший разряд будет иметь вес 128. Для перевода десятичного числа из него каждый раз вычитается вес старшего разряда, если это возможно, затем вес разряда уменьшается в 2 раза.

Пример 10

Разложить натуральное число на простые множители.

```
k=int(input("Введите число "))
print(k,'= ')
l=2
while not(k==1):
    if k%l==0:
        k=k/l
        print(l,end="")
    else:
        l+=1
```

Результат работы программы представлен на рисунке 79.

Рис. 79. Результат работы программы

В данном примере первое простое число `l=2`. В цикле `while` введенное число `k` несколько раз делится на потенциальный простой делитель `l`, если целочисленное деление не может быть выполнено, то ищется следующий простой делитель.

Практическая часть

Цель работы: ознакомление с операторами цикла `while`, `for` в языке программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Население города на 2021 г. насчитывало 620 тыс. человек. Считая темп прироста населения за год равным 3,7%, определить, в каком году оно превысит 1,5 млн человек.
3. Найти сумму нечётных делителей введённого с клавиатуры натурального числа.
4. Найти все натуральные числа из отрезка [1; 200], у которых количество делителей равно *n* (где *n* вводится с клавиатуры).

Указание. Примерный вид программы:

```
n=int(input("Введите кол-во делителей "))
for i in range(1,201):
    k=2
    for j in range (2,i):
        if i%j==0:
            k+=1
    if k==n:
        print(i)
```

5. Найти все четырёхзначные числа, у которых сумма крайних цифр равна сумме средних (например, 3221).

6. Найти все двухзначные числа, которые при умножении на 2 заканчиваются на 8, а при умножении на 3 — на 4.

Выводы

В ходе выполнения лабораторной работы вы получили представление о составлении циклических алгоритмов с использованием операторов `while`, `for` языка программирования Python.

Контрольные вопросы

1. Для чего используются циклы в языке программирования?
2. Какие виды циклов реализованы в языке Python?
3. Каков синтаксис оператора цикла `while`?
4. Каков синтаксис оператора цикла `for`?
5. Для чего используется оператор `break` внутри тела цикла?

Лабораторная работа 4.2. Циклы в языке Python

Теоретическая часть

Воспользоваться материалами из лабораторной работы 4.1.

Практическая часть

Цель работы: ознакомление с операторами цикла while, for языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.

2. Два числа называются дружественными, если каждое равно сумме делителей другого, исключая само это число. Проверить, являются ли два введённых числа дружественными.

Указание. Примерный вид программы:

```
n=int(input("Введите первое число "))
m=int(input("Введите второе число "))
s1=0
for i in range(1,n):
    if n%i==0:
        s1+=i
s2=0
for i in range(1,m):
    if m%i==0:
        s2+=i
if (s1==m) and (s2==n):
    print("yes")
else:
    print("no")
```

3. Вывести на экран все четырёхзначные числа, у которых первая и последняя цифры равны.

4. На тренировке спортсмен ежедневно пробегает некоторую дистанцию, с каждым днём увеличивая её на 10%. Составить программу, определяющую по расстоянию, преодолённому спортсменом в первый день, количество тренировок, после которых ежедневная дистанция превысит *s* км.

5. Вывести на экран таблицу умножения на *n*, где *n* вводится с клавиатуры.

Выводы

В ходе выполнения лабораторной работы вы получили представление о составлении циклических алгоритмов с использованием операторов while, for языка программирования Python.

Контрольные вопросы

1. Какие основные операторы языка Python вы использовали при решении задач лабораторной работы?
2. Какой оператор используется для организации цикла с предусловием в языке Python?
3. Как организовать цикл с постусловием в языке Python?

Лабораторная работа 5.1. Списки в языке Python

Теоретическая часть

Список в языке Python представляет собой структуру для хранения объектов различных типов. Элементы одного списка могут иметь одинаковый тип, но допускаются и смешанные типы. Элементы списка заключаются в квадратные скобки. Все элементы проиндексированы (пронумерованы), начиная с 0.

Примеры списков показаны в таблице 11.

Таблица 11

Примеры списков в Python

Список	Описание
`L=[]`	Пустой список
`L=[5, 10, 15, 20]`	Четыре элемента с индексами 0..3
`L=['pqr',['abc','xyz']]`	Вложенные списки

Важно!

Списки в Python, с одной стороны, похожи на массивы в других языках программирования, например Pascal, но, с другой стороны, имеется ряд отличий. Например, Python размещает элементы списка в памяти, затем размещает указатели на эти элементы. Таким образом, список в Python — это массив указателей.

Опишем ряд существенных свойств списков в языке Python.

Можно обращаться к элементу списка по его индексу, начиная с 0. В языке Python также используют отрицательную индексацию, которая начинается с индекса −1, при этом индекс −1 относится к последнему элементу списка. В таблице 12 представлены различные виды индексации элементов списка.

Таблица 12

Варианты индексации списков в Python

Список L	**8**	**–2**	**6**	**1**	**0**	**2**
Индекс	0	1	2	3	4	5
Отрицательный индекс	−6	−5	−4	−3	−2	−1

Например, элемент `L[3]` равен 1, а элемент `L[-4]` равен 6.

Важно!

Список в языке Python может иметь переменную длину.

Например, можно удалять элементы списка. Оператор `del list[i]` удаляет из списка `list` элемент с индексом `i`, `del l[2]` удаляет элемент с индексом 2.

Результаты работы этого оператора представлены на рисунке 80.

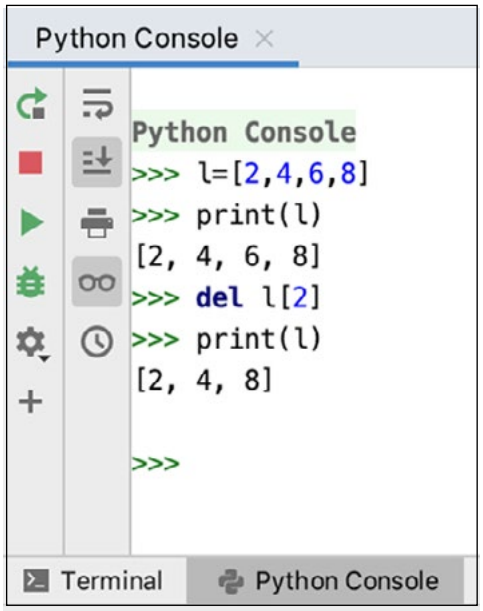


Рис. 80. Удаление элемента из списка

Далее рассмотрим способы задания списка в языке Python.

Можно непосредственно задать список, перечислив его элементы в квадратных скобках через запятую:

```
l=[4,8,10]
```

Также можно использовать функцию `range`, описанную ранее:

```
a=list(range(20))
```

В результате сформируется список значений от 0 до 19.

Для задания элементов списка случайным образом можно использовать функцию `randint` из модуля `random`.

Пример 1

```
from random import randint
a=[randint(-10,10) for i in range(20)]
print(a)
```

В данном примере формируется список из 20 элементов, заданных случайным образом.

Результаты работы программы представлены на рисунке 81.

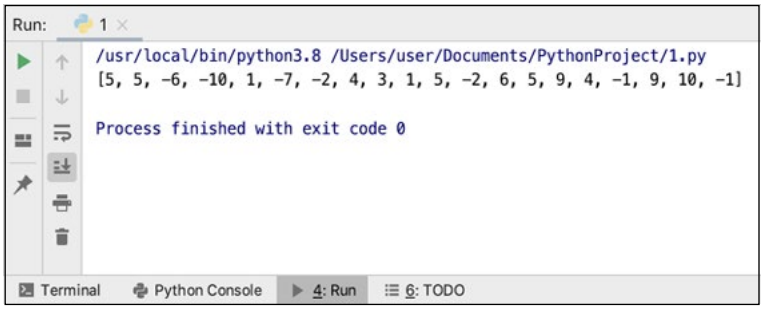


Рис. 81. Вид списка, заданного случайным образом

Ещё один вариант — задать список с клавиатуры:

```
a=[input() for i in range (10)]
```

Пример 2

Для формирования списка с клавиатуры можно использовать следующий вариант:

```
a=[]
n=int(input())
for i in range(n):
    k=int(input())
    a.append(k)
print(a)
```

Результат работы программы представлен на рисунке 82.

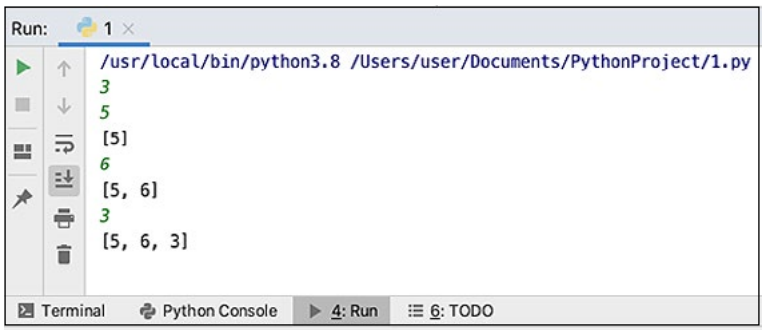


Рис. 82. Результат работы программы

Как видно из последнего примера, для обработки элементов списка часто используют циклы, в частности цикл `for`.

Пример 3

Для получения доступа к элементам списка можно использовать следующие конструкции.

```
for x in ["abc", "cde", "efg"]:
    print(x)
```

или

```
l=["abc", "cde", "efg"]
for x in l:
    print(x)
```

Результат работы программы представлен на рисунке 83.

Рис. 83. Результат работы программы

При работе со списками в языке Python часто используют срезы.

Эти конструкции нужны, чтобы обрезать список, взяв лишь те элементы, которые нам требуются. Срезы работают по следующей схеме:

```
list[начало:конец:шаг]
```

Здесь:

`начало` указывает, с какого элемента нужно начать (по умолчанию равно 0);

`конец` указывает, по какой элемент берутся элементы (по умолчанию равен длине списка);

`шаг` определяет, с каким шагом берутся элементы, например каждый второй или третий (по умолчанию каждый первый).

Приведём примеры работы со срезами.

Пример 4

```
l=[2,4,6,8,10,12,14,16]
print(l[2::2])
```

Результат работы программы представлен на рисунке 84.

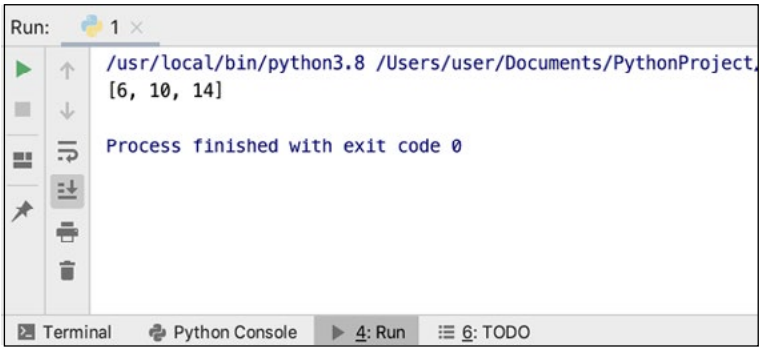


Рис. 84. Результат работы программы

Из списка берётся каждый второй элемент, начиная со второго.

Пример 5

```
l=[2,4,6,8,10,12,14,16]
print(l[::3])
```

Результат работы программы представлен на рисунке 85.

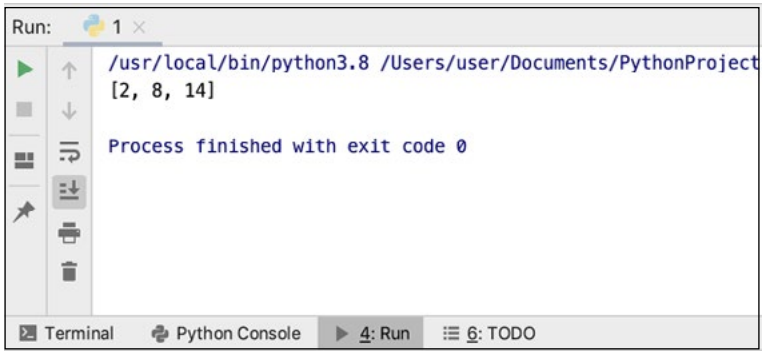


Рис. 85. Результат работы программы

Из списка берётся каждый третий элемент.
Также с помощью срезов можно менять существующие списки.

Пример 6

```
l=[2,4,6,8,10,12,14,16]
l =l[1:6:1]
print(l)
```

Результат работы программы представлен на рисунке 86.

Рис. 86. Результат работы программы

В списке остаются только элементы со второго по шестой.

В языке Python доступно достаточно много встроенных функций для обработки списков. Некоторые из них описаны в таблице 13.

