

< Teach
Me
Skills />

Занятие 4. Функции

1

Операторы присваивания

Формы операторов присваивания

```
: # каноническая  
example = "example"
```

```
: # комбинированное присваивание  
n = 1  
n += 1  
n
```

```
: 2
```

```
: # присваивание кортежей (позиционное)  
a, b = 1, 2  
print(f"a = {a}, b = {b}")
```

```
a = 1, b = 2
```

```
: # присваивание последовательностей  
a, b, c, d = [1, 2, 3, 4]  
print(f"a = {a}, b = {b}, c = {c}, d = {d}")
```

```
a = 1, b = 2, c = 3, d = 4
```

```
: # расширенная операция распаковки последовательности  
a, b, *c = [1, 2, 3, 4]  
print(f"a = {a}, b = {b}, c = {c}")
```

```
a = 1, b = 2, c = [3, 4]
```

Формы операторов присваивания

```
# если нам нужны только первые значения, то обычно остальное записывается в переменную _  
a, b, *_ = (1, 2, 3, [4, 5])  
print(f"a = {a}, b = {b}, _ = {_}")
```

```
a = 1, b = 2, _ = [3, [4, 5]]
```

```
# групповое присвоение одного значения  
a = b = [1, 2, 3]  
print(f"a = {a}, b = {b}")
```

```
a = [1, 2, 3], b = [1, 2, 3]
```

```
# обмен значениями  
a = 1  
b = 2  
a, b = b, a  
print(f"a = {a}, b = {b}")
```

```
a = 2, b = 1
```

2

Функции

Функция

Что такое объект в python?

- Объект в python это все что угодно, то есть все в python является объектом, например, число, строка, список, функция и тд

Что такое функция?

- Функция - объект, принимающий аргументы и возвращающий значения, определяется при помощи ключевого слова **def**

```
: def say_hello():  
    print("Hello world!")  
say_hello()
```

Hello world!

Виды аргументов

Аргументы

```
def say_welcome_message(name):  
    return f"Hello {name}"  
  
message = say_welcome_message("Mary")  
print(message)
```

Hello Mary

Аргументы по умолчанию

```
def say_welcome_message(name, hi_message = "Hi"):  
    return f"{hi_message} {name}"  
  
message = say_welcome_message("Mary")  
print(message)
```

Hi Mary

```
message = say_welcome_message("Mary", "Hello")  
print(message)
```

Hello Mary

Виды аргументов

Аргументы по умолчанию
нужны для того чтобы мы могли
определить какое-то дефолтное
значение и не передавать его
каждый раз

Аргументы

```
def say_welcome_message(name):  
    return f"Hello {name}"  
  
message = say_welcome_message("Mary")  
print(message)
```

Hello Mary

Аргументы по умолчанию

```
def say_welcome_message(name, hi_message = "Hi"):  
    return f"{hi_message} {name}"  
  
message = say_welcome_message("Mary")  
print(message)
```

Hi Mary

```
message = say_welcome_message("Mary", "Hello")  
print(message)
```

Hello Mary

Виды аргументов

Именованные аргументы

нужны для того чтобы не приходилось учитывать порядок передачи аргументов в функцию

```
message = say_welcome_message(hi_message="hi", name="Mary")  
print(message)
```

hi Mary

Сбор аргументов в кортеж

```
def func(*args):  
    print(args)
```

```
func(1, 2, 3, 4, 5)
```

(1, 2, 3, 4, 5)

Сбор аргументов в словарь

```
def func(**kwargs):  
    print(kwargs)
```

```
func(a=1, b=2, c=3)
```

{'a': 1, 'b': 2, 'c': 3}

Обобщенное определение функции

```
def func(*args, **kwargs):  
    print(f"args = {args}")  
    print(f"kwargs = {kwargs}")
```

```
func(1, 2, 3, name="Mary", age=23)
```

args = (1, 2, 3)

kwargs = {'name': 'Mary', 'age': 23}

Return

Что может возвращать функция

- ничего (нет ключевого слова return)
- любой тип данных (число, строку, словарь, другую функцию)
- несколько значений

```
# несколько значений  
def func():  
    a = 1  
    b = 2  
    c = 3  
    return a, b, c
```

```
func()
```

```
(1, 2, 3)
```

```
# ничего  
def func():  
    print("Hello")
```

```
# значение  
def func():  
    return 1 # тут может быть словарь, список и тд, т е любой тип данных
```

