

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе №13 по дисциплине:
основы программной инженерии**

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2021 г.

Ход работы:

1. Позиционные и именованные аргументы

```
def print_these(a, b, c):  
    print(a, "is stored in a")  
    print(b, "is stored in b")  
    print(c, "is stored in c")  
  
print_these(1, 2, 3)
```

print_these()

modul ×

C:\Users\79616\anaconda3\python.exe D:/
1 is stored in a
2 is stored in b
3 is stored in c

2. Результат неверного вывода команд

```
def print_these(a, b, c):  
    print(a, "is stored in a")  
    print(b, "is stored in b")  
    print(c, "is stored in c")  
  
print_these(1, 2)
```

modul ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/13/modul.py
Traceback (most recent call last):
 File "D:/УЧЕБА/ОПИ/13/modul.py", line 6, in <module>
 print_these(1, 2)
TypeError: print_these() missing 1 required positional argument: 'c'

3. Применение опциональных параметров

```
def print_these(a, b, c=None):  
    print(a, "is stored in a")  
    print(b, "is stored in b")  
    print(c, "is stored in c")  
  
print_these(1, 2)
```

print_these()

modul x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ

1 is stored in a
2 is stored in b
None is stored in c

4. Установление трех опциональных параметров

```
def print_these(a=None, b=None, c=None):  
    print(a, "is stored in a")  
    print(b, "is stored in b")  
    print(c, "is stored in c")  
  
print_these(c=3, a=1)
```

modul x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/1

1 is stored in a
None is stored in b
3 is stored in c

5. Применение оператора «звездочка»

```
a = [1, 2, 3]
b = [*a, 4, 5, 6]

print(b)
```

modul ×

C:\Users\79616\anaconda3\python.exe
[1, 2, 3, 4, 5, 6]

6. Использование функций *args и *kwargs

```
def print_scores(student, *scores):
    print(f"Student Name: {student}")
    for score in scores:
        print(score)

print_scores("Jonathan", 100, 95, 88, 92, 99)
```

modul ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/13/modul.py
Student Name: Jonathan
100
95
88
92
99

```
def print_pet_names(owner, **pets):  
    print(f"Owner Name: {owner}")  
    for pet, name in pets.items():  
        print(f"{pet}: {name}")  
  
print_pet_names(  
    "Jonathan",  
    dog="Brock", fish=["Larry", "Curly", "Moe"],  
    turtle="Shelldon"  
)
```

modul x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/13/modul.py

Owner Name: Jonathan

dog: Brock

fish: ['Larry', 'Curly', 'Moe']

turtle: Shelldon

Выполнение примера 1:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()

        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2

    else:
        return None

if __name__ == "__main__":
    print(median())
    print(median(3, 7, 1, 6, 9))
    print(median(1, 5, 8, 4, 3, 9))
```

modul ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/13/modul.py

None

6.0

4.5

Выполнение индивидуального задания (вариант 13)

Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Сумму аргументов, расположенных после минимального аргумента.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def min_sum(*args):
    if args:
        min_index = args.index(min(args))
        summa = 0
        for i in range(min_index + 1, len(args)):
            summa += args[i]
        return summa
    else:
        return None

if __name__ == '__main__':
    print(f"Сумма равна: {min_sum(5, 2, 3, 1, 5, 3, 4, 6)}")
```



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def min_sum(*args):
    if args:
        min_index = args.index(min(args))
        summa = 0
        for i in range(min_index + 1, len(args)):
            summa += args[i]
        return summa
    else:
        return None

if __name__ == '__main__':
    print(f"Сумма равна: {min_sum(5, 2, 3, 1, 5, 3, 4, 6)}")
```

ind x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/13/ind.py

Сумма равна: 18

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Это аргументы, передаваемые в вызов в определённой последовательности (на определённых позициях), без указания их имён. Элементы объектов, поддерживающих итерирование, могут использоваться в качестве позиционных аргументов, если их распаковать при помощи `*`.

2. Какие аргументы называются именованными в Python?

Это аргументы, передаваемые в вызов при помощи имени (идентификатора), либо словаря с его распаковкой при помощи `**`.

3. Для чего используется оператор `*`?

Этот оператор позволяет «распаковывать» объекты, внутри которых хранятся некие элементы.

4. Каково назначение конструкций `*args` и `**kwargs`?

`*args` используется для передачи произвольного числа именованных аргументов функции.

`**kwargs` позволяет передавать произвольное число именованных аргументов в функцию.