Говорим с компьютером на одном языке

Долженкова Мария Львовна,
кандидат технических наук, заведующий кафедрой ЭВМ
Нижегородова Маргарита Владимировна,
кандидат педагогических наук, доцент кафедры САУ
Шмакова Наталья Александровна,
старший преподаватель кафедры САУ

Понятия «алгоритм» и «программа»

Любой человек постоянно встречается с множеством задач: от самых простых и хорошо известных до очень сложных. Для большинства из них существуют определенные правила (инструкции, предписания), объясняющие как решить данную задачу. Например, правила использования бытовой техники или приготовления любимого блюда. Чем более точно и однозначно определены правила, тем быстрее человек овладеет ими и будет эффективнее их применять. Такие правила принято называть алгоритмами.

> Алгоритм – конечная совокупность точно заданных правил или набор инструкций, описывающих порядок действий исполнителя для решения определенной задачи.

Наиболее простым в информатике считается линейный алгоритм.

Пример из математики. Представьте, что товарищ принес вам чертеж прямоугольника и просит сообщить ему площадь этой фигуры.

Какие действия вы должны выполнить, чтобы решить поставленную перед вами задачу? Вот ответ:

1) узнать (измерить) длину одной из сторон прямоугольника и запомнить (или записать) ее;
2) узнать (измерить) длину второй стороны фигуры и запомнить (или записать) ее;
3) рассчитать площадь, перемножив два найденных значения;
4) сообщить результат товарищу.

Алгоритм с разветвляющейся структурой предполагает наличие условия, при котором в случае его выполнения действия выполняются в одном порядке, а в случае невыполнения – в другом.

Пример из жизни. Мы пришли в магазин, выбрали товар и хотим расплатиться.

1) Спросить принимают ли оплату картой (условие);
2) если ответили, что принимают, то достали карту, иначе
3) (если нет) достали наличные;

4) оплатили покупку.

Алгоритм называется циклическим, если в нем имеются действия или наборы действий, которые необходимо выполнить более одного раза. Повторяющиеся алгоритмические действия являются телом цикла. Цикл имеет условие, по которому выполнение циклического алгоритма заканчивается.

Пример из литературы. Сказка Колобок.

1) Выпрыгнул Колобок из окна;
2) покатился Колобок по дорожке;
3) встретил по дороге Зверя;
4) спел Песенку;
5) колобка не съели? Вернуться к пункту 2, иначе
6) конец сказки.

Однако, одного набора инструкций для решения любой задачи недостаточно. Так зная рецепт самого вкусного торта, вы не сможете его приготовить, если под рукой нет всех требуемых ингредиентов. То есть необходимы данные, которые используются алгоритмом для получения результата.

> Программа – это описание на формальном языке, «понятном» компьютеру, последовательности действий, которые необходимо выполнить над данными для решения поставленной задачи.

А что такое – «язык программирования»?

Каждый день, отмечая понравившуюся публикацию в социальных сетях, отправляя письмо через электронную почту, устанавливая будильник на своем смартфоне, рассчитываясь на кассе в магазине, стоя на перекрестке со светофором, мы сталкиваемся с информационными технологиями и даже не задумываемся о том, что за кулисами работают специальные языки, с помощью которых удается объяснить электронному устройству, как выполнить ту или иную операцию. Речь идет о языках программирования.

Языки программирования, – по сути, инструмент «сторителлинга». У вас есть идея, «сюжет», и на языке программирования вы создаете структурированное «произведение», которое компьютер сможет прочитать, понять и выполнить. То есть «оживить» исходный код в виде игры, сайта, приложения или просто включить вашу любимую мелодию.

Со времени создания первых программируемых машин человечество придумало уже более десятка тысяч языков программирования. Каждый год их число пополняется новыми. Каждый язык программирования имеет свой алфавит, синтаксис и семантику.

> Алфавит языка – набор символов, используемых в данном языке, который создан для «общения» человека с компьютером.

Символы алфавита используются для построения базовых элементов программ – минимальных единиц языка, имеющих самостоятельный смысл. Базовые элементы называют лексемами. Лексемами русского языка являются слова русского языка, а знаки препинания и пробелы представляют собой разделители, не образующие лексем. В языках программирования лексическими единицами являются служебные слова, идентификаторы, константы, метки, знаки операций.

1) Служебные (зарезервированные) слова. Их смысл зафиксирован в языке, и поэтому служебные слова нельзя использовать в качестве имен, вводимых программистом.

2) Имена (идентификаторы). Они вводятся для обозначения в программе переменных, констант, типов, процедур и функций.

3) Числа и символьные строки.

4) Знаки операций и разделители. Они формируются из одного или нескольких специальных символов.

