Уроки Python с нуля #3

Комментарии

Комментарии - часть кода, которая позволяет описать определенный участок кода Для того чтобы разработчики будут быстро понимать что происходит в определенной части вашего проекта. Для создания комментариев используется знак #. После # можно прописывать все что угодно, весь текст который будет после # будет считаться комментарием. Если нужно создать комментарий на несколько строк, соответственно, на каждой строке вы прописываете новый знак решетки.

Функция print

Чтобы написать текст:

- print (“Result”)

Чтобы написать число:

- print (5)

Если у вас есть множественный вывод, то вы его можете объединять за счет запятых:

- print (“Result:”, 5, 8, “!”)

Форматы вывода

При выводе значений между ними всегда проставляется определенный пробел и этот пробел мы можем удалить. Для этого в самый конец, нам необходимо установить дополнительное свойство. Оно называется «sep» или же «separate», то есть разделитель. Если я укажу, допустим, sep=”/” то это означает то, что каждый элемент, он разделяется именно за счет вот этой вот палки:

- print (“Result”, 5, 8, “!”, sep=”/”) → Result/5/8/!

Но если я укажу просто пустую строку sep=””, то в таком случае никаких пробелов, добавляться не будет.

- print (“Result”, 5, 8, “!”, sep=””) → Result58!

То есть за счет separate мы указываем как бы символ или несколько символов, за счет которых у нас каждый этот элемент будет выводиться на экран.

Перевод на новою строку

Кроме того, если мы попробуем вывести еще какой-либо текст на второй строчке, допустим, print “second line”. Первый и второй принт выводятся на разных строках.

- print (“Result”, 5, 8, “!”) → Result 5 8 !
- print (“Second line”) → Second line

И если вдруг вы хотите вывести все это на одной линии, то, конечно же, проще всего это записать :

- print (“Result”, 5, 8, “!”, “Second line”) → Result 5 8 ! Second line

Но, помимо этого, вы можете еще использовать дополнительное свойство. Оно называется end, и оно устанавливает конец для самой строки. И по умолчанию каждый принт, в конце добавляет как бы перевод на новую строку. Перевод на новую строку это у нас end. Если мы пропишем , в качестве значения для end просто пустую строку, то все сразу отображается в одну линию. Также есть символ \n который обозначает перевод на новую строку.

- print (“Result”, 5, 8, “!”, end = “”) → Result 5 8 ! Second line
- print (“Second line”)

Также здесь,можно указать какой-то символ.

Специальные символы

Чтобы нам вывести двойную кавычку внутри двойных кавычек, нам необходимо указать, что это будет обычный символ, и для указания того, что это обычный символ, мы должны использовать \.

- print ("Summer /" score", 5, 8, "!") ⟶ Summer " score 5 8 !

Конечно же, второй вариант, как можно было бы ее вывести, так это просто указать одинарные кавычки, а внутри, соответственно, была бы двойная кавычка.

- print ('Summer /" score', 5, 8, "!") ⟶ Summer " score 5 8 !

Ну и если еще говорить относительно специальных символов, то здесь есть символ, который создает табуляцию. Он записывается следующим образом \t. Если мы запустим, то мы замечаем, что вот здесь у нас создается определенный большой пропуск. И по сути этот пропуск является как будто символом табуляции.

- print ("Summer \t score", 5, 8, "!",) ⟶ Summer " score 5 8 !

Кроме того, если вдруг вам необходимо вывести слэш, вы должны написать его дважды. То есть таким образом вы как бы говорите, что вот этот слэш, который будет выводиться, он будет обычным символом.

- print (//) ⟶ /

Математические действия

В методе print, помимо того, что мы можем выводить что-либо, мы также можем производить какие-либо математические операции.

+ - добавление

- - вычитание

* - умножение

/ - деление

% - получение остатка при делении

** - возведение в степень

// - округлить результат деления к целому числу

Математические функции

Также существуют различные функции по типу функции print, которые служат для работы с математическими действиями. Функция min, позволяет нам найти минимальный элемент среди данных элементов, .

- print (min(3,-4, 7 ,15)) ⟶ -4

Также существует противоположная функция, она называется max, и эта функция, она находит максимальный элемент среди тех элементов, что сюда передаются.

- print (max(9, 24, 7 ,-5)) ⟶ 24

Помимо этих функций, также существуют еще дополнительные другие функции, которые также служат для работы с математикой. Ну и, например, функция abs, она находит у нас число по модулю.

- print (abs(-25)) ⟶ 25

Также здесь существуют такие функции как pow, эта функция, она позволяет нам возвести некое число в некую степень, и точно такое же можно было проделывать за счет двух знаков**, но также просто существует еще отдельная функция.