Таблица 13

Основные функции (методы) обработки списков

Функция	Описание
`append(x)`	Добавляет элемент `x` в конец списка
`clear()`	Очищает список
`copy()`	Возвращает копию списка
`count(x)`	Возвращает количество элементов списка со значением `x`

Окончание

Функция	Описание
`extend(L)`	Расширяет список путем добавления в него всех элементов списка `L`
`index(x[,start[,end]])`	Возвращает индекс первого элемента со значением `x` (при этом поиск ведётся от `start` до `end`; если `start` и `end` не указываются, то начиная с нулевой позиции)
`insert(i,x)`	Вставляет в список на `i`-ю позиция значение `x`
`pop([i])`	Удаляет из списка i-й элемент и возвращает его. Если индекс не указан, удаляется последний элемент
`remove(x)`	Удаляет первый элемент в списке, имеющий значение `x`. Возвращает `ValueError`, если такого элемента не существует
`reverse()`	Переворачивает список
`sort()`	Сортирует список

(Квадратные скобки в синтаксисе функций означают необязательные элементы.)

Поскольку данные функции являются методами класса «список», то обращение к ним имеет вид:

```
<имя списка>. <имя метода>
```

Рассмотрим несколько примеров по работе с описанными функциями.

Пример 7

```
l=[2,4,6,8,10,12,14,16]
print(l.index(4))
```

Результат работы программы представлен на рисунке 87.

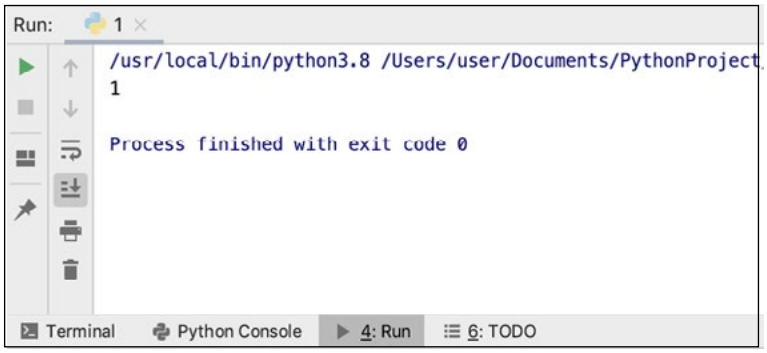


Рис. 87. Результат работы программы

На экран выведется индекс элемента, равного 4. В данном случае это 1.

Пример 8

```
l=[2,4,6,8,10,12,14,16]
print(l.count(12))
```

Результат работы программы представлен на рисунке 88.

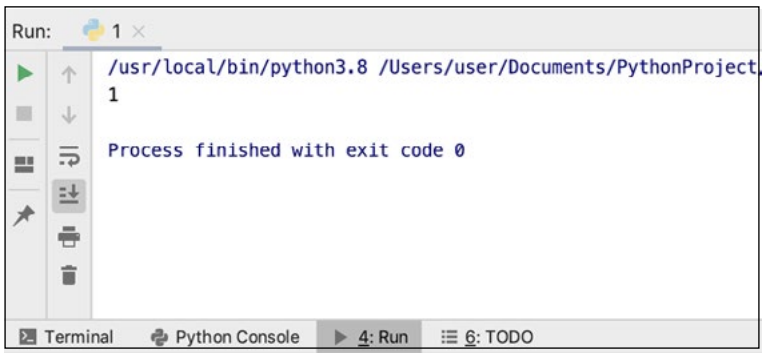


Рис. 88. Результат работы программы

На экран выведется количество элементов в списке, равных 12. В данном случае это один элемент.

Пример 9

```
l=[2,4,6,8,10,12,14,16]
l.append(0)
print(l)
```

Результат работы программы представлен на рисунке 89.

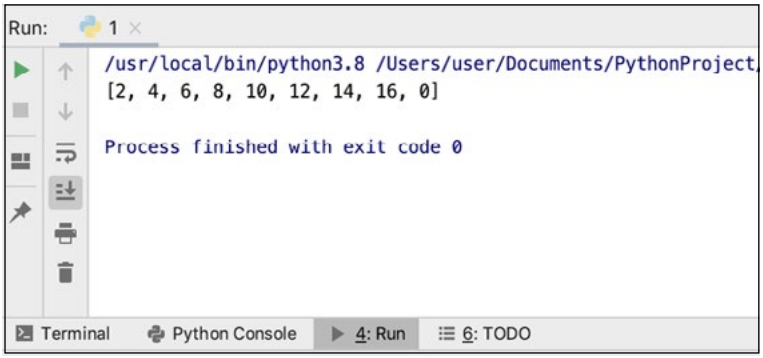


Рис. 89. Результат работы программы

В конец списка добавляется новый элемент — 0.

Пример 10

```
l=[2,4,6,8,10,12,14,16]
z=[3,7,10]
l.extend(z)
print(l)
```

Результат работы программы представлен на рисунке 90.

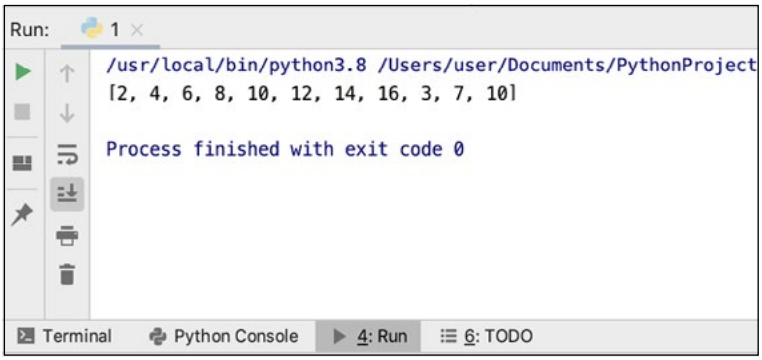


Рис. 90. Результат работы программы

Список `l` расширяется за счет добавления элементов списка `z`.

Далее рассмотрим несколько примеров задач, при решении которых используются списки.

Пример 11

Задать числовой список случайным образом. Заменить значения элементов в списке на их квадраты.

```
from random import randint
z=[randint(-10, 10) for i in range(10)]
print(z)
print("новый список:")
z=[x*x for x in z]
print(z)
```

Результат работы программы представлен на рисунке 91.

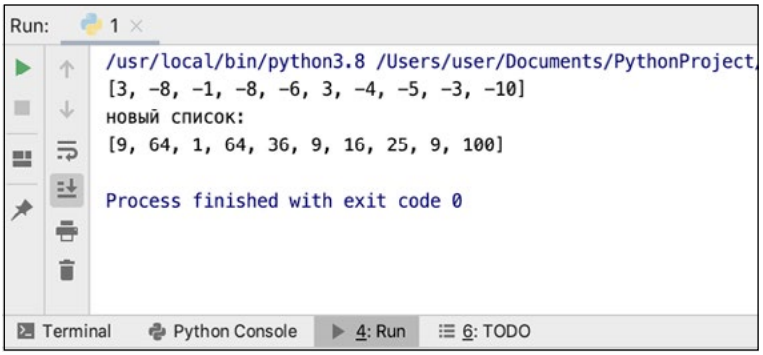


Рис. 91. Результат работы программы

В данном примере задан список из 10 элементов, диапазон элементов списка — от −10 до 10. Затем элементы списка заменяются с использованием перебора с помощью цикла for.

Пример 12

Задан числовой список, вывести на экран те значения списка, которые встречаются в нём более одного раза.

```
c=[3,8,33,-12,16,3,-9,16]
```

```
z=c
print(c)
i=0
while i<len(z):
    if z.count(z[i])>1:
        print(z[i])
        z.remove(z[i])
    i+=1
```

Результат работы программы представлен на рисунке 92.

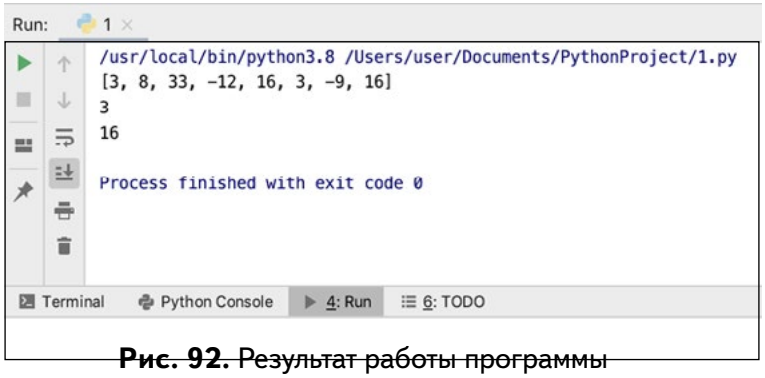


Рис. 92. Результат работы программы

В данном примере используется несколько встроенных функций для работы со списками: `count`, `len`, `remove`. `count` возвращает количество вхождений элемента `z[i]` и записывает это количество в тот же список `z`. Если количество вхождений больше одного, то элемент `z[i]` выводится на экран и удаляется из списка, чтобы избежать повторов при выводе, если элемент входит в список, например 3 раза.

Пример 13

Даны два числовых списка. Составить третий список, в который входят только те элементы второго списка, которые не входят в первый список.

```
L1=[1,4,9,16,25,36,49,64]
L2=[1,2,3,4,5]
L3=[]
for i in range(0,len(L2)):
    if L1.count(L2[i])==0:
        L3.append(L2[i])
print(L3)
```

Результат работы программы представлен на рисунке 93.

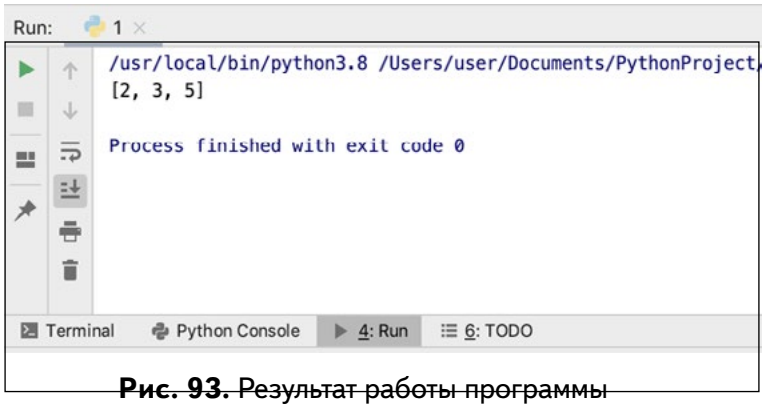


Рис. 93. Результат работы программы

В данном примере используется несколько встроенных функций для работы со списками: `append`, `count`, `len`. С помощью функции `count` находится количество вхождений элемента списка `L2` в список `L1`. Если это количество равно 0 (т. е. элемент не входит в список `L1`), то элемент добавляется в новый список `L3`.

Практическая часть

Цель работы: ознакомление с понятием «список» в языке программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.

2. Дан список некоторых целых чисел, поискать в нём значение 20 и, если оно присутствует, заменить его на 200.

3. Написать программу, которая выводит чётные числа из заданного списка и останавливается, если встречает число 15.

Указание. Примерный вид программы:

```
from random import randint
z=[randint(-10,10) for i in range(10)]
print(z)
for x in z:
    if x==15:
        break
    if x%2==0:
        print(x)
```

4. Заменить в списке нулевые элементы единицами.

5. Дан список из 20 элементов. Сформировать новый список, поместив в него только те элементы из первого списка, которые не превосходят введённое с клавиатуры число.

6. Найти в списке наибольший и наименьший элементы. Вывести их индексы на экран.

Указание. Примерный вид программы:

```
from random import randint
z=[randint(-10, 10) for i in range(10)]
print(z)
m1=max(z)
print("max=",m1)
m2=min(z)
print("min=",m2)
index_max=z.index(m1)
print("индекс max=»,index_max)
index_min=z.index(m2)
print("индекс min=",index_min)
```

Решение данной задачи осуществлено только за счёт встроенных функций языка Python для работы со списками. Максимальный элемент и его индекс были найдены с помощью функций `max` и `index`, аналогично найдены минимальный элемент и его индекс.

Можно ли выполнить решение данной задачи без встроенных функций? Такое решение будет напоминать решение в другом императивном языке программирования, например Pascal. Приведём пример.

```
from random import randint
z=[randint(-10,10) for i in range(10)]
print(z)
m1=z[0]
for x in z:
    if x>m1:
        m1=x
print("max=",m1)
for i in range (0,len(z)):
    if z[i]==m1:
        index_max=i
print("индекс max=»,index_max)
m2=z[0]
index_min=0
for i in range(0,len(z)):
    if z[i]<m2:
        m2=z[i]
        index_min=i
print("min=",m2)
print("индекс min=",index_min)
```

Как видим, второй вариант решения является более длинным и выполняться, соответственно, будет дольше за счёт нескольких обращений к циклу `for`.

7. Найти сумму элементов массива, находящихся между наибольшим и наименьшим элементами.