5) Комментарии. Комментарии не изменяют смысл программы, не влияют на ее выполнение и предназначены для пояснений.

Рисунок 1 – Язык программирования лишь инструмент, с помощью которого идея превращается в исполняемый на компьютере код

> Синтаксис языка – это набор правил, определяющий допустимые конструкции языка.
>
> Семантика языка – задает значения для всех допустимых цепочек языка, служит для определения смысла и однозначного толкования языковых конструкций.

Языки программирования – это формальные языки, предназначенные для записи алгоритмов, исполнителем которых будет компьютер.

Программа представляет собой последовательность команд, которые называются процессорными инструкциями. Во время запуска программы компьютер загружает ее машинный код в оперативную память и начинает

выполнять команду за командой. Машинная команда (код), состоящая из нулей и единиц, указывает, какое именно действие должен выполнить центральный процессор. Программа содержит тысячи последовательностей двоичных кодов соответствующих команд.

Машинный код можно рассматривать как самый низкоуровневый язык программирования. Создавать на нем программы можно и сейчас в ситуации, когда требуется экстремальная оптимизация, но делается это крайне редко в силу громоздкости кода и трудоемкости ручного режима управления ресурсами процессора.

Программы на языках программирования высокого уровня оформляются в привычном для человека виде: числа записываются как десятичные, а команды и другие служебные слова – на естественном (достаточно понятном) языке: PRINT, IF, BEGIN, WHILE, DO и тому подобное.

Однако при этом получается противоречие – человеку-программисту удобно разрабатывать и читать программу, но компьютер такую программу не «поймет»! Для решения данной проблемы служит транслятор.

Сейчас существует два основных способа трансляции – это компиляция программы и ее интерпретация. Когда работает компилятор: исходный код программы целиком переводится в машинный, создается исполняемый файл, соответствующий целевой архитектуре и операционной системе. При интерпретации – команды программы считываются строка за строкой, последовательно переводятся в машинный код и исполняются на текущей программно–аппаратной платформе. Есть еще один вид трансляции – динамическая JIT компиляция, когда исходная программа компилируется в промежуточный байт-код, который преобразуется в машинный – уже во время выполнения программы.

Знакомьтесь – Python

Первый прототип языка Python появился в 1989 году, он состоял из простой виртуальной машины, «парсера» и среды выполнения. В нем также присутствовал базовый синтаксис, операторы, словари, списки, строки и небольшое количество других типов данных. Главная особенность заключалась в том, что будущий Python предлагал гибкую модель расширяемости – то есть каждый программист мог самостоятельно добавить в систему нужные типы объектов. Удобная структура кода позволяла легко разрабатывать проекты и в дальнейшем поддерживать их. С помощью Python удобно производить сложные расчеты, он используется в машинном обучении, имеет библиотеки для работы с нейронными сетями, средства разработки мобильных и веб-приложений. Python мультипарадигмальный язык, поддерживает сразу три парадигмы программирования (структурную, функциональную и объектно-ориентированную). Интерпретируемость – еще одно из главных преимуществ Python, которое позволяет ему с одинаковой эффективностью работать под Linux, Windows и MacOS. Интерпретатор Python можно скачать с сайта python.org.

Python применяется во многих сферах: при создании сайтов и игр, десктопных и мобильных приложений, встречается в решениях, связанных с машинным обучением (искусственный интеллект, нейросети).

Относительно недавно в веб-разработке стали очень популярны Python-фреймворки: Django и Flask, Pyramid, Tornado, AioHTTP, FastAPI, CherryPy. Они облегчают процесс написания кода серверной части приложений на языке Python.

Мобильная разработка на этом языке не очень распространена. Для Android чаще используют Java, C#, C++ или Kotlin, а для iOS – Swift или Objective-C. На Python обычно программируют серверную часть приложения. Например, клиент Instagram для iOS написан на Objective-C, а сервер – на Python.

На Python полностью или частично написаны многие компьютерные игры. На этом языке пишут в основном скрипты взаимодействия персонажей, запуска сцен или обработки событий, хотя реализация пользовательского интерфейса, работа с графикой также возможны.

Язык Python – незаменимый помощник системных администраторов и сетевых инженеров в автоматизации управления серверами и дата-центрами. Постепенно он вытесняет из этой сферы Bash. Один из самых популярных фреймворков для автоматизации тестирования PyTest написан на Python. Он является идеальным инструментом для автоматизации рутинных задач в системном администрировании. На Python разрабатывают встраиваемые системы для различных устройств: им управляются банкоматы Сбера, он используется в Raspberry Pi (самый компактный компьютер – размером меньше банковской карты).

Язык применяется в средствах автоматического регулирования (температуры, давления и так далее), во встраиваемых системах станков с числовым программным управлением, в телекоммуникационном оборудовании. Есть специальная версия языка для программирования микроконтроллеров – MicroPython. Pico – последняя плата от Raspberry Pi – как раз использует его или MicroBit.