Область видимости - LEGB

- L - local - локальный (внутри функций)
- E - enclosed - внешний (в пределах внешних функций)
- G - global - глобальный (в пределах модуля)
- B - built-in - встроенный (print, len, max...)

```
scope = "global"

def func():
    scope = "enclosed"

    def inner_func():
        scope = "local" # заканчивает поиск там где находит переменную
        print(scope)

    inner_func()

func()

local
```

Ключевое слово `nonlocal`

- для того чтобы обновить значение переменной выше по scope

```
: scope = "global"

def func():
    scope = "enclosed"

    def inner_func():
        nonlocal scope # должна существовать во внешнем scope
        scope = "new value"
        print(f"inner_func: {scope}")

    inner_func()
    print(f"func: {scope}")

func()
print(f"global: {scope}")
```

```
inner_func: new value
func: new value
global: global
```

Ключевое слово global

- для того чтобы обновить значение глобальной переменной

Важно: лучше избегать изменения внешних переменных при помощи nonlocal или global

```
scope = "global"

def func():
    scope = "enclosed"

    def inner_func():
        global scope
        scope = "new value"
        print(f"inner_func: {scope}")

    inner_func()
    print(f"func: {scope}")

func()
print(f"global: {scope}")
```

```
inner_func: new value
func: enclosed
global: new value
```

3

Lambda функции

Lambda

Зачем?

- анонимная функция (не имеет названия)
- однострочная
- используется (чаще всего) 1 раз

```
def sum_up(a, b):  
    return a + b
```

```
# может быть сокращена до  
sum_up = lambda a, b: a + b
```

```
sum_up(1, 2)
```


Lambda

Когда использовать?

- в функциях типа `sort`, `map`, `sum`, `max`
- когда она не содержит в себе сложной логики
- когда ее не нужно документировать

```
list_of_tuples = [('IT_VLAN', 320), ('Mngmt_VLAN', 99), ('User_VLAN', 1010), ('DB_VLAN', 11)]  
sorted(list_of_tuples, key=lambda x: x[1])
```

```
[('DB_VLAN', 11), ('Mngmt_VLAN', 99), ('IT_VLAN', 320), ('User_VLAN', 1010)]
```

3

Домашнее задание

Домашнее задание (часть 1)

Часть 1. Работа с функциями:

1. Напишите функцию для проверки на то, является ли строка палиндромом. Палиндром — это слово или фраза, которые одинаково читаются слева направо и справа налево
2. Посчитайте, сколько раз каждый символ встречается в строке (можно использовать словарь для хранения символов и кол-ва раз, которое они встречаются в тексте)
3. Написать функцию, которая принимает в себя заранее неизвестное количество списков и возвращает объединенный список (т е `list_1 = [1, 2, 3]`, `list_2 = [4, 5, 6]`, `result = [1, 2, 3, 4, 5, 6]`)
4. Написать функцию, которая принимает в себя словарь и меняет местами ключи и значения в нем

Домашнее задание (часть 2)

Часть 1. Работа с функциями:

3. Написать лямбду, которая определяет, четное число или нет

Часть 2. (необязательное):

1. Написать функцию, которая приводит snake_case в CamelCase (то есть получили на вход строку str_snake_case, должны преобразовать ее в StrSnakeCase). При этом, если строка не соответствует snake_case (содержит что-то кроме _ и букв в нижнем регистре), то вывести сообщение об ошибке
2. Можно доработать интересные вам задачи из файла, где демонстрация кода, которые мы не успели прорешать

The background is a vibrant yellow color, covered with a dense, repeating pattern of various geometric shapes. These shapes include circles, squares, triangles, and lines, some of which are filled with a fine grid pattern. The shapes are scattered across the entire surface, creating a dynamic and modern aesthetic.

Спасибо за внимание!