

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №13
дисциплины «Программирование на Python»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: Функции с переменным числом параметров в Python

Цель: приобретение навыков по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Проработаны примеры из лабораторной работы

Пример 1. Разработать функцию для определения медианы значений аргументов функции. Если функции передается пустой список аргументов, то она должна возвращать значение None.

Результат работы программы:

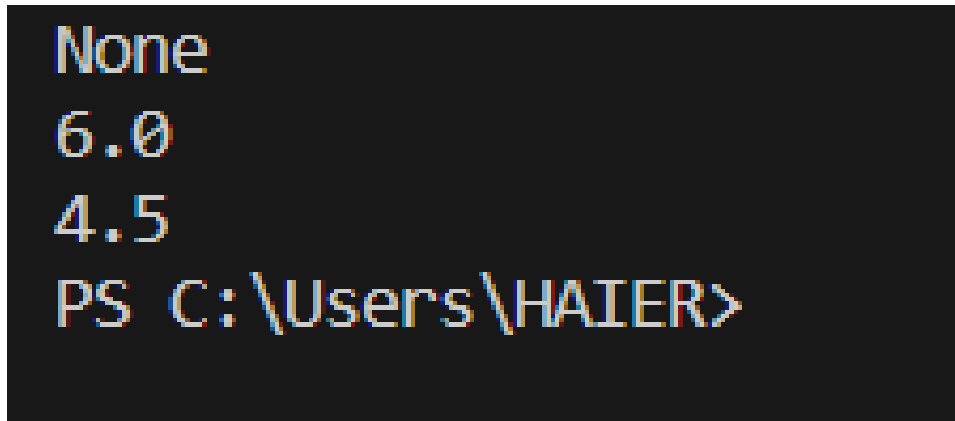


Рисунок 1 – Результат работы программы

Код программы:

```
#!/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))
```

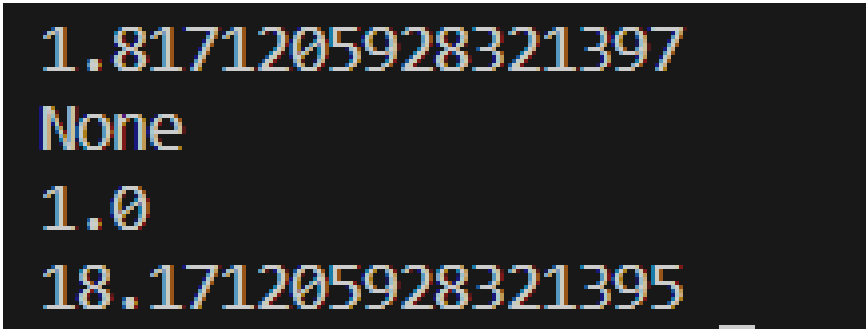
2. Решены задачи из лабораторной работы

1) Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов.

Код программы:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
def geometric_mean(*args):  
    if args:  
        values = [float(arg) for arg in args]  
  
        product = 1  
        for item in values:  
            product *= item  
  
        return product**(1/len(values))  
    else:  
        return None  
  
if __name__ == "__main__":  
    print(geometric_mean(1,2,3))  
    print(geometric_mean())  
    print(geometric_mean(1))  
    print(geometric_mean(10,20,30))
```

Результат работы программы:



```
1.8171205928321397  
None  
1.0  
18.171205928321395
```

Рисунок 2 – Результат работы программы

2) Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов. Если функции передается пустой список аргументов, то она должна возвращать значение None.

Код программы:

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

def garmonic_mean(*args):
    if args:
        values = [float(arg) for arg in args]

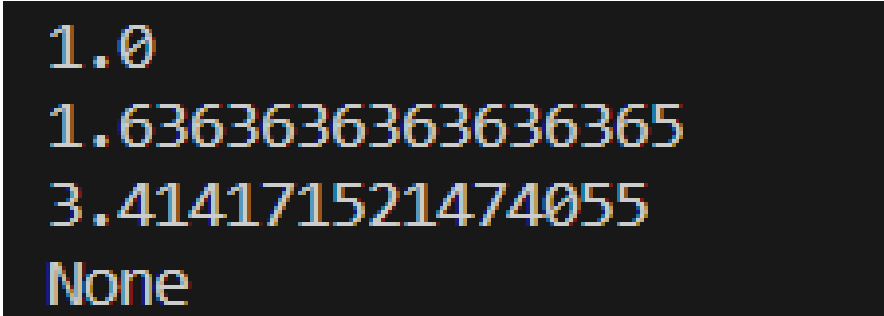
        count = len(values)
        denominator = 0

        for item in values:
            denominator += 1/item

        return count/denominator
    else:
        return None

if __name__ == "__main__":
    print(garmonic_mean(1))
    print(garmonic_mean(1,2,3))
    print(garmonic_mean(1,2,3,4,5,6,7,8,9,10))
    print(garmonic_mean())
```

Результат работы программы:



The screenshot shows the output of the program on a black background with yellow text. The output consists of four lines: '1.0', '1.6363636363636365', '3.414171521474055', and 'None'.