Python используют для создания плагинов и скриптов к готовым программам: для реализации игровой логики или создания дополнительных модулей. Скрипты на Python можно встраивать в программы на других языках, чтобы автоматизировать какие-то задачи.

Задачи по работе с данными востребованы во всех областях. Python – один из наиболее часто используемых в Data Science и работе с естественным языком, это простой и универсальный инструмент для решения любых аналитических задач. С помощью него обслуживают хранилища данных и облачные сервисы. Инженеры данных благодаря Python получили гибкий инструмент для обслуживания сложных конвейеров данных.

Чтобы в десятки раз ускорить создание отчетности и поиск информации или даже обрабатывать объем данных, который не помещается в Excel, достаточно пары команд в Python. С помощью этого языка аналитики данных, маркетологи, продакт-

менеджеры, геймпродюсеры без необходимости обращаться к разработчикам проводят анализ поведения пользователей, считают результативность рекламных кампаний, оптимизируют нагрузку на производство, находят «инсайты» в регулярных событиях на сайте или прогнозируют кеш-флоу своей организации, показатель отказа платящих пользователей.

Python – стандарт для создания и обучения искусственных нейронных сетей. С Python совместимы популярные библиотеки TensorFlow, PyTorch, CatBoost. С помощью него можно смоделировать любой объем данных для обучения модели или проведения хакатона, решать задачи распознавания изображений на фото и видео, создавать алгоритмы для машин и дронов-автопилотов, прогнозировать с использованием временных рядов и данных Минздрава заболеваемость в регионе, обучать неотличимых от человека чатботов, синтезировать музыку и голос, помогать банкам предотвращать мошенничество. А журналисты используют Python для дата-сторителлинга – создания интерактивных историй и расследований на открытых данных.

Как и где писать на Python

Команды для интерпретатора можно писать в обычном текстовом редакторе, но, чтобы сделать процесс создания программных продуктов на выбранном языке удобным, используют редакторы кода или IDE – интегрированные среды разработки – комплекс программных средств включающие транслятор, текстовый редактор (позволяющий копировать, удалять и перемещать фрагменты программ), справочную систему, средства отладки (поиска ошибок в программе) и другие сервисы.

IDLE – редактор, поставляемый вместе с Python. В нем есть оболочка – интерактивный интерпретатор. Его возможности обширны: автозавершение кода, подсветка синтаксиса, подбор отступа и базовый встроенный отладчик.

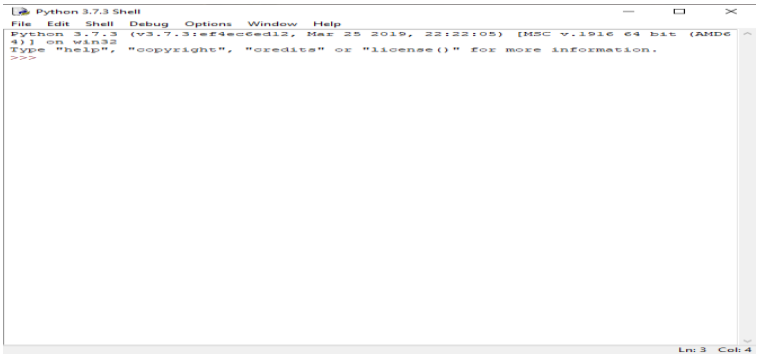

Рисунок 2 – Внешний вид IDLE

Плюсы: легкий, подходит для начинающих.

Минусы: не подходит для сложных проектов, не хватает продвинутых функций.

Sublime Text – свободное программное обеспечение. Редактор работает с несколькими языками программирования. Sublime Text тонко настраивается и дополняется пакетами для отладки, автозавершения кода и так далее.

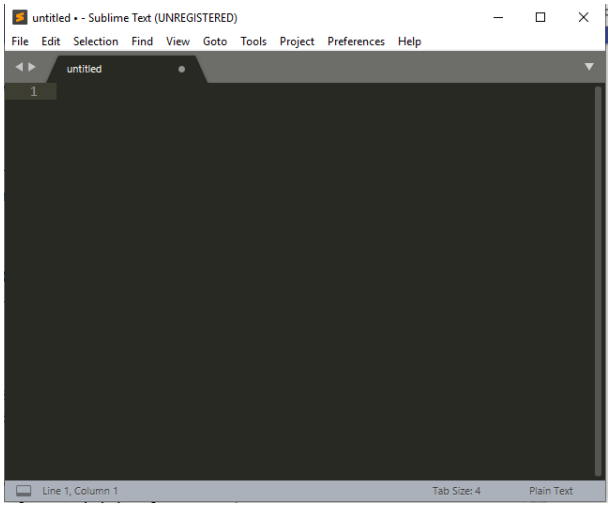

Рисунок 3 – Внешний вид Sublime Text

Плюсы: простой и по большей части бесплатный, тонко настраивается, компактный и эффективный.

