Функции

Date: 01/04/2024

Tochilkina Ksenya Stanislavovna

Функции

Мы уже успели воспользоваться большим количеством функций, например, print() и input(), list(), map() и другие. Но все эти функции были либо встроенными, либо импортированными из модулей и библиотек языка Python.

Давайте разберемся в общей концепции функций. **Функция** — это фрагмент кода, выполняющий ту или иную задачу. Кроме того, это один из способов повторного использования кода, ведь одну и ту же функцию можно вызывать в своих программах снова и снова. Функции являются важной частью языка программирования Python. Они позволяют создавать блоки кода, которые могут быть вызваны и выполнены в разных частях программы. Функции помогают организовать код, делая его более читаемым, модульным и повторно используемым.

Объявление функций

Функции в Python определяются с использованием ключевого слова def, за которым следует имя функции и круглые скобки, содержащие аргументы функции. Аргументы - это значения, которые могут быть переданы в функцию при ее вызове. После круглых скобок следует двоеточие, а затем блок кода, который будет выполнен при вызове функции.

Пример определения функции:

```
def greeting(name):
    print("Hello, " + name + "!")
```

Функция состоит из трех частей:
* имени,

* аргументов,
* тела.

В приведенном выше фрагменте кода вы можете увидеть ключевое слово def. Это определяет блок кода, который представляет функцию. Имена функций должны следовать тем же правилам, что и имена переменных. Имя этой функции – greeting, после нее следуют скобки и двоеточие.

Внутри скобок находится список аргументов, который может быть пустым или, как в нашем случае иметь один или несколько аргументов.

Тело – это блок кода, идущий сразу после строки, которая начинается с def (сокращение от define — определить). Блок кода должен иметь отступ (табуляцию).

Вызов функций

Функции вызываются с использованием имени функции, за которым следуют круглые скобки, содержащие значения аргументов, если они есть. При вызове функции код внутри функции выполняется.

Пример вызова функции:

```
greeting("John")
```

Аргументы по умолчанию

Иногда вам нужно, чтобы функции изменяли свое поведение в различных условиях. Для этого вы можете использовать аргументы функции. Допустим, вы хотите поздороваться не только с пользователем “Гость”, но и с пользователем под любым именем.

```
1  def greeting(name="Гость"):
2      print("Привет, " + name + "!")
3
4  greeting()
5  greeting('Ksenya')
```

```
Привет, Гость!
Привет, Ksenya!

Process finished with exit code 0
```

Есть три варианта отделения переменных:
1. Отделение с помощью запятых:
print('Hi',name,'How are u?)

2. Отделение с помощью знака "+":
print('Hi'+name+'How are u?')

3. Отделение с помощью форматирования:
print(f'Hi {name} How are u?')

Возвращение значений

Функции могут возвращать значения с помощью ключевого слова return. Например, предположим, что мы создали функцию, которая конвертирует километры в метры, и нужно сохранить конвертированное значение для использования в последующих вычислениях, а непросто отобразить его напрямую на экране. Значение, возвращаемое функцией, может быть использовано в другой части программы.

Пример:

```
def converter(km):
    return km*1000

km = int(input('Введите км: '))
print (f"{km} километров = {converter(km)} метров")
```

```
D:\pythonProject\venv\Scripts\python.exe D:
Введите км: 8
8 километров = 8000 метров

Process finished with exit code 0
```

Анонимные функции

Что такое lambda-функции в Python?

Lambda-функции в языке программирования Python - это анонимные функции, которые могут быть определены без использования ключевого слова def. Они обычно используются для создания простых функций, которые не требуют большого количества кода и могут быть использованы внутри других функций или выражений.

Как определить lambda-функцию в Python?

Определение lambda-функции в Python осуществляется с использованием ключевого слова lambda, за которым следуют аргументы функции, двоеточие и выражение, которое должно быть выполнено функцией.

