Функции

Язык python имеет особенность в построчном исполнении кода. Например, если мы последователь хотим вывести в консоль имя и возраст человека, мы сделаем следующее:

```
>>name = ‘Анна’
>>age = 18
>>print(name)
>>print(age)
```

Предположим, что мы хотим вывести несколько разных человек с различными именами и возрастами. Что в таком случае потребуется сделать? Последовательно повторять вышеуказанные строчки кода несколько раз. Например:

```
>>name = ‘Анна’
>>age = 18
>>print(name)
>>print(age)
>>name = ‘Александр’
>>age = 19
>>print(name)
>>print(age)
>>name = ‘Олег’
>>age = 18
>>print(name)
>>print(age)
```

Чем больше данных о различных людях мы захотим вывести в консоль, тем больше одинаковых строк появится в коде. Код становится нагроможденным и не читаемым. Как же сделать так, чтобы однотипные операции можно было бы прописать 1 раз, а потом просто исполнять их с разными данными? (в примере операции – это присвоение имени переменным и вывод в консоль, а параметры – непосредственно имена (Анна, Александр, Олег) и возраста (18,19,18))

При решении таких задач в python используются специальные объекты, которые называются функциями.

В данном занятии будет рассмотрено:

- Что есть функция
- Способы задания функций
- Вызовы функций
- Типы параметров функций

Понятие функции

Возвращаемся к нашему примеру с именами. Для того, чтобы выделить этот код в отдельную структуру, которая будет срабатывать каждый раз, когда мы хотим вывести в консоль имя и возраст нового человека, мы можем создать функцию и назвать ее

printing_name_and_age() и вызывать ее каждый раз при необходимости вывода в консоль информации об имени и возрасте разных людей.

Как задается функция

Для того, чтобы создать функцию необходимо использовать ключевое слово def (от defend). Далее необходимо задать имя функции. Для имен функций действуют те же правила, что и для имен переменных. После имени функции пишут скобки, а после скобок двоеточие. Вернемся к нашему примеру с выводом имени и возраста в консоль. Создадим функцию printing_name_and_age(), которая будет выводить в консоль нужные нам значения:

```
>>name = ‘Анна’
>>age = 18
>>def printing_name_and_age():
>>      print(name)
>>      print(age)
```

Как видно из примера, та часть кода, которая включена в функцию, имеет отступ от края строки (как в случае с if-else). Для того, чтобы код в функции исполнился, необходимо ее вызвать. Для вызова функции необходимо написать ее название. В нашем примере:

```
>>name = ‘Анна’
>>age = 18
>>def printing_name_and_age():
>>      print(name)
>>      print(age)
>> printing_name_and_age()
```

Последняя строка – это вызов функции. Если не осуществить вызов функции, то код, находящийся внутри нее (все, что с отступами после двоеточия) не будет исполнен. С использованием функций наш код преобразуется в следующий:

```
>>def printing_name_and_age():
>>      print(name)
>>      print(age)
>>name = ‘Анна’
>>age = 18
>> printing_name_and_age()
>>name = ‘Александр’
>>age = 19
>> printing_name_and_age()
>>name = ‘Олег’
>>age = 18
>> printing_name_and_age()
```

Как видим, у нас сократилось количество повторяющихся строк. Однако, нам бы хотелось не назначать каждый раз значения переменным name и age, а просто вызывать ее с разными параметрами.

Входные параметры функции

В python есть возможность задавать входные параметры функций. Входные параметры записываются в скобках после названия функции и отделяются запятой. Вернемся к нашему примеру. Мы хотим, чтобы функция def printing_name_and_age() принимала в качестве входных параметров имя и возраст людей, а потом выводила их в консоль. С использованием параметров функции это будет выглядеть следующим образом:

```
>>def printing_name_and_age(name, age):
>>      print(name)
>>      print(age)
```

Значения этих параметров будет задаваться при вызове функции внутри скобок. Для вывода данных из более раннего примера код будет выглядеть следующим образом:

```
>>def printing_name_and_age(name, age):
>>      print(name)
>>      print(age)
>>printing_name_and_age('Анна', 18)
>>printing_name_and_age('Александр', 19)
>>printing_name_and_age('Олег', 18)
```

В консоль будет выведено:

```
Анна
18
Александр
19
Олег
18
```

Как видно из примера, при вызове функции в скобках мы указали параметры, которые подставились в соответствующую переменную.