Минусы: для удобства требует дополнительных пакетов.

Visual Studio Code – бесплатный редактор кода от Microsoft для Windows, Linux и MacOS. Его возможности – отладка, подсветка синтаксиса, интеллектуальное завершение кода, предопределенные фрагменты кода, рефакторинг и интеграция с Git. Поддерживаются различные языки программирования. Для начала работы с Python может понадобиться несколько дополнительных пакетов, но установить их довольно просто. Редактор постоянно обновляется.

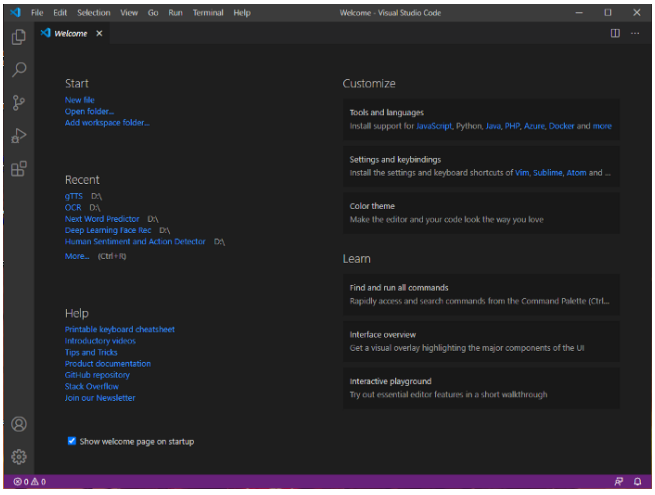

Рисунок 4 – Внешний вид Visual Studio Code

Плюсы: потребляет немного памяти по сравнению с другими громоздкими инструментами разработки, имеет встроенный терминал и прост в использовании.

Минусы: иногда терминал работает не так, как хотелось бы.

Jupyter Notebook – это веб-приложение с открытым исходным кодом, позволяющее создавать документы с выполняемым интерактивно кодом, уравнениями, визуализациями, простым текстом. Jupyter Notebook используется для очистки и преобразования данных, численного и статистического моделирования, визуализации данных, машинного обучение и многого другого. Этот редактор – хороший вариант для начала работы с наукой о данных и машинным обучением. Среда Jupyter Notebook широко используется во многих успешных компаниях.

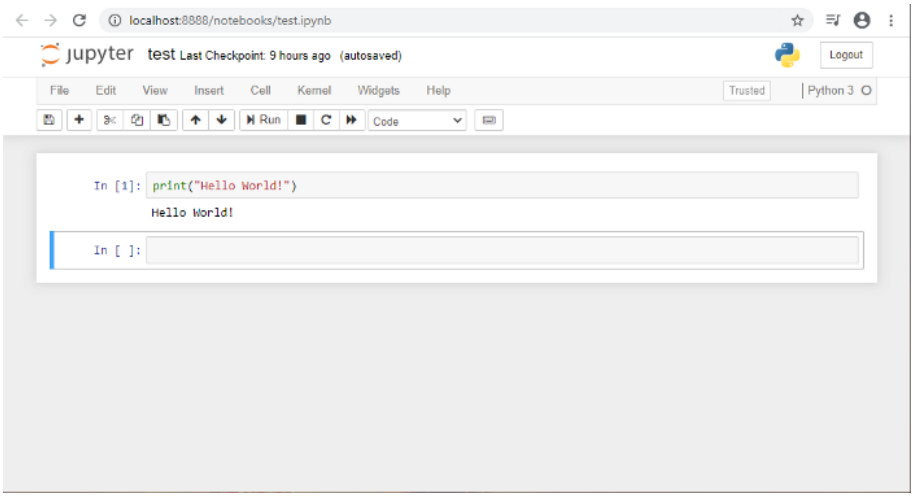

Рисунок 5 – Внешний вид Jupyter Notebook

Плюсы: лучшая платформа для начала работы с наукой о данных, легко делиться файлами и визуализациями, разметка и другие дополнительные функции.

Минусы: нет мощных функций из некоторых IDE.

PyCharm – это интегрированная среда разработки специально для Python. Разработана компанией JetBrains. Редактор разработан специально для Python, так что имеет широкий набор возможностей, таких как автозавершение и инспекции кода, подсветка ошибок, исправления, отладка, система контроля версий и рефакторинг. IDE доступна на Microsoft Windows, Linux и MacOS. Есть бесплатная и платная профессиональная версии.