Указание. Примерный вид программы:

```
from random import randint
z=[randint(-10,10) for i in range(10)]
print(z)
index_max=z.index(max(z))
index_min=z.index(min(z))
s=0
if index_max>index_min:
    for i in range(index_min+1,index_max):
        s+=z[i]
else:
    for i in range(index_max+1,index_min):
        s+=z[i]
print("сумма=",s)
```

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе со списками в языке программирования Python.

Контрольные вопросы

1. Дайте определение списка в языке программирования Python. Приведите примеры списков.
2. Как можно обратиться к элементу списка в языке программирования Python?
3. Для чего используется срез при работе со списками в языке Python?
4. Какие основные встроенные функции для работы со списками в языке Python вы можете назвать?

Лабораторная работа 5.2. Списки в языке Python»

Теоретическая часть

Воспользоваться материалами из лабораторной работы 5.1.

Практическая часть

Цель работы: ознакомление с понятием «список» в языке программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Задан список `X` из 30 элементов. Поместить в список `Y` все отрицательные элементы списка `X` (в порядке их следования), сменив знак чисел, в список `Z` — все положительные элементы списка `X` (в порядке их следования). Подсчитать количество нулевых элементов.
3. Удалить из списка максимальный элемент.
4. Удалить из списка первый отрицательный элемент.
5. Задан список, содержащий как положительные, так и отрицательные числа. Составить программу перестановки отрицательных чисел в конец списка, а положительных — в начало.

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе со списками в языке программирования Python.

Контрольные вопросы

1. Какой структурой данных вы пользовались при решении задач лабораторной работы?
2. Какие основные операторы языка Python вы использовали при решении задач лабораторной работы?
3. Какой функцией надо воспользоваться для подсчёта количества элементов с данным значением в списке?

Лабораторная работа 5.3. Списки в языке Python

Теоретическая часть

Воспользоваться материалами из лабораторной работы 5.1.

Практическая часть

Цель работы: ознакомление с понятием «список» в языке программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Удалить из списка все нулевые значения.
3. Дан список `a` из 30 элементов. Найти максимальное из значений `a[0]+a[29]`, `a[2]+a[28]`,… , `a[14]+a[15]`.
4. Проверить, составляют ли элементы списка геометрическую прогрессию.
5. Даны два списка `A` и `B` по 12 элементов. Получить список `C` по правилу: `c[i]=max(a[i],b[i])`.

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе со списками в языке программирования Python.

Контрольные вопросы

1. Какие основные операторы языка Python вы использовали при решении задач лабораторной работы?

2. Какие основные функции языка Python для работы со списками вы использовали в своей работе?

3. Какие способы задания списка вы использовали в своей работе?

Лабораторная работа 6.1. Работа со строками в языке Python

Теоретическая часть

Строковые данные используются в языке Python для записи текстовой информации. Строки в Python представляют собой упорядоченные последовательности символов. Каждый символ имеет порядковый номер в таблице, которая называется кодовой таблицей. По умолчанию в языке Python используется стандарт кодовой таблицы Unicode. Строки в Python обрамляются кавычками или апострофами, при этом важно, чтобы с обоих концов строки использовались кавычки одного и того же типа.

Важно!

Важно отметить, что языке Python не используется символьный тип char, привычный для многих языков программирования, например Pascal, C++. То есть один символ, заключенный в апострофы, также представляет собой строку.

В таблице 14 приведены примеры задания строк.

Таблица 14

Примеры строк в Python

Задание строки	Описание
`S=''`	Пустая строка
`S='A'`	Строка, состоящая из одного символа
`S='Hello'`	Строка, состоящая из нескольких символов, заключена в апострофы
`S="good"`	Строка, состоящая из нескольких символов, заключена в кавычки

Также в строках в языке Python могут использоваться специальные непечатаемые символы. В таблице 15 представлен перечень таких символов, которые носят название экранированных последовательностей: это последовательности, начинающиеся с символа «\», за которым следует один символ или более.

Таблица 15

Экранированные последовательности

Экранированная последовательность	Описание
`\n`	Перевод строки
`\t`	Горизонтальная табуляция
`\v`	Вертикальная табуляция
`\r`	Возврат в начало строки

Пример 1

```
s="abc \n efg"
print(s)
```

Результат работы программы представлен на рисунке 94.

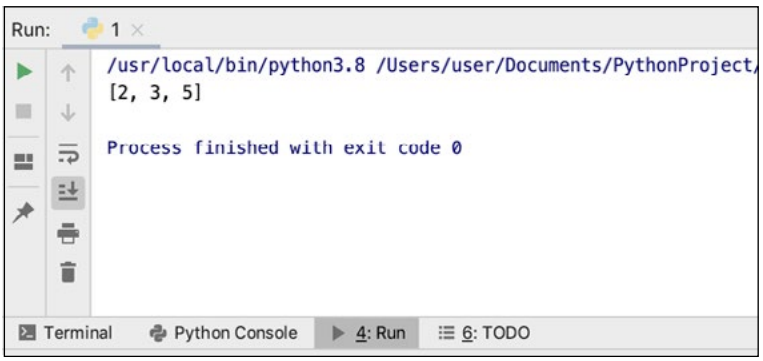


Рис. 94. Результат работы программы

Аналогично спискам, строки в языке Python являются упорядоченными последовательностями символов, следовательно, могут быть проиндексированы. Для обращения к отдельному символу в строке можно указать имя строки, за которым следует число (индекс) в квадратных скобках [ ]. Так же, как и в списках, нумерация символов начинается с индекса 0. Например, обращение `s[2]` к строке `s="hello"` возвращает символ «l». Индексы также могут быть представлены и отрицательными значениями (табл. 16).

Таблица 16

Варианты индексации строк в Python

Строка S	**l**	**e**	**s**	**s**	**o**	**n**
Индекс	0	1	2	3	4	5
Отрицательный индекс	–6	–5	–4	–3	–2	–1

Важно!

Следует обратить внимание на то, что строки в языке Python, в отличие от списков, являются неизменяемыми!

После того как строка будет создана, её нельзя будет изменить, т. е. все операции над строками производятся путём создания новых строк. Например, нельзя изменить отдельный символ строки: обращение `s[3]='x'` вызовет ошибку.

Так же, как и для списков, для строк в языке Python доступны различные срезы. Формат среза строки схож с форматом среза списка:

`str[начало:конец:шаг]`, где:

`начало` указывает, с какого элемента нужно начать (по умолчанию равно 0);

`конец` указывает, по какой элемент берутся элементы (по умолчанию равен длине списка);

`шаг` определяет, с каким шагом берутся элементы, например каждый второй или третий (по умолчанию каждый первый).

Приведём примеры работы со срезами строк.

Пример 1

```
word='strength'
print(word[4:6])
```

Выводит на экран 2 символа.
Результаты работы программы представлен на рисунке 95.

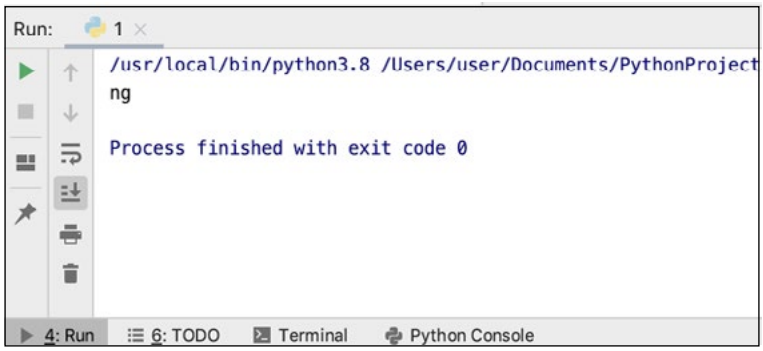


Рис. 95. Результат работы программы

Пример 2

```
word='strength'
print(word[:2])
```

Выводит на экран 2 символа.
Результат работы программы представлен на рисунке 96.

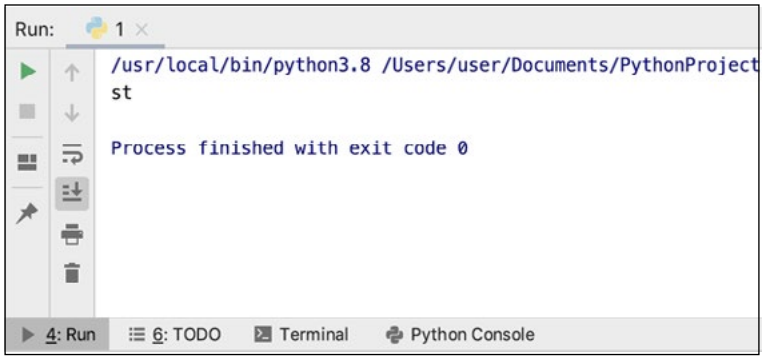


Рис. 96. Результат работы программы

Рассмотрим основные операции над строками, разрешённые в языке Python (табл. 17).

Таблица 17

Операции над строками в Python

Оператор	Описание	Пример

+	Сложение (конкатенация) строк. В результате применения возвращается строка, равная «склейке» указанных строк	`a='py'`
		`b='th'`
		`b='on'`
		`s=a+b+c`

Окончание

Оператор	Описание	Пример
*	Умножение строк. Оператор создаёт несколько копий строки, формат оператора: s*n или n*s, где s — это строка, а n — натуральное число	s='ab' sn=s*4
in	Оператор принадлежности, который возвращает True, если подстрока входит в строку, и False, если не входит	**if** 'z' **in** s: print(5)
>, <, >=, <=, ==, !=.	Сравнение строк	S1="ab" S2="xy" if S1>S2: print("Ok")

Пример 3

```
s1="abc"
s2="123"
s3=s1+s2
s4=s1*3
print(s3,' ',s4)
if s3>s4:
    print(s3)
else:
    print(s4)
```

Результат работы программы представлен на рисунке 97.

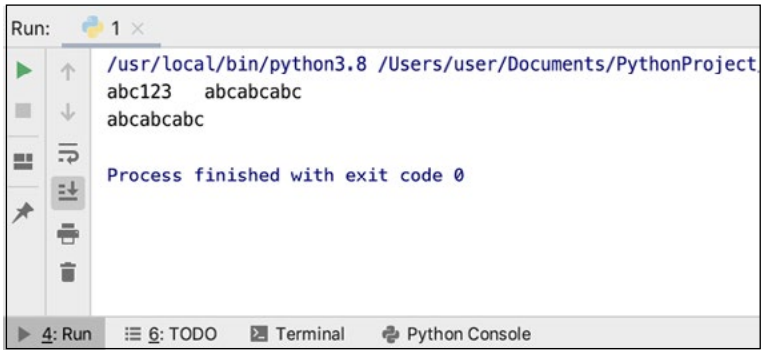


Рис. 97. Результат работы программы

Также в языке Python используется достаточное число встроенных функций для обработки строк. Опишем некоторые из них в таблице 18.

Таблица 18

Основные функции (методы) для обработки строк

Функция	Описание
ord(x)	Возвращает код символа x в ASCII
chr(n)	Возвращает символ, код которого в ASCII равен n

Окончание

Функция	Описание
`len()`	Возвращает длину строки
`isnumeric()`	Возвращает True, если строка представляет собой число
`lower()`	Переводит строку в нижний регистр
`upper()`	Переводит строку в верхний регистр
`find(str[,start[,end]])`	Возвращает индекс подстроки в строке. Если подстрока не найдена, возвращается число −1
`replace(old,new[,num])`	Заменяет в строке одну подстроку (`old`) на другую (`new`). `num` ограничивает количество замен
`split([d[,num]])`	Разбивает строку на подстроки в зависимости от разделителя `d`. `num` определяет максимальное количество частей для разбиения
`join(strs)`	Объединяет строки в одну строку, вставляя между ними определённый разделитель `strs`

(Квадратные скобки в синтаксисе функций означают необязательные элементы.)
Полный список методов строк можно увидеть, если вызвать функцию dir( ).

Пример 4

Заменить в строке все символы «o» на «a».

```
s1="good morning"
s2=s1.replace('o','a')
print(s2)
```

Результат работы программы представлен на рисунке 98.

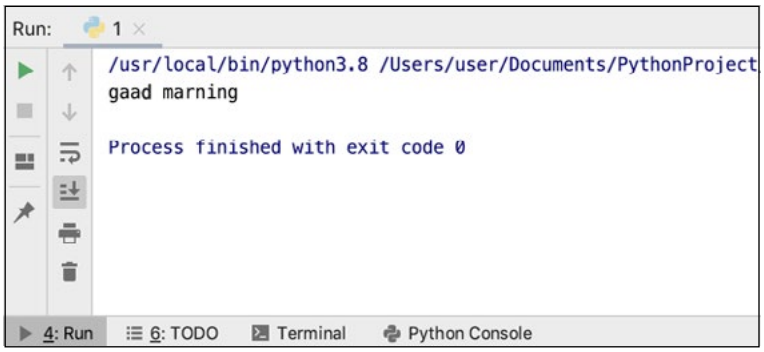


Рис. 98. Результат работы программы

В данном примере используется встроенная функция `replace`, которая заменяет все требуемые по условию символы.

