Возвращаемые значения

В данном занятии будет объяснено, как использовать результаты вычисления функций для их дальнейшего использования для вычислений.

Оператор return

До сих пор мы видели только функции, которые выводят какой-то результат в консоль. Функции также могут возвращать значение пользователю, чтобы это значение можно было изменить или использовать позже. Когда результат исполнения функции может быть сохранен в переменной, он называется возвращаемым значением функции. Мы используем ключевое слово return, чтобы вернуть значение функции.

Предположим, у нас есть две функции. Первая складывает 2 числа, вторая вычитает из первого второе. Данные функции реализованы следующим образом:

```
>>def summing(a, b):
    >>result = a+b
    >>print(result)

>>def subtraction (a, b):
    >>result = a-b
    >>print(result)
```

Предположим, нам нужно вычислить следующее выражение: 2+3-1. Как видно из формулы, мы используем как сложение, так и вычитание последовательно. Вызовем первую функцию сложения и попробуем присвоить значение переменной:

```
>>x=2
>>y=3
>>z=1
>>result_sum = summing(x, y)
>>print(result_sum)
```

В консоль выведется:

```
5
None
```

Мы видим, что result_sum не содержит результат суммирования, а содержит None. Как поместить в переменную result_sum результат суммирования? Для этого мы используем оператор return.

```
>>def summing(a, b):
    >>result = a+b
    >>return result

>>def subtraction (a, b):
    >>result = a-b
   >> return result
```

```
>>x=2
>>y=3
>>z=1
>>result_sum = summing(x, y)
>>print(result_sum)
```
В результате будет выведено:

```
5
```

То есть оператор return позволил поместить значение результата суммирования в переменную result_sum. Почему нельзя просто вызвать переменную result из функции? Это происходит из-за областей видимости.

Области видимости

Допустим, у нас есть функция из последнего упражнения, которая создает строку о специальном элементе:

```
>>def create_special_string(special_item):
   >>return "Our special is " + special_item + "."
```

Что если мы хотим получить доступ к переменной special_item вне функции? Можем ли мы использовать ее?

```
>>def create_special_string(special_item):
   >>return "Our special is " + special_item + "."

>>print("I don't like " + special_item)
```

Если мы попытаемся запустить этот код, мы получим NameError, сообщая нам, что 'special_item' не определен. Переменная special_item была определена только внутри пространства функции, поэтому она не существует вне функции. Область действия special_item - только функция create_special_string.

Переменные, определенные вне области действия функции, могут быть доступны внутри тела функции:

```
>>header_string = "Our special is "

>>def create_special_string(special_item):
   >>return header_string + special_item + "."
>>print(create_special_string("grapes"))
```

Здесь нет ошибки. header_string может использоваться внутри функции create_special_string, потому что область действия header_string - весь файл. Этот файл будет выводить:

```
Our special is grapes.
```

Задание

1. Функция define_age создает переменную с именем age, которая представляет собой разницу между текущим годом и годом рождения, оба из которых являются входными данными для функции. Добавьте строку, чтобы возраст возвращался в качестве результата вычислений для дальнейшего использования.

>>def calc_age (current_year, birth_year):

#возраст = текущий_год - рождение_год

2. Вне функции вызовите Calculate_age со значениями 2049 (current_year) и 1993 (birth_year) и сохраните значение в переменную my_age.

3. Вызовите Calculate_age со значениями 2049 (current_year) и 1953 (birth_year) и сохраните значение в переменной с именем dads_age.

Выведите на консоль строку «Мне X лет, а моему отцу Y лет», с my_age, где X и dads_age, где Y.

Несколько возвращаемых значений

Иногда может понадобиться вернуть более одного значения из функции. Мы можем вернуть несколько значений, разделив их запятой:

```
>>def square_point(x_value, y_value):
 >> x_2 = x_value * x_value
 >> y_2 = y_value * y_value
 >> return x_2, y_2
```

Эта функция принимает значение x и значение y и возвращает их оба в квадрате. Мы можем получить эти значения, указав их при вызове функции:

```
>>x_squared, y_squared = square_point(1, 3)
>>print(x_squared)
>>print(y_squared)
```

В консоль выведется:

```
1
9
```

Задание

1. Напишите функцию с именем get_boundaries (), которая принимает два параметра: числовой параметр target и числовой параметр margin.

Следует создать две переменные:

• low_limit: target минус margin.

• high_limit: margin прибавить к target

2.Возвратить значения low_limit и high_limit из функции в указанном порядке.

3.Вызовите функцию с параметром target, равным 100 и с margin 20. Сохраните возвращаемые значения в переменные, называемые low_limit и high_limit.

4.Выведите в консоль строку:

Нижний предел: low_limit, верхний предел: high_limit

Со значениями low и high, которые вы получили из функции get_boundaries ().

Задание

1. Определите функцию с именем repeat_stuff, которая принимает два входа, stuff и num_repeats.

Мы хотим, чтобы эта функция вывела в консоль строку с количеством повторений nre_repeats. Пока только поместите пустой оператор print внутри функции.

2. Вне функции вызовите repeat_stuff.

   Вы можете использовать значение "Row" для stuff и 3 для num_repeats.

3. Измените оператор вывода внутри repeat_stuff на returnstatement.

Он должен возвращать stuff * num_repeats.

Примечание: умножение строки просто приводит к повторению значения строки! Например:

```
"na"*6
```

приводит к строке "nanananananana".

4. Присвойте параметру num_repeats значение по умолчанию 10.

5. Соедините результат repeat_stuff ("Row", 3) и строку "Your Boat." вместе и сохраните результат в переменную с именем lyrics.

6. Создайте переменную с именем song и присвойте ей значение repeat_stuff, вызываемого с только с stuff.

7. Вывести песню в консоль.