Плюсы IDE: разработана специально для Python, поддерживает виртуальные среды Anaconda.

Минусы: требует минимум 8 Гб оперативной памяти для стабильной работы

Spyder – это мощная научная интегрированная среда программирования, написанная на Python, для Python. Она разработана учеными, инженерами и аналитиками данных для них самих. Spyder обладает уникальным сочетанием возможностей. Продвинутое редактирование, анализ, отладка и профилирование сочетается с возможностями исследования данных, интерактивного выполнения, глубокой инспекции кода и красивой визуализацией

Плюсы: хорошее рабочее окружение для просмотра интерпретаций и кодирования в два окна, широкий выбор опций исключительно для Python.

Минусы: устаревший интерфейс.

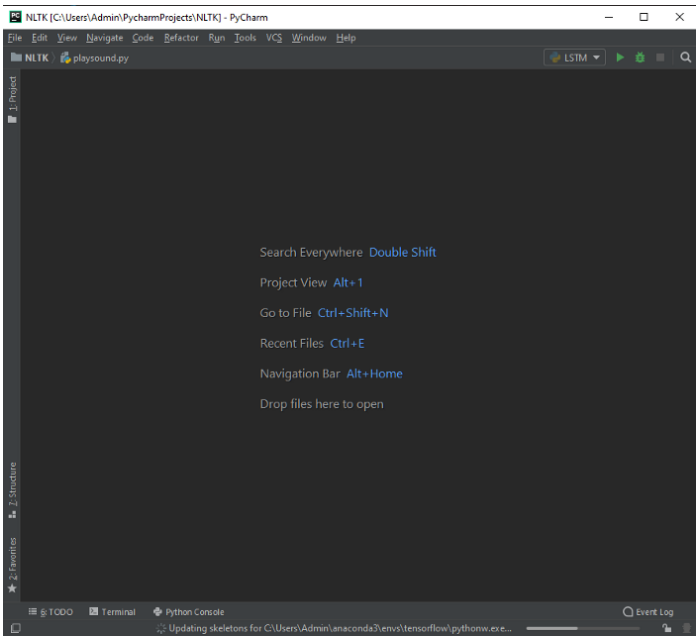

Рисунок 6 – Внешний вид PyCharm

В данном курсе мы будем использовать в качестве инструмента для создания программ PyCharm.

Начинаем разговаривать

Синтаксис языка Python, как и сам язык, очень прост.

Конец строки является концом инструкции (точка с запятой не требуется).

```
Инструкция1
Инструкция2
…
ИнструкцияN
```

Вложенные инструкции в Python записываются после двоеточия, которым завершается основная инструкция, обычно с отступом под строкой основной инструкции. Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока он был одинаков.

```
Основная инструкция:
    Инструкция1 вложенного блока1
    Инструкция2 вложенного блока1:
        Инструкция1 вложенного блока2
        …
    ИнструкцияN вложенного блока1
```

Несколько специальных случаев. Иногда возможно записать несколько инструкций в одной строке, разделяя их точкой с запятой. Но не делайте это слишком часто! Помните об удобочитаемости. А лучше вообще так не делайте.

```
Инструкция1; Инструкция2; Инструкция3
```

Допустимо записывать одну инструкцию в нескольких строках. Достаточно ее заключить в пару круглых, квадратных или фигурных скобок

```
(ваша инструкция очень длинная и
не входит в экранную строку)
```

Более подробно с правилами оформления кода можно ознакомиться в документе Python Enhanced Proposal (https://defpython.ru/pep8).

Вывод информации на экран

Что такое функция в программировании – узнаем позже. Пока будем считать, что print() – это такая команда языка Python, которая выводит то, что в ее скобках на экран. Именно скобки () указывают, что print – это функция. Например, чтобы вывести на экран приветствие, надо записать:

```
print('Привет!')
```

или

```
print("Привет!")
```

В результате выполнения инструкции на экран будут выведены все символы, указанные в кавычках, включая начальные и конечные пробелы.

В скобках можно указывать число.

Целое:

Инструкция	**На экране будет выведено**
print(5)	5
print(–2)	-2

Дробное:

Инструкция	**На экране будет выведено**
print(3.14)	3.14
print(–143.222)	-143.222

Арифметическое или логическое выражение (если в скобках стоит выражение, то сначала оно выполняется, после чего print() уже выводит результат данного выражения):

Инструкция	На экране будет выведено
print(5 + 3)	8
print('Здравствуй' +' '+ 'мир!')	Здравствуй мир!
print('+' * 10)	++++++++++

Несколько значений, в том числе разного типа, разделяя их запятой:

Инструкция	На экране будет выведено
print('1 + 2 = ', 3)	1 + 2 = 3

При построении арифметических выражений нужно помнить о типах используемых данных. Так попытка выполнить функцию

```
print(1 + 'z')
```

приведет к ошибке типа

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Приведенную выше операцию все-таки можно выполнить, если превратить число 1 в строку "1". Для изменения одних типов данных в другие в языке Python предусмотрен ряд встроенных в него функций (что такое функция в принципе, вы узнаете в следующих лекциях). Поскольку мы пока работаем только с тремя типами (int, float и str), рассмотрим вызовы соответствующих им функций приведения – int(), float(), str().