Пример 5

В строке заменить пробелы знаком тире «—». Если встречается подряд несколько пробелов, то их следует заменить одним знаком «—», пробелы в начале и конце строки удалить.

```
s=input("Введите строку: ")
i=0
while s[i]==' ':
    i+=1
s=s[i:]
i=len(s)
while s[i-1]==' ':
    i-=1
s=s[:i]
sn=s[0]
i=1
while i<len(s):
    if s[i]!=' ':
        sn+=s[i]
    elif s[i-1]!=' ':
        sn+='-'
    i+=1
print(sn)
```

Результат работы программы представлен на рисунке 99.

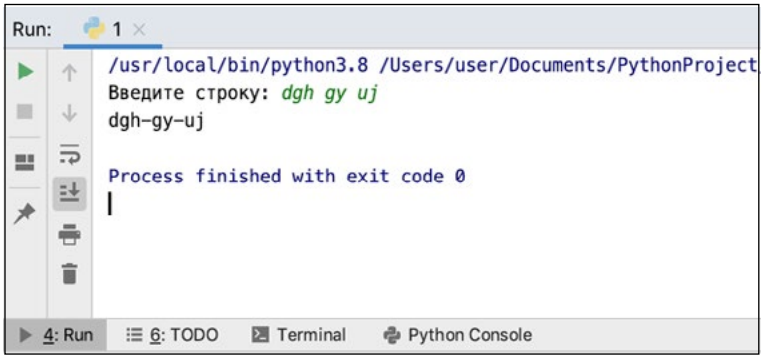


Рис. 99. Результат работы программы

В данном примере используется встроенная функция `len`, которая возвращает длину строки, также используются индексы для обращения к отдельному символу строки. Здесь не рекомендуется использовать функцию `replace`, так как она заменит все пробелы на тире, что противоречит условию задачи.

Пример 6

Определить, является ли введённая строка палиндромом, т. е. читается ли одинаково в прямом и обратном направлении (например, КАЗАК).

```
s=input("Введите строку: ")
sp=''
for i in range(len(s)-1,-1,-1):
    sp=sp+s[i]
if s==sp:
    print("палиндром")
else:
    print("не палиндром")
```

Результат работы программы представлен на рисунке 100.

Рис. 100. Результат работы программы

В данном примере получаем строку `sp`, обратную к строке `s` в цикле `for` с уменьшением шага. Затем две строки сравниваются, если они равны, то строка является палиндромом.

Можно предложить и другой вариант, в котором получаем обратную строку с использованием среза.

```
s=input("Введите строку: ")
sp=s[::-1]
if s==sp:
    print("палиндром")
else:
    print("не палиндром")
```

Результат работы программы представлен на рисунке 101.

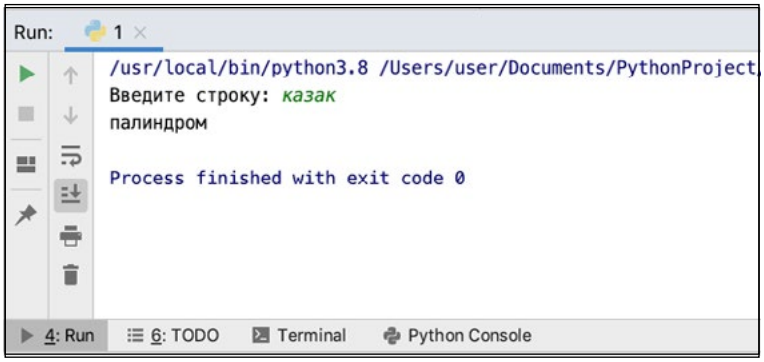


Рис. 101. Результат работы программы

Эту задачу можно ещё усложнить, если предположить, что в строке могут встречаться пробелы. Обычно в таких фразах пробелы не учитываются при проверке. Например, в знаменитой фразе «А роза упала на лапу Азора» пробелы игнорируются. Предварительно перед проверкой пробелы удаляются с помощью функции `replace`.

```
s=input("Введите строку: ")
if ' ' in s:
    s1=s.replace(' ','')
sp=s1[::-1]
```

```
if s1==sp:
    print("палиндром")
else:
    print("не палиндром")
```

Результат работы программы представлен на рисунке 102.

Рис. 102. Результат работы программы

Практическая часть

Цель работы: ознакомление с понятием «строка» языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Заменить в строке все символы «а» на «mama».
3. Для каждого символа, введённого с клавиатуры, указать, сколько раз он встречается в строке. Сообщение об одном символе должно выводиться не более одного раза.

Указание. Примерный вид программы:

```
s=input("Введите строку: ")
i=0
l=list(s)
i=0
while i<len(l):
    k=l.count(l[i])
    print(l[i],k)
    t=l[i]
    while t in l:
        l.remove(t)
```

4. Отредактировать предложение, удаляя из него лишние пробелы, оставляя только по одному пробелу между словами.
5. Проверить, можно ли из букв слова *X* составить слово *Y*, используя каждую букву слова *X* не более одного раза.
6. Подсчитать количество гласных букв в строке.

Указание. Примерный вид программы:

```
s=input("Введите строку: ")
k=0
gl='аоеиуяыёэю'
for x in s:
    if x in gl:
        k+=1
print("кол-во гласных ",k)
```

7. Найти процентное содержание цифр в исходном тексте.

8. Найти сумму чисел, встречающихся в строке.

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе со строками в языке программирования Python.

Контрольные вопросы

1. Дайте определение строки в языке программирования Python. Строки должны заключаться в кавычки или апострофы?

2. Можно ли изменять строку в языке программирования Python? Можно ли изменять отдельный символ строки?

3. Какие основные встроенные функции для работы со строками в языке Python вы можете назвать?

Лабораторная работа 6.2. Работа со строками в языке Python

Теоретическая часть

Воспользоваться материалами из лабораторной работы 6.1.

Практическая часть

Цель работы: ознакомление с понятием «строка» языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.

2. Для строки и указанного символа определить номер первого и последнего вхождения этого символа в строку.

3. Заменить в самом длинном слове строки буквы «a» на «b».

4. Удалить из строки «лишние» пробелы, т. е. оставить только один пробел в последовательности пробелов.

5. Удалить из строки все символы, заключённые в скобки. Например, для строки `"abcd(123)efg"` удалить подстроку `"123"` и получить в результате `"adcdefg"`.

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе со строками в языке программирования Python.

Контрольные вопросы

1. Какие основные функции для работы со строками в языке Python вы использовали при решении задач лабораторной работы?

2. Какие способы задания строк вы использовали в лабораторной работе?

3. Возможен ли следующий фрагмент программы на языке Python?

```
s='pqr'
s1='123'
s+=s1
```

Лабораторная работа 7.1. Работа с функциями в Python

Теоретическая часть

С функциями в языке Python мы встречаемся постоянно, например при работе со строками. Дадим определение функции: это именованный фрагмент программного кода, к которому можно обратиться из другого места вашей программы (но есть lambda-функции, у которых нет имени). Часто программист создаёт функции для работы с данными,

которые передаются ей в качестве аргументов, также функция может формировать некоторое возвращаемое значение.

Справочник

Функции в программировании используются для следующих целей: многократного использования в программе одного и того же фрагмента кода; разделения сложной программы на составные части — процедурной декомпозиции. Функции в Python используются для реализации вспомогательного алгоритма.

Рассмотрим создание функции. В общем виде функция имеет следующий формат:

```
def <имя_функции>(par1,par2,...,parN):
    операторы
```

Описание функции начинается со строки заголовка, в которой оператор def определяет имя функции, с которым будет связан объект функции, далее следует список параметров, который может быть пустым либо состоять из некоторого числа параметров в круглых скобках. Имена параметров в строке заголовка будут связаны с аргументами, передаваемыми в функцию в точке вызова. Возврат значения функцией осуществляется с помощью ключевого слова `return`, после которого указывается возвращаемое значение. Слово `return` может располагаться в любом месте в теле функции, этот оператор завершает работу функции и передает результат вызывающей программе.

Оператор `return` выглядит следующим образом:

```
return <выражение>
```

Оператор return является необязательным — если он отсутствует, работа функции завершается, когда достигается конец тела функции.

Вызов функции может осуществляться следующим образом:

```
<имя_функции>(arg1,arg2,...,argN)
```

В функцию передаются конкретные аргументы — записываются вместо параметров.

Вызов функции может также быть включён в состав выражения.

Количество аргументов и параметров при вызове функции должно совпадать (но можно запрограммировать переменное количество принимаемых аргументов). В качестве аргументов могут выступать как конкретные значения, так и переменные.

Таким образом, общая структура работы с функцией выглядит, как показано на рисунке 103.

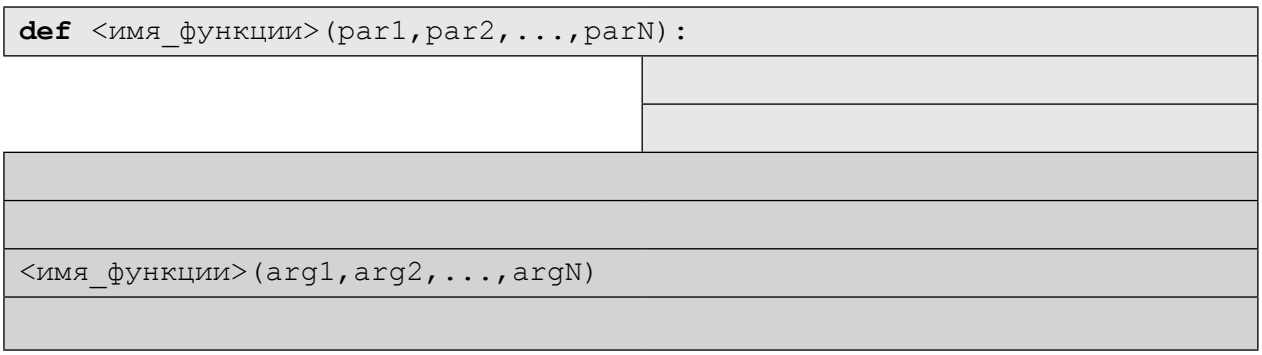


Рис. 103. Схема работы с функцией

Приведём примеры описания и вызова простых функций.

Пример 1

```
def summ1(x,y):
    return x+2*y
x=int(input())
y=int(input())
print(summ1(x,y))
```

В этом примере описана функция с двумя параметрами, которая вычисляет выражение $x + 2 \cdot y$. Далее в основной программе с клавиатуры вводятся два целых числа, на экран выводится значение функции `summ1`, аргументами которой выступают введённые числа.

Результат работы программы приведен на рисунке 104.

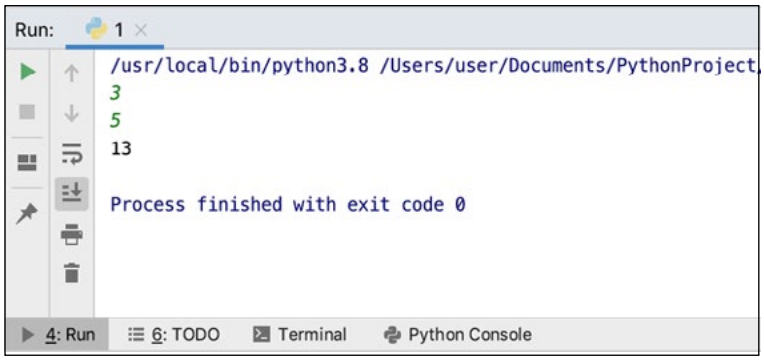


Рис. 104. Результат работы программы

Пример 2

```
def mmm(a,b,c):
    if (a>b) and (a>c):
        return a
    elif (b>c) and (b>a):
        return b
    elif (c>a) and (c>b):
        return c
print(mmm(3,6,1))
```

В этом примере описана функция mmm, которая возвращает наибольшее из трёх чисел. В основной программе осуществляется вывод на экран значения функции от аргументов 3, 6, 1.

Результат работы программы представлен на рисунке 105.

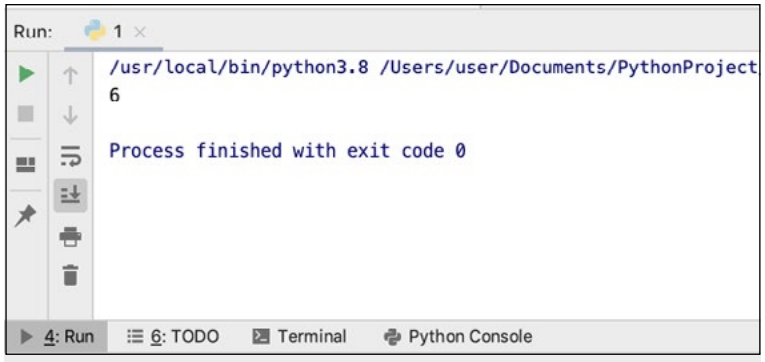


Рис. 105. Результат работы программы

Практическая часть

Цель работы: ознакомление с понятием «функция» языка программирования Python, описанием и вызовом функции.