Вот пример определения lambda-функции, которая умножает число на 2:

```
result = lambda x: x * 2
print(result(5))  # Вывод: 10
```

В этом примере x - это аргумент функции, а x * 2 - это выражение, которое будет выполнено функцией. После определения lambda-функции, ее можно вызвать, передав нужные аргументы.

Как использовать lambda-функции в Python?

Lambda-функции в Python могут быть использованы в различных ситуациях. Они могут быть переданы в качестве аргументов другим функциям, использованы внутри списковых выражений или применены к последовательностям данных с помощью функций высшего порядка, таких как map, filter и reduce.

Вот несколько примеров использования lambda-функций в Python:

- Передача lambda-функции в качестве аргумента другой функции:

```
def apply_function(func, x):
    return func(x)

result = apply_function(lambda x: x * 2, 5)
print(result)  # Вывод: 10
```

- Применение lambda-функции к последовательности данных с помощью функции map:

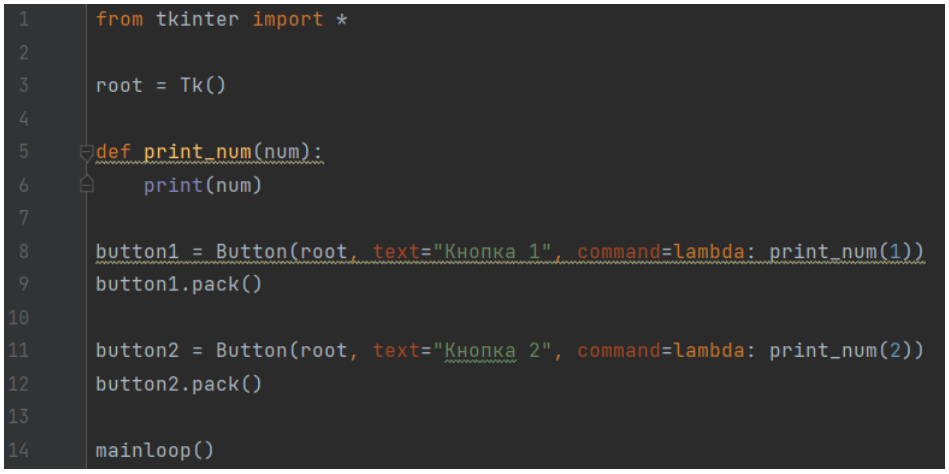
```
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers)  # Вывод: [1, 4, 9, 16, 25]
```

Первый аргумент map() — это функция, которую мы используем для применения к каждому элементу. Python вызывает функцию один раз для каждого элемента итериируемого объекта, который мы передаем в map() , и возвращает измененный элемент в объект map.

Lambda-функция с библиотекой tkinter

Можно использовать lambda-функции вместе с библиотекой tkinter для определения обработчиков событий и других функций.

Вот пример использования lambda-функций с библиотекой tkinter:



```python
from tkinter import *

root = Tk()

def print_num(num):
    print(num)

button1 = Button(root, text="Кнопка 1", command=lambda: print_num(1))
button1.pack()

button2 = Button(root, text="Кнопка 2", command=lambda: print_num(2))
button2.pack()

mainloop()
```

В этом примере создаются две кнопки с помощью библиотеки tkinter. Каждая кнопка имеет связанную с ней lambda-функцию в качестве обработчика события command. Когда кнопка нажимается, соответствующая lambda-функция вызывается и выводит число в консоль.

Какие преимущества и ограничения у lambda-функций в Python?

Преимущества lambda-функций в Python:

- Краткость: lambda-функции позволяют определить функцию в одной строке кода без необходимости использования ключевого слова def.
- Удобство использования: lambda-функции могут быть использованы внутри других функций или выражений без необходимости определения отдельной функции.

Ограничения lambda-функций в Python:

- Ограниченность выражениями: lambda-функции могут содержать только одно выражение, что ограничивает их использование в сложных сценариях.
- Отсутствие имени: lambda-функции не имеют имени, что может затруднить их отладку и понимание кода.