Инструкция	На экране будет выведено
print(str(1) + 'a')	1a
print(int('3') + 4)	7
print(float('3.2') + int('2'))	5.2
print(str(4) + str(1.2), 4+float('1.2'))	41.2 5.2

Между указанными в print() значениями (будем называть их «список вывода») выводится один пробел. Этот разделитель можно изменить на любой другой символ (или последовательность символов). Для этого после списка вывода указывается дополнительный параметр sep в который заносится требуемый символ разделитель.

Инструкция	На экране будет выведено
print("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun", sep = "-")	Mon-Tue-Wed-Thu-Fri-Sat-Sun

print(1, 2, 3, sep = "//")	1//2//3

Во всех перечисленных случаях каждая новая инструкция print() выводит соответствующие значения на следующей строке. Параметр end позволяет изменить данное действие, указав любой другой символ или строку.

Инструкция	На экране будет выведено
print("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun", end = "**")	Mon Tue Wed Thu Fri Sat Sun**

Переход на новую строку обозначается символом '\n'. Если присвоить это значение параметру end, то никаких изменений в работе функции print вы не увидите. Однако если надо отступить одну дополнительную строку после вывода, то можно сделать так:

print(10, end = '\n\n')

Если результатом арифметической операции является вещественное число, количество разрядов дробной части можно ограничить, выполнив форматирование – следует записать строку: % точку, количество цифр в дробной части и букву f, затем указать % и выводимое значение.

Инструкция	На экране будет выведено
print(1/3)	0.3333333333333333
print('%.3f' % (1/3))	0.333
print('%.1f' % 243.234)	243.2
print('%.2f' % 243.238)	243.24
print('%.3f' % (1/2))	0.500

Можно установить также общее количество позиций экрана для вывода числа.

Инструкция	На экране будет выведено
print('%5.3f' % (5/3))	1.667
print('%5.1f' % -243.234)	-243.2
print('%7.2f' % 243.238)	243.24
print('%5.3f' % (1/2))	0.500
print('%7.3f' %(3000/7)	428.571

Можно также обеспечить форматированный вывод целых чисел. В этом случае в формате вывода указывается только общее количество позиций экрана, а также

буква d (говорящая о том, что нужно вывести целое число в десятичной системе счисления).

Инструкция	На экране будет выведено
print('%10d' % 5)	_____5

Перед форматированным выводом числа (и/или после него) можно записать также текст.

Инструкция	На экране будет выведено
print('Диаметр луны ', '%7.1f' %3447.8, ’ км’)	Диаметр луны 3447.8 км
print('%7.1f' % (8123.5 / 7), ' км')	1160.5 км

Для форматированного вывода данных используется также метод format(). Этот метод форматирует значение-аргумент, указанный в круглых скобках, по шаблону (образцу), который указан в кавычках и фигурных скобках.

Инструкция	На экране будет выведено
print("This is a {0}. It's {1}.".format("ball", "red"))	This is a ball. It's red.
print("This is a {0}. It's {1} {2}.".format(1, "a", "number"))	This is a 1. It's a number.
print('{:7.3f}{:7.3f}'.format((1/3), (1/9)))	__0.333__0.111

Третий способ создания форматированных строк – f-строки. Перед их открывающей кавычкой прописывается буква f. В самой строке внутри фигурных скобок записываются выражения на Python, которые исполняются, когда интерпретатор преобразует строку-шаблон в обычную. В выражении сначала выполняется сложение, после этого значение округляется до одного знака после запятой.

Инструкция	На экране будет выведено
print(f'price - {1.33 + 0.2:.1f}')	price - 1.5

До сих пор мы работали с числами (целыми и дробными) и строками, то есть познакомились с тремя типами данных.

– Целые числа (тип int) – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0);

– числа с плавающей точкой (тип float) – дробные, они же вещественные, числа (например, 1.45, -3.789654, 0.00453). Примечание: для разделения целой и дробной частей здесь используется точка, а не запятая;

– строки (тип str) – набор символов, заключенных в кавычки (например, "ball", "What is your name?", 'dkfjUUv', '6589').