Ход работы

1. Открыть среду разработки PyCharm.
2. Написать функцию, которая возвращает процент от числа.

Указание. Примерное описание функции:

```
def procent(x,n):
    return x/100*n
```

Данная функция принимает два аргумента: само число и требуемый процент. Возвращает процент, рассчитанный по формуле.

Примерный вид программы:

```
def procent(x,n):
    return x/100*n
x=int(input())
n=int(input())
print(procent(x,n))
```

3. Написать функцию, которая вычисляет наибольший общий делитель двух целых чисел.

Указание

Для разработки функции можно использовать алгоритм Евклида, который представляет собой эффективный алгоритм нахождения наибольшего общего делителя. Данный алгоритм получил своё название в честь известного греческого математика Евклида. Алгоритм был описан автором в VII и X книгах «Начал». Это один из старейших численных алгоритмов, используемых в наше время. Данный алгоритм представлен на рисунке 106.

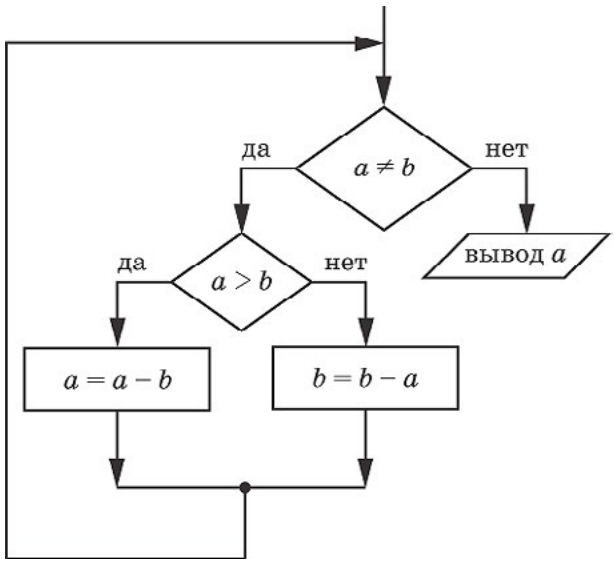


Рис. 106. Блок-схема алгоритма Евклида

Примерный вид описания функции:

```
def nod(a,b):
    while a!=b:
        if a>b:
            a-=b
        else:
            b-=a
return a
```

Данная функция принимает два аргумента — числа, для которых необходимо вычислить наибольший общий делитель. Возвращает функция одно значение — найденный делитель.

Примерный вид программы:

```
def nod(a,b):
    while a!=b:
        if a>b:
            a-=b
        else:
            b-=a
    return a
x=int(input())
y=int(input())
print(nod(x,y))
```

4. Написать функцию, которая проверяет, является ли введённое число простым.

5. Написать функцию, которая выводит только гласные английские буквы введённой строки.

Указание. Примерный вид программы:

```
def glas(a):
    k=0
    while k<len(a):
        if a[k] in 'aeiouyAEIOUY':
            print(a[k])
        k+=1

a=str(input())
glas(a)
```

6. Написать функцию, которая для любого целого аргумента выводит количество цифр в нём.

7. Написать функцию нахождения длины отрезка по координатам его концов.

8. Написать функцию, определяющую, является ли заданное число палиндромом (например, число 12721 — палиндром, одинаково читается слева направо и справа налево).

Выводы

В ходе выполнения лабораторной работы вы получили представление о разработке функций в языке программирования Python.

Контрольные вопросы

1. Для чего используются функции в программировании?
2. Как выглядит описание функции в языке Python?
3. Для чего используется оператор return при описании функции в языке Python?

Лабораторная работа 7.2. Работа с функциями в Python

Теоретическая часть

Воспользоваться материалами из лабораторной работы 7.1.

Практическая часть

Цель работы: ознакомление с понятием «функция» языка программирования Python, описанием и вызовом функции.

Ход работы

1. Открыть среду разработки PyCharm.
2. Написать функцию нахождения длины отрезка по координатам его концов. Используя эту функцию, решить следующую задачу: даны координаты трёх вершин треугольника. Найти длины всех его сторон.
3. Написать функцию, которая сравнивает введённые числа и результат выдаёт в виде знаков «>», «<» или «=».
4. Написать функцию, которая по введённому *n* выводит на экран *n*-е число Фибоначчи.

Выводы

В ходе выполнения лабораторной работы вы получили представление о разработке функций в языке программирования Python.

Контрольные вопросы

1. Какой вид имеет описание функции в языке Python?
2. Какие основные операторы языка Python вы использовали при решении задач лабораторной работы?
3. Можно ли описать функцию в языке Python, не используя оператор `return`?

Лабораторная работа 8. Кортежи в языке Python

Теоретическая часть

Важно!

Кортеж можно определить как неизменяемый список. Таким образом, в отличие от работы со списком, нельзя удалять элементы из кортежа или добавлять элементы в кортеж.
Кортежи часто используются для представления неизменяемой последовательности разнородных объектов. Кортежи неизменяемы, значит, хорошо защищены от изменений, как намеренных, так и случайных. По размеру кортежи меньше, чем списки.

Кортежи создаются способом, похожим на создание списка: перечислением элементов в круглых скобках. Примеры:

```
s=(2,6,9,10)
s1=("ab","bc","cd")
```

Пустые круглые скобки, так же как и в случае списка, обозначают пустой кортеж: `s4=()`.

Второй способ представляет собой перечисление элементов через запятую:

```
s3 ="a","b","c","d"
```

Третий способ — использование функции `tuple` при помощи генераторов:

```
a=tuple(i for i in range(0,20))
```

Можно переделать список в кортеж, также используя функцию `tuple`, например:

```
L=["one","two","three"]
Lp=tuple(L)
```

Возможно и обратное преобразование — кортежа в список или строку — с помощью функций `list` и `join`.

Пример 1

```
cor1=(1,2,3)
l1=list(cor1)
print(l1)
```

Результат работы программы представлен на рисунке 107.

Рис 107. Результат работы программы

Пример 2

```
cor2=('a','b','c')
s1=''.join(cor2)
print(s1)
```

Результат работы программы представлен на рисунке 108.

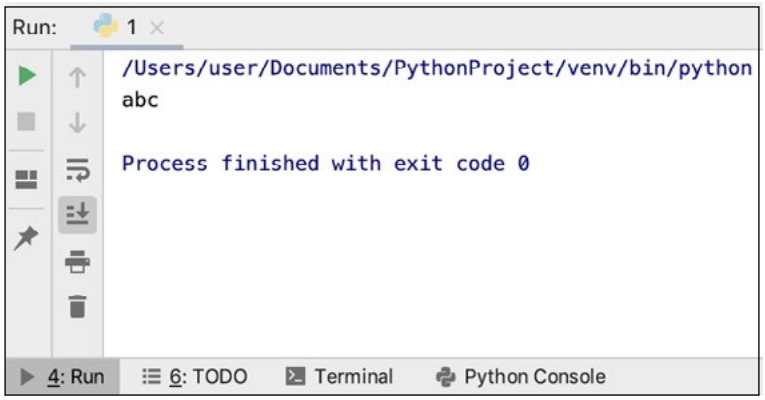


Рис. 108. Результат работы программы

Индексация элементов кортежа не отличается от индексации строк и списков: нумерация начинается с нуля, возможно использование отрицательных индексов (табл. 19).

Таблица 19

Варианты индексации кортежей в Python

Кортеж C	**one**	**two**	**three**	**four**	**five**	**six**
Индекс	0	1	2	3	4	5
Отрицательный индекс	−6	−5	−4	−3	−2	−1

Способ получения элементов кортежа также схож с получением элементов списка: в квадратных скобках указывается индекс элемента, который необходимо получить.

```
cor=("day","month","year")
cor[2]
```

Результат работы программы представлен на рисунке 109.

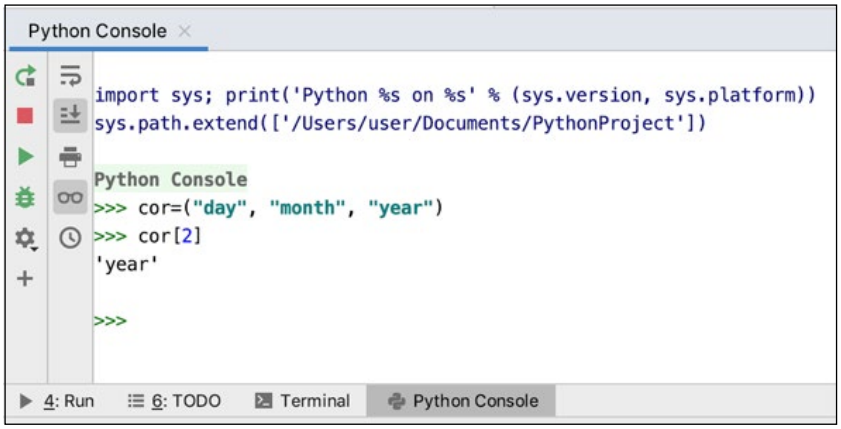


Рис. 109. Результат работы программы

Как было сказано ранее, в языке Python кортежи являются неизменяемыми (аналогично строкам). То есть нельзя присвоить новое значение какому-то отдельному элементу кортежа или, например, применить методы `append()` или `extend()`.

Такое обращение вызовет ошибку:

```
cor=("day","month","year")
cor[0]="night"
```

Но мы можем создать новый кортеж из имеющихся.

Пример 3

```
cor=("day","month","year")
cor1=cor+cor
print(cor1)
```

Результат работы программы представлен на рисунке 110.

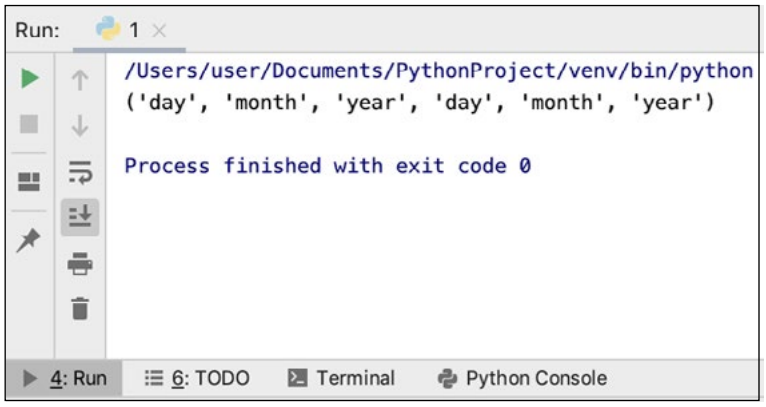


Рис. 110. Результат работы программы

Опишем основные операции над кортежами, допустимые в языке Python (табл. 20).

Таблица 20

Операции над кортежами

Оператор	Описание	Пример

`+`	Сложение (конкатенация)	`c1=(1,2,3)`
		`c2=(2,3,4)`
		`c3=c1+c2`

`*`	Умножение кортежа на число. Оператор создаёт несколько копий кортежа, формат оператора: `c*n` или `n*c`, где `c` — это строка, а `n` — натуральное число	`c1=(1,2,3)` `c2=cor1*3`

`in`	Оператор принадлежности, который возвращает `True`, если элемент входит в строку, и `False`, если нет	`nams=(«Irina","Marina")`
		`s="Irina"`
		`if s in nams:`
		`  print("В вашем классе есть ",s)`

Для перебора элементов кортежа можно использовать такие стандартные циклы, как `for` и `while`.

Пример 4

```
nams=("Irina", "Marina", "Olga")
for s in nams:
    print("В вашем классе есть ",s)
```

Результат работы программы представлен на рис. 111.

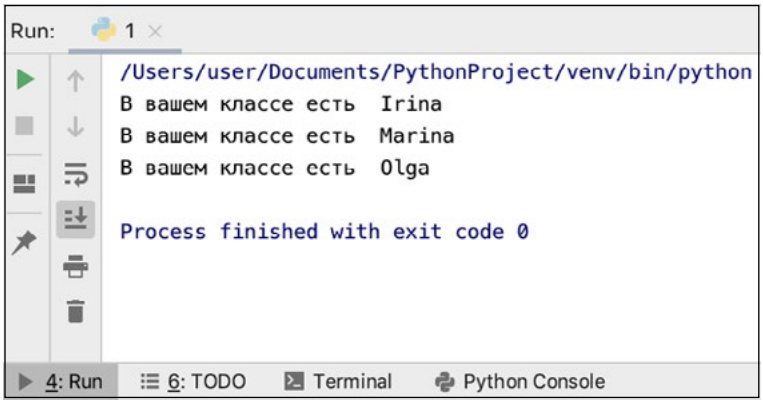


Рис. 111. Результат работы программы

Кортежи могут быть сложного вида, т. е. в своём составе содержать другие кортежи в качестве элементов. Например:

```
cor=(1,2,(3, 4),(5, 6, 7))
```

В состав данного кортежа cor входят ещё два кортежа. Для доступа к элементам такого сложного кортежа используются вложенные индексы.