Но как программа узнала, какой параметр куда подставить? На самом деле, значения подставились в параметры функции в порядке их следования. Так как при создании функции мы в скобках написали сначала name, потом age, то 'Анна' подставилось в параметр name, как указанное первым в скобках при вызове функции. Если бы был изменен вызов функции, то есть поменяли бы местами параметры 'Анна' и 18 так:

```
>>def printing_name_and_age(name, age):
>>      print(name)
>>      print(age)
>>printing_name_and_age(18, 'Анна')
```

То вывод бы выглядел так:

```
18
Анна
```

Как видно, тогда в name записалось значение 18, а в age значение 'Анна'. То есть значения параметров должны быть перечислены в вызове функции в том порядке, в котором заданы

сами параметры при создании функции. Так как значения параметров при вызове функции задаются в зависимости от позиции параметров при создании функции, такие параметры называют позиционными.

Для вызова функции необходимо в скобках прописывать значения для всех имеющихся в функции параметров. В нашем примере в функции задано 2 параметра name и age. Что будет, если задать значение только для одного из параметров? Например,

>>printing_name_and_age(‘Анна’)

В этом случае будет выведено сообщение об ошибке:

```
TypeError: printing_name_and_age() missing 1 required positional argument: 'age'
```

Ключевые параметры

Предположим, что мы выводим в консоль информацию о студентах 1-го курса. Мы знаем, что большинству студентов 18 лет. Людей с другим возрастом всего 3 человека. Как быть в этой ситуации?

Для таких случаев существуют параметры, которые имеют значения по умолчанию. То есть мы можем задать значение, которое будет подставляться в соответствующий параметр в случае, если мы его не укажем при вызове функции. Аргументы, имеющие значения по умолчанию, называются ключевыми

В нашем с именами и возрастом можно сделать так:

```
>>def printing_name_and_age(name, age=18):
>>      print(name)
>>      print(age)
>>printing_name_and_age(‘Анна’)
```

Тогда будет выведено:

```
Анна
18
```

То есть можно заметить, что при вызове функции возраст мы не указали, однако при создании функции было прописано значение по умолчанию age=18, поэтому не вывелась ошибка, а в консоль было выведено значение 18 в качестве возраста.

Если мы хотим, чтобы возраст был какой-то отличной от значения по умолчанию цифрой, то мы должны указать ее при вызове функции. Например, для вывода данных Александра, которому 19 лет, вывод будет выглядеть так:

>>printing_name_and_age(‘Александр’, 19)

Тогда весь наш исходный код будет выглядеть так:

```
>>def printing_name_and_age(name, age=18):
>>      print(name)
>>      print(age)
>>printing_name_and_age(‘Анна’)
>>printing_name_and_age(‘Александр’, 19)
>>printing_name_and_age(‘Олег’)
```

Необходимо запомнить, что ключевые аргументы, имеющие значения по умолчанию всегда указываются в конце, то есть после позиционных. Поэтому следующий код:

```
>>def printing_name_and_age(age=18, name):
>>      print(name)
>>      print(age)
>>printing_name_and_age(‘Анна’)
```

Выдаст ошибку:

```
SyntaxError: non-default argument follows default argument
```

Задание

1. Определите функцию с именем create_spreadsheet (), которая принимает один аргумент title и выводит в консоль только одну строку «Создание электронной таблицы с именем» + title.

2. Вызовите create_spreadsheet с значением title «Загрузки».

3. Добавьте параметр row_count в определение функции. Установите значение по умолчанию 1000.

4. Измените оператор print «Создание электронной таблицы с названием title with row_count lines», где title и row_count заменяются их соответствующими значениями.

Помните, чтобы объединить число в строковый объект, вам сначала нужно привести row_count к строке, используя str (). В противном случае вы получите ошибку TypeError.

5. Вызовите create_spreadsheet () с названием набора в «Приложения» и row_countset в 10.