Все эти данные хранятся в ячейках памяти компьютера. Когда мы вводим число, оно помещается в какую-то ячейку памяти. Но как потом узнать, куда именно? Как впоследствии обращаться к этим данными? Необходимо как-то пометить соответствующую ячейку.

Раньше, при написании программ на машинном языке, обращение к ячейкам памяти осуществляли с помощью указания их адреса, то есть конкретно сообщали, куда положить данные и откуда их взять. Однако с появлением ассемблеров при обращении к данным стали использовать словесные переменные, что куда удобней для человека.

Переменные

Переменная – имя для доступа к области памяти компьютера, выделенной для хранения информации определенного типа. Переменную можно сравнить с ящиком: имя переменной – это название ящика, а значение переменной – это то, что хранится в нем. То есть каждая переменная характеризуется именем и типом, и хранит значение.

В именах переменных можно использовать буквы, цифры (но имя не может начинаться с цифры) и знак подчеркивания «_». Желательно давать переменным «говорящие» имена, чтобы можно было сразу понять, зачем нужна та или иная переменная. Строчные и заглавные буквы различаются, то есть переменные с именами dlina и Dlina – это две разные переменные.

Есть набор слов, которые нельзя использовать в качестве имен переменных, так как эти слова «зарезервированы» в языке Python для определенных целей (эти слова называют «зарезервированными», или «служебными», или «ключевыми»).

Создание переменной происходит в момент выполнения операции присваивания значения.

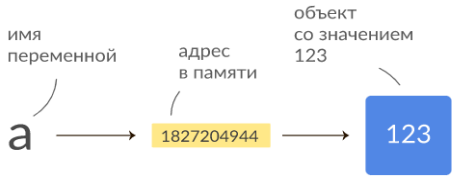

Рисунок 7 – a = 123

Переменной a присвоено целочисленное значение 123. Это значит, что на созданный объект (любые данные в Python являются объектами), представляющий

собой целое число со значением 123, находящийся в определенной области памяти, теперь ссылается переменная a, и обращаться к этому объекту следует по имени a.

> Важно: переменная в Python не хранит значение напрямую – она хранит лишь ссылку на объект (адрес).

Отметим, что справа от знака присваивания может находиться не только простое значение, но и выражение любой степени сложности – с использованием скобок, математических операторов, чисел и переменных, созданных на более ранних этапах.

Программа	На экране будет выведено
apples = 100 eat_day = 5 day = 7 apples = apples - eat_day * day print(apples)	65

Здесь фигурируют три переменные: apples, eat_day и day. Каждой из них присваивается свое значение. Выражение apples = apples - eat_day * day сначала выполняет подвыражение, стоящее справа от знака равенства. После этого его результат присваивается переменной apples, в результате чего ее старое значение (100) теряется. В подвыражении apples - eat_day * day вместо имен переменных на самом деле подставляются их значения, то есть числа 100, 5 и 7.

Запись a = b = 100 называется каскадным присваиванием и равносильна паре инструкций.

```
b = 100
a = b
```

Посмотрим, что при этом произойдет (рисунки 8-10).

Рисунок 8 – Значение переменных

Python не создает новый объект для записи в a – он просто создает переменную, которая ссылается на тот же объект, что и переменная b.

Пусть в какой-то момент вы захотели поменять значение переменной b.

```
b = 500
```

Python создал новый объект – целое число, и теперь переменная b ссылается на новый объект.

Рисунок 9 – Создание нового объекта

Предположим, что далее в программе вы присвоили b новое значение.

```
b = 'tree'
```

Создается новый объект-строка, и переменная b ссылается на него. На объект со значением 500 больше никто не ссылается. Следовательно, он больше не доступен и будет удален сборщиком мусора (тем самым освободив место в памяти).

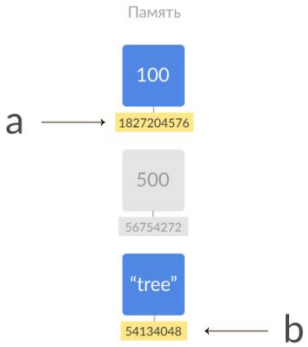

Рисунок 10 – Создание нового объекта

Подумайте, что мы увидим на экране в этом случае:

```
a = 1
b = 2
a = b
b = a
print(a, b)
```

А в этом?

```
a = 1
b = 2
b = a
a = b
print(a, b)
```

А как поменять значения местами? В Python разрешено множественное присваивание. Запись a, b, c = 7, 2, –5 равносильна отдельным инструкциям a = 7, b = 2, c = –5. В этом случае каждая переменная будет ссылаться на свой объект. Такое присваивание можно использовать при обмене данными между переменными. Например, так.

```
x, y = y, x
```

Объекты, на которые ссылаются переменные, могут задаваться не только внутри программы, но и вводиться в программу извне. Мы говорим о вводе данных пользователем.