- print (pow(5, 3)) → 125

Ну и также здесь можно еще было бы рассмотреть такую функцию как round, эта функция, она округляет у нас числа. Причем round он всегда округляет к ближайшему, например, к большему либо к меньшему, а если мы прописываем просто // , то это всегда идет округление к меньшему, что мы могли с вами ранее и как раз заметить.

- print (round(5 / 3)) → 2
- print (5 // 3) → 1

Получение данных

Для получения значений от пользователя по сути используется такая встроенная функция, которая называется input. Кроме того, функция input, она может принимать еще параметры в этом параметре вы можете просто, ну как бы выписать определенную строку, которая будет отображаться пользователю.

- input (Your age:) → Your age: İ

Уроки Python с нуля #4

Что такое переменные?

Переменная - это ячейка в памяти, куда мы можем занести некую информацию и далее к этой информации мы можем ссылаться. Мы можем видоизменять эту информацию, мы можем удалить эту информацию, мы можем ее просто вывести на экран и, в принципе, можем проделать с ней любые другие действия.

- number = 12

Создание переменной

Для создания переменной нам необходимо просто указать для нее название, причем название может быть каким угодно, главное не использовать специальные символы. Лучше всего назвать переменные по смыслу. В качестве значения для этой переменной я хочу установить число, чтобы мне присвоить к этой переменной некое значение, я должен указать знак равенства и дальше указываю просто определенное значение.

- number = 3

Робота с переменной

После ее создания вы первое, что, наверное, можете сделать, так это просто вывести ее на экран. Дабы вывести эту переменную на экран, мы можем обратиться к функции print и просто внутрь этой функции мы передаем с вами саму переменную, переменную number.

- number = 12 → 12
- print(number)

Помимо вывода, вы также можете, конечно же, изменять значение самой переменной. То бишь, когда вы создаете переменные, вы можете изначально ей присвоить одно значение, а потом в ходе выполнения программы вы можете присвоить ей другое значение.

- number = 12
- print(“Result:”, number) → Result: 12
- number = 7 → Result:7
- print(“Result:”, number)

По необходимости вы переменные можете удалять. Для этого необходимо обращаться к директиве del, или же полностью она расшифровывается как delete, и дальше вы просто указываете определенную переменную, которую вы хотите удалить.

- number = 12 ⟶
- del number

Хранение разных значений

Помимо хранения целых чисел, как это у нас было сейчас сделано, мы также в переменных можем хранить и какие-либо другие значения. Например число с точкой.

- number = 12.5

Помимо того, что мы можем хранить вот подобные числа, мы также в переменных можем хранить определенные строки. Строку я могу помещать как в одинарной, так и в двойные кавычки, от этого разницы никакой не будет.

- word = “hello!” ⟶ hello!
- print(word)

Таким образом, мы можем переменно хранить, во-первых, целые числа, числа с точкой и еще строки.

Ну и помимо вот этих вот всех типов данных, существует еще один такой нестандартный тип данных, он называется булевый, и за счет этого типа данных мы можем хранить в переменной одно из двух возможных значений, либо true, либо false, подобные значения они зачастую используются в различных проверках.

- boolean = True ⟶ True
- print(boolean)

- boolean = False ⟶ False
- print(boolean)

Типы данных

В языке Python не существует строгой типизации. Это означает то, что при создании переменной нам не нужно указывать, какой тип данных будет в этой переменной. В то же время в каждой переменной у нас есть определенный тип данных.

Integer (int) - это специальный тип данных, который отвечает у нас именно за целые числа.

- number = 12 #int

Когда мы создаем число с точкой, мы создаем число с типом данных float.

- number = 12.5 #float

Когда мы создаем строку, мы создаем тип данных string.

- word = “hello!” #string

Ну и когда мы создаем с вами некую boolean переменную, то мы создаем переменную на основе такого типа данных, как bool.

- bolean= True #bool

Все эти типы данных, они не явно здесь существуют. Разные типы данных и, соответственно, их сложение невозможно.

- word = “Hello” #string
- boolean = True #bool ⟶ Error
- print(boolean + word)

И на самом деле справиться с этой ошибкой можно несколькими способами. Во-первых, конечно же, можно тут все выводить через запятую, это первый вариант.

- word = “Hello” #string
- boolean = True #bool → Hello True
- print(boolean, word)

А второй вариант, это приведение типов.

Приведение типов

- digit = -4.604560 #float
- word = “Hello” #string → Error
- print (digit + word)

И чтобы нам float преобразовать к строке, мы с вами должны вот здесь воспользоваться такой функцией, которая называется как str. Это функция, она берет определенное число, будь то float, или же будь то integer, и это число она преобразует к строке. Если я такое запущу, мы замечаем, что никаких изменений здесь не произошло. У нас все обработалось корректно, никаких ошибок у нас больше.