Пример 5

```
cor=(1,2,(3, 4),(5, 6, 7))
print(cor[2])
print(cor[2][0])
```

Результат работы программы представлен на рисунке 112.

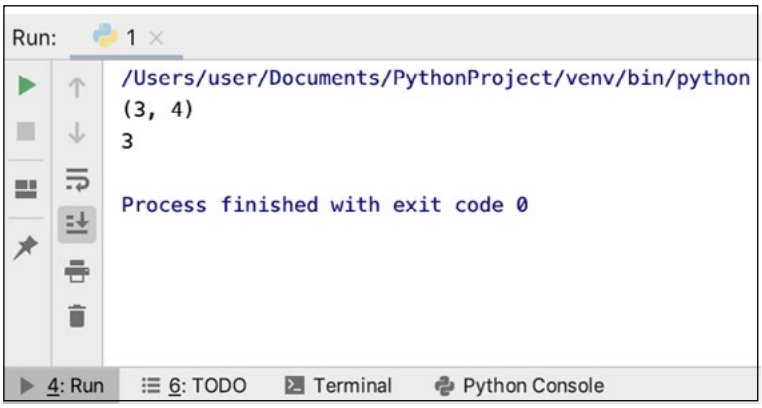


Рис. 112. Результат работы программы

Опишем основные функции языка Python для работы с кортежами (табл. 21). По сути они совпадают с ранее описанными функциями для работы со списками.

Таблица 21

Основные функции (методы) обработки кортежей

Функция	Описание
`len()`	Возвращает количество элементов кортежа
`min()`	Возвращает минимальный элемент кортежа
`max()`	Возвращает максимальный элемент кортежа
`zip()`	Возвращает кортеж элементов из последовательностей, переданных в качестве аргументов
`count(x)`	Возвращает количество элементов со значением `x`, входящих в кортеж
`index()`	Возвращает индекс элемента, входящего в кортеж
`sorted()`	Возвращает список, элементы которого — отсортированные элементы кортежа

Как видно из таблицы, к кортежам нельзя применить функции, результат которых изменит кортеж, например `insert`.

Конечно, кортежи имеют определённое сходство со списками в языке программирования Python. Зачем же тогда нужны кортежи? Опишем их основные преимущества:

1) так как кортежи неизменяемы, то их использование обеспечивает сохранность данных от непреднамеренных изменений;

2) работа с кортежами в языке Python осуществляется быстрее, чем работа со списками;

3) кортежи занимают меньше места, чем списки, — осуществляется экономия памяти;

4) в некоторых ситуациях синтаксис языка Python позволяет использовать только кортежи.

Пример 6

Задан список, содержащий в своём составе кортежи. Найти все кортежи, которые содержат элемент, равный 5.

```
c=[(14,5),(23,41),(5,62),(1,3),(3,3)]
new_cor=tuple(item for item in c if item[0]==5 or item[1]==5)
print(new_cor)
```

Результат работы программы представлен на рисунке 113.

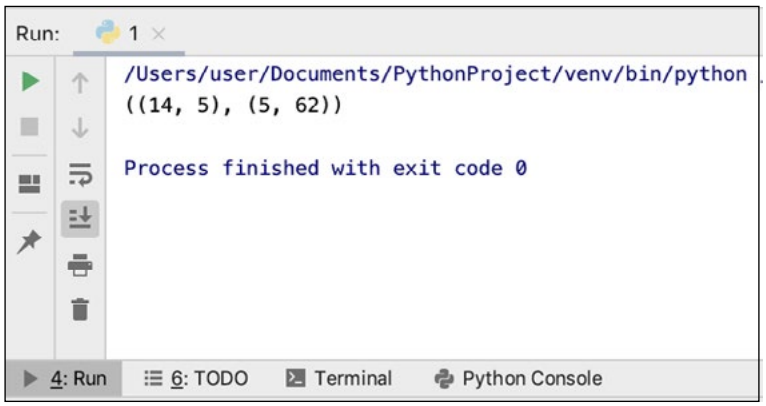


Рис. 113. Результат работы программы

Практическая часть

Цель работы: ознакомление с понятием «кортеж» языка программирования Python.

Ход работы

1. Открыть среду разработки PyCharm.
2. Заполнить кортеж десятью случайными целыми числами от 0 до 3 включительно. Подсчитать количество нулей в получившемся кортеже.

Указание. Примерный вид программы:

```
from random import randint
tup1=tuple(randint(0,3) for i in range (10))
print(tup1)
k=tup1.count(0)
print("количество 0",k)
```

3. Написать функцию, которая задаёт кортеж, заполненный случайными числами из указанного промежутка. Длина кортежа также передаётся в качестве аргумента функции.

Указание. Примерный вид функции и программы:

```
from random import randint
def fill_tup(a,b,n):
    return tuple(randint(a,b) for i in range (n))
k=fill_tup(1,6,5)
print("новый кортеж",k)
```

4. Получить кортеж из элементов списка, входящих в него только один раз.

Указание. Примерный вид программы:

```
l=[3,7,1,9,4,2,6,8,2]
new_l=[]
for x in l:
    if l.count(x)==1:
        new_l+=[x]
new_t=tuple(new_l)
print(new_t)
```

Выводы

В ходе выполнения лабораторной работы вы получили представление о работе с кортежами в языке программирования Python.

Контрольные вопросы

1. Дайте определения кортежа в языке Python.
2. Чем отличаются кортежи от списков, строк в языке Python?
3. Какие функции по работе с кортежами вы можете назвать?

Примеры конспектов уроков

Тема урока «Циклы в языке Python»

Тип урока: открытие нового знания.

Цель урока: ознакомление учащихся с понятием списка в языке Python, особенностями списков. Основные приёмы работы со списками.

Планируемые результаты:

предметные: получение информации о работе цикла с параметром в языке Python, получение основных навыков построения циклических алгоритмов;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

I. Этап постановки цели и задач урока, мотивации к учебной деятельности — 5 мин.

Деятельность учителя: предлагает рассмотреть сегодняшнюю тему «Циклы в языке Python». Вопросы: что вы понимаете под циклами? Часто ли циклы встречаются в нашей жизни?

Деятельность учеников: дают определение цикла, приводят примеры циклов.

II. Этап актуализации знаний и пробного учебного действия — 8 мин.

Деятельность учителя: предлагает учащимся решить следующие задачи.

Задача 1. Найти сумму трёх чисел, введённых с клавиатуры.

Решение

```
s=0
x=int(input())
y=int(input())
z=int(input())
s=x+y+z
print("сумма=",s)
```

Задача 2. Найти сумму четырёх чисел, введённых с клавиатуры.

Решение

Для решения второй задачи учащиеся предлагают ввести ещё одну переменную. В итоге решение принимает следующий вид:

```
s=0
x=int(input())
y=int(input())
z=int(input())
w=int(input())
s=x+y+z+w
print("сумма=",s)
```

Задача 3. Найти сумму пяти чисел, введённых с клавиатуры.

Для решения этой задачи также можно предложить добавить ещё одну переменную.

Тогда учитель задаёт вопрос: а если необходимо найти сумму 100 чисел? Надо ли тогда использовать 100 переменных? Другой вопрос: а если неизвестно заранее количество чисел, которые необходимо просуммировать? При этом каждый раз для 100 переменных будет выполняться одна и та же группа действий — ввести переменную и суммировать её значение.

В этих случаях лучше, когда есть группа действий, которую можно выполнить многократно. Предлагается использовать циклы.

III. Этап изучения нового материала — 16 мин.

Деятельность учителя: объясняет новый материал.

В языке Python используются два вида цикла: цикл с предусловием и цикл с параметром.

Начнём с цикла с предусловием. Цикл с предусловием — один из самых часто используемых при составлении программ. Цикл данного вида продолжает выполнять блок операторов, пока условное выражение продолжает возвращать истину.

Блок-схема цикла с предусловием представлена на рисунке 114.

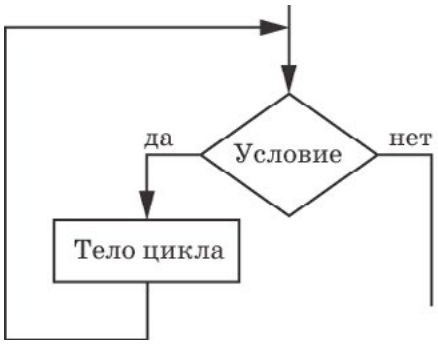


Рис. 114. Блох-схема цикла с предусловием

Формат оператора цикла с предусловием:

```
while <условие>:
    <тело цикла>
```

Как видно из блок-схемы, при выполнении цикла `while` сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передается на следующий оператор после тела цикла `while`. Если условие истинно, то выполняется тело цикла, после чего условие проверяется снова. Так продолжается, пока условие истинно. Как только условие станет ложным, работа цикла завершится и управление передастся следующему оператору после цикла.

Приведём примеры цикла с предусловием.

Пример 1

```
i=0
while i<15:
    print(i)
    i+=1
```

Пример 2

```
k=int(input("k="))
i=0
while i<k:
    print(i)
    i+=2
```

Ещё один вид цикла — цикл с параметром. Цикл с параметром выполняет указанный набор операторов заданное количество раз, которое определяется начальным и конечным значениями параметра цикла и шагом цикла. Блок-схема данного цикла представлена на рисунке 115.

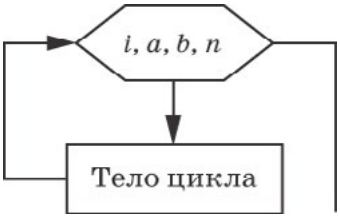


Рис. 115. Блох-схема цикла с параметром

На рисунке 115:
i — параметр (счётчик) цикла;
a — начальное значение параметра;
b —конечное значение параметра;
n — шаг цикла (приращение счётчика).
Формат оператора цикла с параметром:

```
for <параметр цикла> in <имя последовательности или функция
                         range()>:
    <тело цикла>
```

Одна из особенностей цикла `for` в Python состоит в том, что набор значений параметра цикла можно задать с помощью функции `range()`.

Прежде чем привести примеры данного цикла, разберём более подробно функцию `range()`. Данная функция может применяться с несколькими параметрами:

`range(A)` — создаётся последовательность чисел [`0, A-1`];

`range(A,B)` — создаётся последовательность чисел [`A, B`), это полуинтервал, `B` не входит в последовательность;

`range(A,B,N)` — создаётся последовательность [`A, B`) с шагом `N`, при этом шаг может быть отрицательным.

Приведём примеры работы цикла с параметром.

Пример 3

```
for i in range(5):
    print(i)
```

Пример 4

```
for i in range(2,16):
    print(i)
```

IV. Этап проверки понимания и первичного закрепления — 8 мин.

Деятельность учителя: предлагает учащимся для решения следующие задачи.

1. Вывести на экран последовательность чисел: 1, 2, 3, ..., *N*, где *N* вводится с клавиатуры.

2. Вывести на экран последовательность чисел 1^2, 2^2, 3^3, ... N^2.

В помощь для решения представленных выше задач учитель может предложить следующую «шпаргалку» (рис. 116).

`range(A)` — создаётся последовательность от 0 до *A* – 1, т. е. [0, *A* – 1]	`range(A,B)` — создаётся последовательность от *A* до *B*, *B* не входит, т. е. [*A*, *B*)	`range(A,B,N)` — создаётся последовательность от *A* до *B*, *B* не входит, т. е. [*A*, *B*), с шагом *N*, шаг может быть отрицательным
`range(6)`	`range(3,8)`	`range(3,8,2)`
0, 1, 2, 3, 4, 5	3, 4, 5, 6, 7	3, 5, 7

Рис. 116. «Шпаргалка» для функции `range`

Деятельность учеников: выполняют задания учителя.

V. Этап контроля усвоения и коррекции ошибок — 4 мин.

Деятельность учителя: предлагает учащимся ответить на следующие вопросы.

1. Какое вы можете дать определение цикла?
2. Какие виды цикла определены в языке Python?
3. Какой формат имеет цикл с параметром?
4. Какой формат имеет цикл с предусловием?

Деятельность учеников: отвечают на вопросы учителя.

VI. Информация о домашнем задании, инструктаж по его выполнению — 2 мин.

Деятельность учителя: предлагает учащимся решить дома следующую задачу, которая была предложена в начале урока: найти сумму 10 чисел, введённых с клавиатуры.

VII. Этап рефлексии деятельности на уроке — 2 мин.

Деятельность учителя: предлагает учащимся оценить свою деятельность на уроке. Для этого необходимо ответить на следующие вопросы.

1. Что нового я узнал сегодня на уроке?
2. Поможет ли мне эта информация решать задачи на языке Python?
3. Считаю ли я свою работу на уроке успешной?

Тема урока «Работа со списками в Python»

Тип урока: открытие нового знания.

Цель урока: ознакомление учащихся с понятием списка в языке Python, особенностями списков. Основные приёмы работы со списками.