Ввод данных в программу

За ввод в программу данных с клавиатуры в Python отвечает функция input().

– При вызове функции input() выполнение программы приостанавливается до тех пор, пока пользователь не введет текст на клавиатуре (приложение может ждать бесконечно долго).

– После нажатия на клавишу «Enter», функция input() считывает данные и передает их приложению (символ завершения новой строки не учитывается).

– Полученные данные присваиваются переменной и используются дальше в программе.

Программа	На экране будет выведено
print('Введите фамилию', end = ': ') name = input() print('Введите город проживания', end = ': ') city = input() print(f'Вас зовут {name}, Вы живете в городе {city}.')	Введите фамилию: ИВАНОВ Введите город проживания: КИРОВ Вас зовут ИВАНОВ, Вы живете в городе КИРОВ.

Можно совместить сообщение-подсказку и вызов функции ввода данных.

```
fam = input('Введите фамилию: ')
im = input('Укажите имя: ')
```

Заметим, что функция input() всегда возвращает в программу строковое значение, даже если ввести число. Но что делать, если надо получить число? Ответ: использовать функции преобразования типов.

Программа	На экране будет выведено
qty = int(input("Сколько апельсинов? ")) price = float(input("Цена одного? ")) summa = qty * price print("Заплатите ", summa, " руб.")	Сколько апельсинов? 100 Цена одного? 5 Заплатите 500 руб.

В данном случае выполняется функция input(), которая возвращает строку, а функция int() или float() сразу преобразует в число. Только после этого происходит присваивание переменной, то есть она сразу получает численное значение. Вообще, для того чтобы проверить тип переменной используется функция type().

Программа	На экране будет выведено
a = input() print(a, type(a))	12

a = 20 print(a, type(a)) a = a / 2 print(a, type(a))	12 <class 'str'> 20 <class 'int'> 10.0 <class 'float'>

Возможно присвоить данные введенные в одну строку через символ-разделитель в разные переменные. С этой целью используется метод split().

Программа	На экране будет выведено
fam, name, god = input().split(', ') print(f'fam: {fam} name: {name} god: {god}')	Иванов, Иван, 2002 fam: Иванов name: Иван god: 2002
a, b, c = input().split() print(f'a: {a}, b: {b}, c: {c}')	test 13 100 a: test, b: 13, c: 100

И еще немного о форматировании

Мы уже говорили об операторе %, обеспечивающем подстановку некоторого значения в строку (%s – подстановка строки, %d – десятичного числа). Теперь посмотрим, как он работает с переменными.

Заставим программу познакомиться с нами.

Программа
name = input('Как Вас зовут') print('Hello, %s' % name) kurs = int(input('На каком курсе ты учишься?')) print('Вау, %s. Ты уже на %d курсе' % (name, kurs)) age = int(input('А сколько тебе лет?')) print(f'Ты получишь диплом бакалавра, когда тебе исполнится {age+4-kurs}')
На экране увидим
Как Вас зовут Иван Hello, Иван На каком курсе ты учишься? 2 Вау, Иван. Ты уже на 2 курсе А сколько тебе лет? 22 Ты получишь диплом бакалавра, когда тебе исполнится 24.

Практическая работа

1) Напишите программу, которая предлагала бы пользователю решить пример 4.25 * 100 – 50/2. Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число?

2) Запросите у пользователя четыре числа. Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.

Контрольные вопросы

1) Что такое «алгоритм решения задачи»?
2) В чем особенности алгоритма, который называют «программой»?
3) Почему языки программирования высокого уровня так называются?
4) Что такое «транслятор»? Какие функции он выполняет?
5) Какие виды трансляторов вы знаете? В чем особенность каждого вида?
6) Что включает в себя система программирования?
7) Что можно указывать в скобках в инструкции print()? Что будет выведено на экран в том или ином случае?
8) Можно ли указывать в скобках несколько значений одного типа? Что при этом будет выведено на экран между ними? Как изменить этот разделитель?
9) Можно ли указывать в скобках несколько значений разного типа?
10) Что надо сделать, чтобы после выполнения инструкции print() следующие данные выводились на той же строке?
11) В чем особенность вывода на экран вещественных значений?
12) Как можно ограничить количество цифр в дробной части вещественного числа при его выводе на экран?
13) Какие величины в программе называют «переменными»?
14) Чем характеризуется каждая переменная?
15) Каковы правила присвоения имен переменным?
16) Почему желательно переменным давать «говорящие» имена?
17) С помощью какой инструкции можно ввести в программу значение переменной в ходе ее выполнения?
18) В чем заключается особенность ввода в программу в ходе ее выполнения числовых значений переменных?
19) Почему желательно выводить на экран подсказку перед вводом данных?