Базовые синтаксические конструкции

Comments

Как ни странно, первая вещь, которую мы будем изучать – как правильно показать компьютеру, какую часть текста игнорировать. Текст, написанный в программе, но не выполняемый компьютером, называют комментарием. Python интерпретирует что-либо после знака # как комментарий.

Комментарии могут:

• Обеспечить контекст для того, почему какая-то часть программы написана именно так:

```
# This variable will be used to count the number of times anyone tweets the word persnickety
persnickety_count = 0
```

• Помочь другим людям прочитать Ваш код:

```
# This code will calculate the likelihood that it will rain tomorrow
complicated_rain_calculation_for_tomorrow()
```

• Игнорировать часть кода, чтобы проверить, как работает программа без нее:

```
# useful_value = old_sloppy_code()
useful_value = new_clean_code()
```

Print

Теперь необходимо научить наш компьютер общаться. Данное умение ценно: компьютер может ответить на многие вопросы, которые мы имеем о том, «как» или «почему» или «что» он делает. В python функция print () используется, чтобы иметь возможность получать информацию о выполнении компьютером тех или иных операций. Сообщение, которое будет напечатано, должно быть окружено кавычками:

```
# from Mary Shelley's Frankenstein
print("There is something at work in my soul, which I do not understand.")
```

В данном примере мы хотим от нашей программы, чтобы она вывела в консоль выдержку из известной книги. Печатные слова, которые появляются в результате работы функции print (), являются выходом. Результат выполнения данной функции был бы:

```
There is something at work in my soul, which I do not understand.
```

Строки

Программисты определяют блоки текста как строки. В нашем последнем приложении мы создали строку «Привет мир!». В python строка окружена двойными кавычками («Привет мир») или одинарными кавычками ('Привет мир'). Это не имеет значения, какой вид Вы используете.

Переменные

Языки программирования предлагают метод для того, чтобы хранить данные для повторного использования. Предположим, есть приветствие, к которому мы хотим добавить дату, которую мы должны повторно использовать, или идентификатор пользователя, который необходимо запомнить. Для этого можно создать переменную, которая может хранить некое значение. В python мы назначаем переменные при помощи знака равенства (=).

```
message_string = "Hello there"
# Prints "Hello there"
print(message_string)
```

В вышеупомянутом примере мы храним сообщение «Hello there» в переменной, названной message_string. Название переменных не может содержать пробелов или символов кроме подчеркивания (_). Имена переменных не могут начинаться с чисел, но у них могут быть числа после первого символа (например, cool_variable_5 приемлемо).

Неспроста мы называем эти объекты «переменными». Если контекст программы изменяется, мы можем обновить переменную, при этом выполнить тот же логический процесс.

```
# Greeting
message_string = "Hello there"
print(message_string)

# Farewell
message_string = "Hasta la vista"
print(message_string)
```

Выше, мы создаем переменную message_string, присваиваем необходимое значение и выводим приветствие. После того, как мы приветствуем пользователя, мы хотим попрощаться. Мы тогда присваиваем message_string новое значение и выводим новое сообщение.

Задание:

1. Создайте переменную clothes. Присвойте ей значение «домашняя одежда»
2. Выведите в консоль следующее: «У меня большой гардероб». Следующей строкой выведите: «Утром лучше всего подходит домашняя одежда» с использованием созданной переменной. Вывести ту же строку для дня, вечера и ночью.
3. Проделать тоже упражнение для завтрака, обеда и ужина введя переменную meal и заменив первую строку на «мои предпочтения в еде».

Ошибки

Люди подвержены к допущению ошибок. Люди также, как правило, отвечают за создание компьютерных программ. В языках программирования существует возможность нахождения ошибок кода, а также их разъяснения.

Python называет эти ошибки «Errors» и указывает на местоположение ошибки символом ^.

Существует 2 вида возможных ошибок в python:

- SyntaxError – означает, что что-то не так со способом, которым Ваша программа написана — например, несоответствующая пунктуация, которая не должна быть при данной операции. Также недостающая круглая скобка может привести к такого рода ошибке.
- NameError - происходит, когда интерпретатор Python не может распознать слово. Код, который содержит что-то, что похоже на переменную, которой не существует.

Числа

В python существует несколько числовых типов данных. Существует несколько вариантов хранения чисел. Какой способ выбрать зависит от цели, а также от того, какое число необходимо хранить.

Integer или int, является целым числом. Такой тип не имеет никакой десятичной запятой и содержит все целые числа (1, 2, 3...), а также отрицательные целые числа и 0. Если бы ты считал число людей в комнате, количество мармеладок в банке или количество кнопок на клавиатуре, то ты, вероятно, использовал бы целое число.

Тип float, является десятичным числом. Это может использоваться, чтобы представлять числа с плавающей запятой, а также точные измерения. Если ты измерял длину своей стены спальни, вычисляя среднюю экзаменационную отметку седьмого класса, то вероятно, использовал тип float.

Числа присваиваются переменным и использоваться непосредственно в программном коде:

```
an_int = 2
a_float = 2.1

print(an_int + 3)
# prints 5
```

Выше мы определили целое число и число с плавающей запятой как переменные an_int и a_float. Мы вывели в консоль сумму переменной an_int с числом 3. Мы называем число 3 здесь цифрой, не используя переменную.

Вычисления

Python выполняет сложение, вычитание, умножение и деление с, - *, и/.

```
# Prints "500"
print(573 - 74 + 1)

# Prints "50"
print(25 * 2)

# Prints "2.0"
print(10 / 5)
```

Заметьте, что, когда мы выполняем деление, у результата появляется десятичный разряд. Это вызвано тем, что python преобразовывает весь ints в float при выполнении деления. В более старых версиях python (2.7 и ранее) не происходило этого преобразования, и деление целого числа всегда округляется в меньшую сторону к самому близкому целому числу.

Деление может бросить свою собственную специальную ошибку: ZeroDivisionError. Python поднимет эту ошибку, пытаясь разделиться на 0.

Изменяющиеся числа

Переменные, которые являются назначенными числовыми значениями, можно рассматривать как сами числа. Две переменные могут быть сложены друг с другом, перемножены, или разделены друг на друга. Выполнение арифметических действий на переменных не заменяет переменную — Вы можете только обновить переменную, используя знак =.

```
coffee_price = 1.50
number_of_coffees = 4

# Prints "6.0"
print(coffee_price * number_of_coffees)
# Prints "1.5"
print(coffee_price)
# Prints "4"
print(number_of_coffees)

# Updating the price
coffee_price = 2.00

# Prints "8.0"
print(coffee_price * number_of_coffees)
# Prints "2.0"
print(coffee_price)
# Prints "4"
print(number_of_coffees)
```

Мы создаем две переменные и присваиваем числовые значения. Выполняем вычисление на них. При этом переменные не обновляются! Когда мы обновляем coffee_price переменную и выполняем вычисления снова, используются обновленные ценности для переменной!

Задание

Вы решили создать одеяло из лоскутов! Чтобы вычислить количество лоскутов (квадратов), необходимо создать 2 переменные: quilt_width и quilt_length. Давайте сделаем это первое одеяло 8 квадратами в ширину и 12 квадратами в длину. Вывести в консоль количество квадратов для создания одеяла!

Оказывается, что одеяло потребовало немного больше материала, чем Вы имеете под рукой! Давайте сделаем одеяло с 8 областями в длину. Какое количество квадратов получится?

Самостоятельно! Рассчитать площадь прямоугольника со сторонами 23, 13.