Планируемые результаты:

предметные: получение информации о списках в языке Python, основных навыков обработки списков;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

I. Этап постановки цели и задач урока, мотивации к учебной деятельности — 5 мин.

Деятельность учителя: сообщает учащимся, что сегодня они познакомятся со структурой данных в языке Python. Если ранее на занятиях учащиеся работали с переменными, которые хранили только одиночные значения, то списки дадут нам возможность хранить наборы данных.

II. Этап актуализации знаний и пробного учебного действия — 8 мин.

Деятельность учителя: предлагает решить учащимся несколько задач.

Деятельность учеников: решают задачи, предложенные учителем.

Задача 1. Вывести на экран все чётные числа от 1 до 50.

Решение

```
for i in range (1,51):
    if i%2==0:
        print(i)
```

Задача 2. Найти сумму *n* чисел, введённых с клавиатуры.

Решение

```
n=int(input())
s=0
for i in range (1,n+1):
    k=int(input())
    s+=k
print("сумма=",s)
```

Задача 3. Даны *n* чисел, найти сумму чисел, меньших среднего арифметического.

Решение

При решении третьей задачи учащиеся сначала попробуют решить её аналогично второй задаче, но будут испытывать затруднения. Поэтому учитель предлагает перейти к новой теме, а затем попробовать решить третью задачу с использованием списков.

III. Этап изучения нового материала — 16 мин.

Деятельность учителя: объясняет новый материал.

Списки — это упорядоченная и изменяемая последовательность элементов. Элементы одного списка могут иметь одинаковый тип, но допускаются и смешанные списки. Элементы списка заключаются в квадратные скобки.

Учитель приводит примеры списков, просит учащихся указать тип элементов списка:

```
[31, 772, 903,-19, 62]
[55.2, 90.4, 87.12]
```

```
["abc "«, "cde", "efg"]
[12, 144, "xor", "and"]
```

Деятельность учеников: называют типы списков.

Учитель подчеркивает, что списки могут состоять из различных объектов: чисел, строк и даже других списков. В последнем случае списки называют вложенными. Пустой список не содержит элементов, обозначается так: `a=[]`.

Пример вложенного списка:

```
[[123],[2,4,6],[2,4,8]].
```

Далее рассматриваются способы задания списка:

1) в квадратных скобках через запятую перечислить его элементы:

```
a=[4,8,10]
```

2) использовать функцию `list`:

```
a=list(range(20));
```

3) использовать функцию функцию `randint` из модуля `random` для задания элементов списка случайным образом:

```
a=[randint(-10,10) for i in range(20)]
```

4) задать список с клавиатуры:

```
a=[input() for i in range(10)]
```

Также для формирования списка с клавиатуры можно предложить следующий фрагмент:

```
a=[]
n=int(input())
for i in range(n):
    k=int(input())
    a.append(k)
```

Учитель также отмечает, что каждый элемент списка имеет свой номер — индекс. С помощью индекса можно обратиться к каждому элементу. Нумерация элементов списка начинается с нуля.

Рассмотрим пример для списка `A`:

Индекс	0	1	2	3	4
Элемент	6	9	−1	2	0

Элемент с индексом 2 равен −1: `A[0]=-1`.

Учитель сообщает, что в Python также реализована отрицательная индексация. Отрицательная индексация начинается с конца списка:

Индекс	−5	−4	−3	−2	−1
Элемент	6	9	−1	2	0

Элемент с индексом −1 равен 0: `A[-1]=0`.

Иногда отрицательную индексацию удобнее использовать для получения последнего элемента в списке, потому что не нужно знать длину списка, чтобы получить доступ к последнему элементу.

Элементы списка можно выводить на экран. Это можно сделать или с помощью отдельного оператора `print`:

```
print(a)
```

или использовать следующий фрагмент:

```
a=[2,5.9,10,8]
for i in range(len(a)):
    print(a[i])
```

либо вывести элементы списка в одной строке:

```
a=[2,5.9,10,8]
for i in range(len(a)):
    print(a[i],end=" ")
```

Также учитель рассматривает внутреннее представление списков: хранятся указатели на объекты, а не сами объекты. Python размещает элементы списка в памяти, затем размещает указатели на эти элементы. Таким образом, список в Python — это массив указателей.

IV. Этап проверки понимания и первичного закрепления — 8 мин.

Деятельность учителя: предлагает учащимся для решения следующие задачи.

1. Задать список `m=[3,9,0,6,-2]`. Вывести на экран третий и последний элементы.
2. Задать список `a=["mother","father","sister","brother"]`. Вывести на экран элементы списка один за другим.
3. Задать список длиной 10 случайным образом. Вывести список на экран разными способами.

Деятельность учеников: выполняют задания учителя.

V. Этап контроля усвоения и коррекции ошибок — 4 мин.

Деятельность учителя: предлагает учащимся ответить на следующие вопросы.

1. Какое определение списка в языке Python вы можете дать?
2. Как можно задать список в языке Python?
3. Как вывести список на экран?
4. Как обратиться к элементу списка?
5. Какие способы индексации элементов списка в языке Python вы знаете?

Деятельность учеников: отвечают на вопросы учителя.

VI. Информация о домашнем задании, инструктаж по его выполнению — 2 мин.

Деятельность учителя: предлагает учащимся решить дома следующую задачу: задать список длиной 15 случайным образом. Вывести на экран сумму пятого и десятого элементов.

VII. Этап рефлексии деятельности на уроке — 2 мин.

Деятельность учителя: предлагает учащимся оценить свою деятельность на уроке. Для проведения данного этапа можно создать интерактивный ресурс, например в сервисе https://learningapps.org/ (рис. 117).

Рис. 117. Ресурс для проведения этапа рефлексии

Тема урока «Работа со строками в Python»

Тип урока: систематизация знаний.

Цель урока: систематизация и закрепление знаний учащихся о работе со строками в языке Python, особенностях строк. Закрепление основных приёмов работы со строками.

Планируемые результаты:

предметные: получение информации о строках в языке Python, закрепление основных навыков обработки строк;

метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные);

личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

I. Этап постановки цели и задач урока, мотивации к учебной деятельности — 5 мин.

Деятельность учителя: предлагает учащимся вспомнить работу на предыдущих уроках: с какими структурами данных велась работа?

Деятельность учеников: отвечают на вопросы учителя.

Далее учитель сообщает, что на сегодняшнем уроке будет продолжена работа со строками в языке Python.

II. Этап актуализации знаний и пробного учебного действия — 8 мин.

Деятельность учителя: предлагает учащимся ответить на вопросы следующего теста по теме «Строки в языке Python».

1. К элементу строки в языке Python можно обратиться

а) по индексу

б) с помощью оператора `print`
в) с помощью вывода на экран
г) только используя строку целиком

2. Функция `len(строка)` возвращает
а) первый элемент строки
б) длину строки
в) список из символов строки
г) пустую строку

3. Что выведет на экран следующий фрагмент программы?

```
s='123'
for k in s:
    print(s)
```

а)
```
1
2
3
```
б) `1 2 3`
в) `123 123 123`
г) `1 1 1`

III. Этап закрепления нового материала — 16 мин.

Деятельность учителя: предлагает учащимся решить несколько задач на языке Python. При решении этих задач потребуются знания, как работать с позиционными системами счисления. Для этого учащимся можно предоставить небольшую «шпаргалку» по данной теме (рис. 118).

Рис. 118. «Шпаргалка» по системам счисления

Список задач:

1) проверить, является ли введённая строка двоичным числом;
2) перевести введённое с клавиатуры двоичное число в десятичное;
3) перевести введённое пользователем десятичное число в шестнадцатеричное.

Примерное решение данных задач представлено ниже.

Задача 1

```
s=input("s=")
f=0
for i in s:
```

```
        if i=='0' or i=='1':
            f+=1
    if f==len(s):
        print("двоичное число")
    else:
        print("не двоичное число")
```

Задача 2

```
s=input("s=")
v=0
k=0
for i in range(len(s)-1,-1,-1):
    v+=int(s[i])*2**k
    #print(s[i])
    k+=1
print(v)
```

Задача 3

```
n=int(input())
s=''
while n>16:
    r=n%16
    n=n//16
    if r<10:
        buf=chr(r+48)
    else:
        buf=chr(r+55)
    s=buf+s
if n!=0:
    if n<10:
        buf=chr(n+48)
    else:
        buf=chr(n+48)
    s=buf+s
print(s)
```

Деятельность учеников: выполняют решение задач.

IV. Этап проверки понимания и первичного закрепления — 8 мин.

Деятельность учителя: предлагает учащимся решить несколько задач как вручную, так и с помощью разработанных программ.

1. Перевести число из двоичной системы счисления в десятичную:
а) 10101011;
б) 11001110;
в) 1010101.
Перевести число из десятичной системы счисления в шестнадцатеричную:
а) 388;
б) 290;
в) 1035.

Деятельность учеников: выполняют решение задач, сравнивают скорость решения, делают выводы.

V. Этап контроля усвоения и коррекции ошибок — 4 мин.

Деятельность учителя: предлагает учащимся ответить на следующие вопросы.

1. Какие функции для работы со строками в языке Python вы сегодня использовали?
2. Как следует выполнять перевод числа из двоичной системы счисления в десятичную?

VI. Информация о домашнем задании, инструктаж по его выполнению — 2 мин.

Деятельность учителя: сообщает домашнее задание: перевести число из десятичной системы счисления в n-ичную, где $2 \leq n \leq 9$.

VII. Этап рефлексии деятельности на уроке — 2 мин.

Деятельность учителя: предлагает учащимся оценить свою деятельность на уроке, ответив на следующие вопросы.

1. Что нового я узнал сегодня на уроке?
2. Считаю ли я свою сегодняшнюю работу на уроке успешной?
3. Знания из каких тем курса информатики я сегодня использовал при решении задач на уроке?

Форма аттестации, примеры контрольно-оценочных материалов

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных программ, фронтальных опросах, проводимых учителем.

Также в тематическом планировании предполагаются две промежуточные контрольные работы.

Контрольная работа для проверки полученных навыков по темам «Условный оператор if», «Циклы в языке Python»

1. Найти расстояние между двумя точками, заданными на плоскости их координатами.
2. Вычислить значение функции $y(x) = x^2 - 7x + 8$ для заданного с клавиатуры значения аргумента x.
3. Определить, сколько положительных среди трёх введённых с клавиатуры чисел.
4. Проверить, принадлежит ли точка с координатами (x, y) части фигуры, изображённой на рисунке 119.

Найти сумму нечётных делителей введенного с клавиатуры натурального числа.

Два числа называются дружественными, если каждое равно сумме делителей другого, исключая само это число. Найти все дружественные числа, не превосходящие k.

Найти все трёхзначные числа, которые при увеличении на 2 делятся на 3.

Найти все четырёхзначные числа, у которых сумма крайних цифр равна сумме средних (например, 3221).

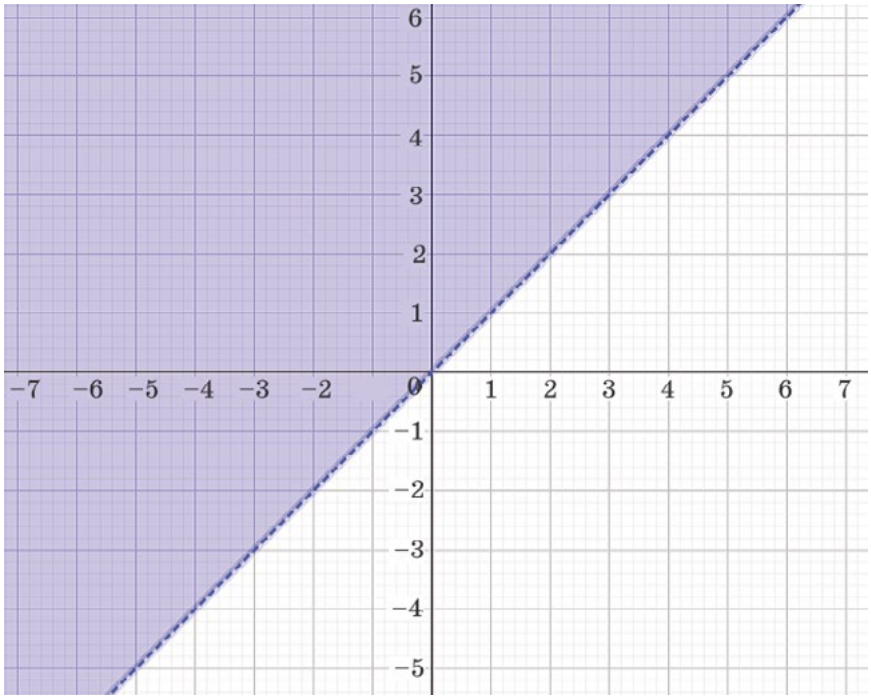


Рис. 119. Иллюстрация к задаче

Контрольная работа для проверки полученных навыков по темам «Списки в языке Python», «Работа со строками в языке Python»

1. В списке *X* из 50 элементов найти наименьший элемент и заменить его значением суммы всех элементов, предшествующих ему.
2. Даны два списка. Получить третий список, включая в него только те элементы, которые встречаются в исходных списках только один раз.
3. В списке *X* из 30 элементов найти наибольший элемент, после чего нормировать все элементы списка, разделив их значения на значение наибольшего элемента. Значение наибольшего элемента, его номер вывести на экран.
4. Отредактировать предложение, удаляя из него лишние пробелы, оставляя только по одному пробелу между словами
5. Дана строка, содержащая не менее 5 слов, за последним словом — точка. Вывести все слова последовательности, предварительно преобразовав каждое из них по следующему правилу: перенести первую букву в конец слова, затем, если слово нечётной длины, то удалить его среднюю букву.
6. Найти процентное содержание цифр в исходном тексте.
7. Для каждого символа введённого с клавиатуры слова указать, сколько раз он встречается в строке. Сообщение об одном символе должно выводиться не более одного раза.

