

21. Объявление и вызов функции в Python. Параметры функции со значением по умолчанию и комментирование функции. Получение информации о функции. Способы передачи параметров при вызове функции.

Общий синтаксис объявления метода в Python:
def название(аргументы):

Функция в Python объявляется с помощью def:

```
def example_f(a):  
    return a**2
```

#Вызов

```
example_f(2)
```

Вызов происходит из основного main.

Параметры функции по умолчанию задаются там же (в параметрах) через:

```
def example_f(a=2):  
    return a**2
```

#Вызов

```
example_f()
```

Тогда можно вызывать ее без аргументов и она будет отдавать значение по умолчанию.

Комментировать то, что выполняет функция по стандарту PEP8 следует в самой функции:

```

Users > georgiydemo > Desktop > check.py > ...
1  def example_f(a=2):
2      """
3      Метод возведения в степень числа
4      """
5
6      return a**2
7
8  #Вызов
9  if __name__ == "__main__":
10     for i in range(5):
11         print(example_f(i))

```

Также функция может принимать произвольное количество параметров:

```

# распаковывание последовательности в списке параметров функции:
def all_summa(*t):
    """Функция принимает произвольное количество параметров"""
    res = 0
    for i in t:
        res += i
    return res

```

Для получения информации о функции необходимо воспользоваться встроенными полями для объекта def:

```

def example_f(a=2):
    """
    Метод возведения в степень числа
    """

    return a**2

print(example_f.__doc__)
#Метод возведения в степень числа

```

Также в интерактивном режиме запуска Python или в Jupyter Notebook можно воспользоваться командой `help`, она выдаст всю информацию о модуле или методе:

```
Users > georgiydemo > Desktop > 🐘 check.py > ...  
1  def multi_example(a=1, b=2, c=4):  
2      |      return a+b+c  
3  
4  if __name__ == "__main__":  
5      |      print(help(multi_example))
```

Способы передачи параметров при вызове функции.


Передавать параметры функции можно тремя способами.
Во-первых, простым перечислением аргументов:

```
Users > georgiydemo > Desktop > 🐘 check.py > ...  
1  def multi_example(a, b, c):  
2      |      return a+b+c  
3  
4  if __name__ == "__main__":  
5      |      result = multi_example(1,2,4)
```

Во-вторых, с помощью *и list

```
Users > georgiydemo > Desktop > 🐘 check.py > ...  
1  def multi_example(a, b, c):  
2      |      return a+b+c  
3  
4  if __name__ == "__main__":  
5      |      l = [1,2,4]  
6      |      result = multi_example(*l)
```

В-третьих, с помощью ** и dict

Users > georgiydemo > Desktop >  check.py > ...

```
1 def multi_example(a, b, c):
2     return a+b+c
3
4 if __name__ == "__main__":
5     this_dict = {"a": 1, "b" : 2, "c" : 4}
6     result = multi_example(**this_dict)
```

Если есть значения по-умолчанию для функции, то можно и ничего не передавать совсем

Users > georgiydemo > Desktop >  check.py > ...

```
1 def multi_example(a=1, b=2, c=4):
2     return a+b+c
3
4 if __name__ == "__main__":
5     result = multi_example()
```

Или предавать только часть аргументов, указывая имя аргумента:

Users > georgiydemo > Desktop >  check.py > ...

```
1 def multi_example(a=1, b=2, c=4):
2     return a+b+c
3
4 if __name__ == "__main__":
5     result = multi_example(c=5)
6     print(result) #8
```