- digit = -4.604560 #float
- word = “Hello” #string → -4.604560 Hello
- print (str(digit)+ word)

В то же время, мы давайте вот попробуем с вами еще одну какую-то переменную создать, например, я ее назову как str.num, и это будет переменная, которая будет содержать число, только это число, оно будет в формате именно строки. То есть здесь, опять же, тип данных у этой переменной именно как string. Я, предположим, хочу добавить к number, я хочу добавить еще str.num. Если я такое попробую сейчас запустить, у нас будет вызываться ошибка, потому что number – это у нас int, а str.num – это у нас string.

- number = 5 #int
- str_num= “5” #string → Error
- print (number + str_num)

Опять же, разные типы данных, поэтому вызывается ошибка. Ну и чтобы нам это все корректно здесь обработать, мы просто можем вызвать другую функцию, она называется int, ну и, соответственно, str – он приводил у нас какое-либо число к строке, а int – он делает обратное, он приводит определенную строку к числу. При этом, если мы попытаемся вот эту вот функцию использовать для такой переменной как word, то, конечно же, у нас будет вызываться ошибка, потому что в этой переменной у нас, ну, в принципе не находится число, да, здесь находится текст, поэтому все-таки при попытке преобразования

- number = 5 #int
- str_num= “5” #string → 10
- print (number + int(str_num))

Кроме того, для интереса мы давайте еще здесь попробуем вывести word. Конечно же, ну, то есть вот это вот наши перемены, конечно же, я могу это вывести через запятую, в таком случае все это обработается корректно.

- number = 5 #int
- word = “Result” #string
- str_num= “5” #string → Result: 10
- print (word, number + int(str_num))

Но, предположим, я все-таки хочу через знак плюс вывести это все. Чтобы мне такое реализовать, мне нужно вот это вот сложение, это все нужно преобразовать к типу данных string. Ну и чтобы это сделать, я предлагаю вот что указать. Мы здесь просто вызываем функцию str, а внутрь этой функции мы помещаем сложение вот этих вот чисел. То есть после того, как будет выполнено сложение, у нас сработает функция str, все это дополнительно

будет еще объединено с переменой word, и если это все запустить, вот оно все абсолютно корректно здесь сработало.

- number = 5 #int
- word = “Result:” #string
- str_num= “5” #string ⟶ Result: 10
- print (word + str(number + int(str_num)))

Всегда выполните преобразование типов данных. Для этого используются такие функции как str, int, также можно использовать такую функцию как float, она соответственно преобразовывает к типу данных float, или же можно использовать такую функцию как bool, она позволяет преобразовать к типу данных bool.

Робота с пользователем

Я хочу разработать такую программу, где мы будем получать от пользователя два числа, и дальше мы будем выводить пользователю все возможные математические операции, связанные с этими двумя числами.

- `num1 = int(input ("Enter first number:"))`
- `num2 = int(input ("Enter second number:"))`
- `print("Result:", num1 + num2 )`
- `print("Result:", num1 - num2 )`
- `print("Result:", num1 * num2 )`
- `print("Result:", num1 / num2 )`
- `print("Result:", num1 ** num2 )`
- `print("Result:", num1 // num2 )`

То есть здесь вы можете производить различные удобные вам математические

Сокращенные действия

Кроме того, если вам нужно выполнить некую математическую операцию, связанную с одной и той же переменной, то это можно сделать несколькими способами. Вот, например, предположим, что к переменной 1, я бы хотел добавить число 5, это можно записать вот в таком формате, это будет абсолютно корректно работать.

- `num1 = int(input ("Enter first number:"))`
- `num1 = num1 + 5`

Но также можно использовать более сокращенный формат. Здесь мы просто указываем плюс равно и дальше добавляем как бы определенное число.

- `num1 = int(input ("Enter first number:"))`
- `num1 += 5`

Помимо добавления, здесь вы можете также производить вычитание, это означает то, что от этой переменной вы вычитаете число 5.

- `num1 = int(input ("Enter first number:"))`
- `num1 -= 5`

Также здесь можно производить умножение, деление, либо же остаток при делении.

- `num1 = int(input ("Enter first number:"))`
- `num1 *= 5`

- `num1 = int(input ("Enter first number:"))`
- `num1 /= 5`

- `num1 = int(input ("Enter first number:"))`
- `num1 %= 5`

И еще один момент, который я забыл упомянуть, это то, что на самом-то деле вы можете умножать не только числа, но также можете еще и умножать строки.

- `word = “Hi”`
- `print(word * 2)`  `HiHi`