Можно предложить следующие критерии оценивания контрольных заданий (табл. 22).

Таблица 22

Критерии оценивания контрольных заданий

Набранный балл	Оценка	Критерий
5	Высокий уровень	Получен полный и развёрнутый ответ на вопрос, приведены иллюстрирующие ответ примеры, получены ответы на дополнительные вопросы преподавателя
4	Средний уровень	Получен полный и развёрнутый ответ на вопрос, приведены иллюстрирующие ответ примеры, но не получены ответы на дополнительные вопросы преподавателя
3	Низкий уровень	Получен неполный ответ на вопрос, не приведены иллюстрирующие ответ примеры, получены неполные ответы на дополнительные вопросы преподавателя

Материалы для организации и проведения учебно-исследовательской и проектной деятельности школьников

Проект по программированию представляет собой проект, результатом которого является программа для решения той или иной задачи. Особенностью является то, что одна и та же задача в зависимости от уровня проработки может быть решена как начинающим, так и опытным программистом.

При выполнении проекта по программированию учащиеся имеют следующие возможности: выработать умение самостоятельно формулировать цели и задачи проекта, планировать свою деятельность, повысить уровень программирования на языке Python, получить умение представлять результаты своей деятельности.

Проект может разрабатываться индивидуально или группой учащихся. Если задача достаточно сложная, то проект может быть разбит на подзадачи, подпроекты. Каждую подзадачу будут выполнять различные группы участников проекта. Например, одна группа занимается разработкой алгоритма, другая группа — непосредственно написанием и отладкой кода на языке Python, третья — подготовкой к презентации проекта.

План работы над проектом по программированию может совпадать с этапами разработки программы (рис. 120).

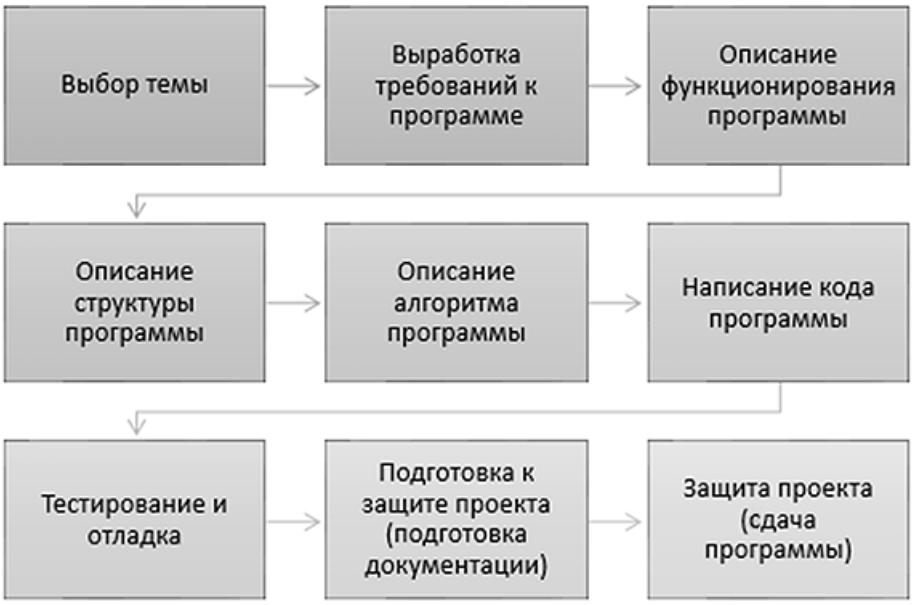


Рис. 120. Этапы проекта

В помощь участникам проекта можно предложить заполнить следующий учётный лист.

Проект по программированию

Тема проекта:
Творческое название (при наличии):
Основополагающий вопрос:
Авторы:
1.
2.
3.
...
Предметная область:
Краткая аннотация:
Этапы выполнения проекта:

При подготовке к защите проекта учащимся необходимо сделать презентацию и доклад, в котором отражаются основные этапы разработки программы, представлен алгоритм решения задачи, дан листинг программы, сформулированы основные результаты работы. Можно предложить в помощь учащимся заполнить следующий чек-лист.

1. Аннотация.
2. Содержание.
3. Постановка задачи:
   а) возможности использования программы;
   б) описание интерфейса.
4. Формализация алгоритма:
   а) перечень подпрограмм (при наличии);
   б) описание алгоритма (блок-схема или подробное словесное описание алгоритма).
5. Листинг программы (текст программы).
6. Тестовые примеры:
   а) результаты работы;
   б) скриншоты результатов работы.
7. Описание размещения.
8. Требования к программным и аппаратным средствам.

Для оценивания проекта могут быть разработаны специальные оценочные листы. В таблице 23 представлен пример оценочного листа.

Таблица 23

Лист оценивания проекта

Критерий оценивания	1-я группа	2-я группа	...
Актуальность темы			
Соответствие содержания проекта заявленной теме			
Техническая сложность разработанной программы			

Окончание

Критерий оценивания	1-я группа	2-я группа	...
Оригинальность алгоритма			
Дизайн интерфейса			
Степень разработанности программы			
Применение программы для решения аналогичных задач			
Итоговое количество баллов			

Темы проектов

Ниже приведены возможные темы исследовательских проектов учащихся.

1. Конвертор чисел (перевод числа в *n*-ичную систему счисления).
2. Шифровальщик текста (реализация шифра Цезаря).
3. Компьютерный тест.
4. Реализация игры «Камень, ножницы, бумага».
5. Калькулятор для ипотеки.

Ниже приведён пример проекта «Перевод числа в римскую систему счисления».

Для реализации данного проекта необходимо сначала рассмотреть способы представления чисел в римской системе счисления. Учащиеся могут представлять следующую информацию.

Как известно, римская система счисления является непозиционной. Для записи чисел используются буквы латинского алфавита.

В таблице 24 указано соответствие между арабскими числами и их римским представлением.

Таблица 24

Соответствие между арабским числами и римским представлением

Число	Римское представление
1	I
5	V
10	X
50	L
100	C
500	D
1000	M

Для записи числа используется следующее правило: каждый меньший знак, поставленный слева от большего, вычитается из него; каждый меньший знак, поставленный справа от большего, прибавляется к нему.

Примеры представления чисел в римской системе счисления:

IV, XXI, CCLX.

Далее учащиеся переходят к составлению алгоритма решения: последовательно вычитать из числа 1000, 500, 100 и т. д. Каждое вычитание добавляет к представлению числа новый символ.

Для получения решения в языке Python предлагается использовать кортеж:

```
[1000,900,500,400,100,90,50,40,10,9,5,4,1],
["M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"]
```

В первой строке будут храниться основные значения чисел в арабском представлении, а во второй строке — обозначения римских чисел.

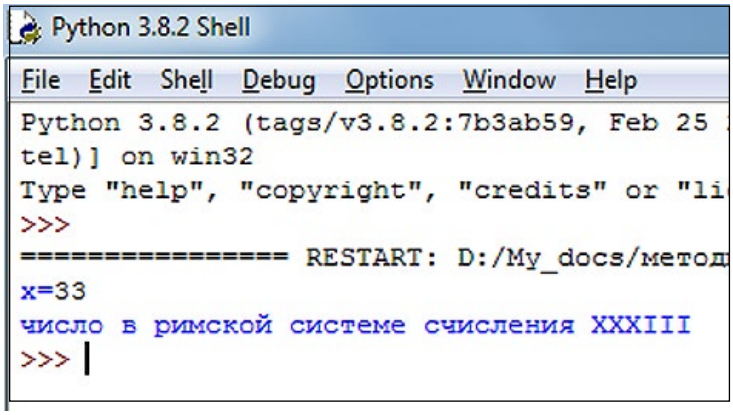
Программа на языке Python:

```
cif = zip(
[1000,900,500,400,100,90,50,40,10,9,5,4,1],
["M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"])
def ATR(num):
    if num<=0 or num>=4000 or int(num)!=num:
        raise ValueError('Input should be an integer between
                         1 and 3999')
    result=[]
    for d, rim in cif:
        while num>=d:
            result.append(rim)
            num-=d
    return ''.join(result)
x =int(input("x="))
print('число в римской системе счисления ',ATR(x))
```

`cif` представляет собой кортеж, сформированный с помощью функции `zip()`.

Затем описана функция `ATR`, которая и осуществляет перевод своего аргумента num в римское представление. Результат возвращается как строка.

Результат работы программы представлен на рисунке 121.



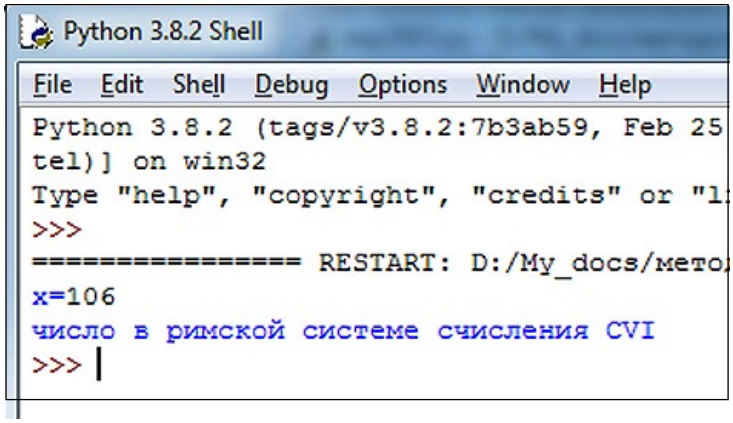


Рис. 121. Результат работы программы

Источники информации

Бэрри П. Изучаем программирование на Python. — М., 2017. — 624 с.

Буйначев С. К. Основы программирования на языке Python: учебное пособие. — Екатеринбург: Изд-во Урал. ун-та, 2014. — 91 с.

Бхаргава А. Грокаем алгоритмы: иллюстрированное пособие для программистов и любопытствующих. — СПб.: Питер, 2017. — 288 с.

Гэддис Т. Начинаем программировать на Python / пер. с англ. 4-е изд. — СПб.: БХВ-Петербург, 2019. — 768 с.

Мюллер Дж. Python для чайников. — СПб. : Диалектика, 2019. — 416 с.

Луридас П. Алгоритмы для начинающих: теория и практика для разработчика. — М. : Эксмо, 2018. — 608 с.

Лутц М. Изучаем Python, пер. с англ. 3-е изд. — СПб.: Символ Плюс, 2009. — 848 с.

Рафгарден Т. Совершенный алгоритм. Жадные алгоритмы и динамическое программирование. — СПб.: Питер, 2020. — 256 с.

Рейтц К., Шлюссер Т. Автостопом по Python. — СПб. : Питер, 2017. — 336 с.

Фёдоров Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для прикладного бакалавриата. — М. : Издательство Юрайт, 2019. — 161 с.

Python 3 для начинающих:

https://pythonworld.ru/samouchitel-python

Учебник по языку программирования Python (хабраиндекс):

https://habr.com/ru/post/61905/

Python/Учебник Python 3.1:

https://ru.wikibooks.org/wiki/Python/%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA_Python_3.1

Python для начинающих 2021 — уроки, задачи и тесты:

https://pythonru.com/uroki/python-dlja-nachinajushhih

Содержание

Григорьев Сергей Георгиевич
Родионов Михаил Алексеевич
Акимова Ирина Викторовна

Реализация дополнительной общеобразовательной программы по тематическому направлению «Программирование на языке Python» с использованием оборудования центра цифрового образования детей «IT-куб»

Методическое пособие

Под редакцией С. Г. Григорьева

Центр Естественно-научного и математического образования
Руководитель Центра *З. Г. Гапонюк*
Ответственный за выпуск *О. А. Полежаева*
Редактор *О. А. Полежаева*
Художественное оформление *Т. В. Глушкова*
Компьютерная вёрстка и техническое редактирование *И. Ю. Соколова*
Корректор *Е. В. Плеханова*

В содержание