Рисунок 3 – Результат работы программы

3) Самостоятельно подберите или придумайте задачу с переменным числом именованных аргументов. Приведите решение этой задачи

Написать функцию, переводящую аргументы в тип str, проводящую их конкатенацию и возвращающую получившуюся строку. Если функции передается пустой список аргументов, то она должна возвращать значение None.

Код программы:

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

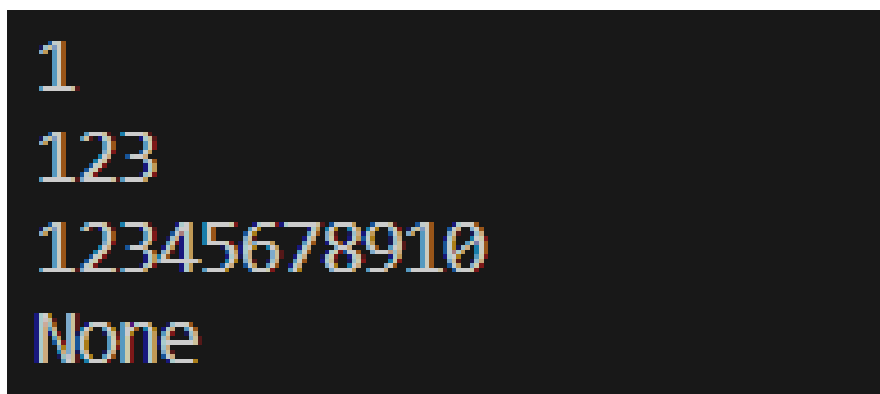
def concatenation(*args):
    if args:
        values = [str(arg) for arg in args]

        string = ""
        for item in values:
            string += item

        return string
    else:
        return None

if __name__ == "__main__":
    print(concatenation(1))
    print(concatenation(1,2,3))
    print(concatenation(1,2,3,4,5,6,7,8,9,10))
    print(concatenation())
```

Результат работы программы:



```
1
123
12345678910
None
```

Рисунок 4 – Результат работы программы

3. Выполнено индивидуальное задание

Вариант 29 (11)

Напишите функцию, принимающую произвольное количество аргументов, и возвращающую сумму аргументов, расположенных после первого положительного аргумента. Если функции передается пустой список аргументов, то она должна возвращать значение None.

Код программы:

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

def f(*args):
    if args:
        sum = 0
        test_positive = False

        for item in args:
            if float(item) > 0 or test_positive == True:
                if test_positive == False:
                    test_positive = True
                    sum -= float(item)

                sum += float(item)

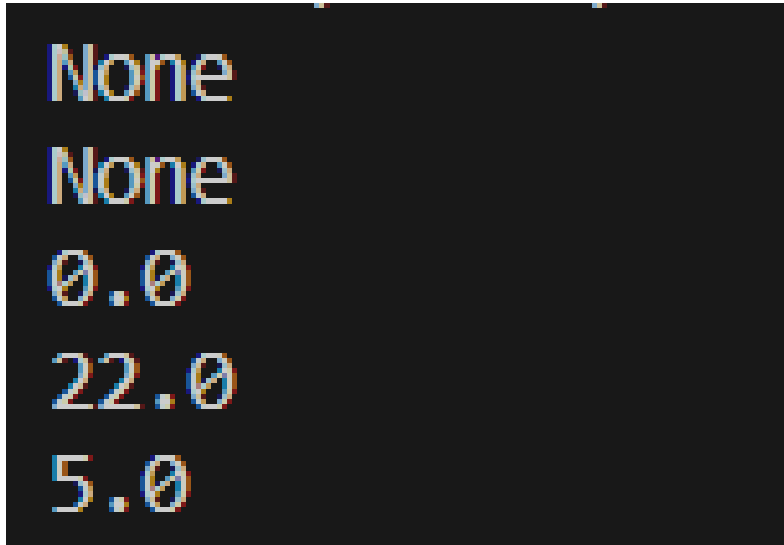
        #Если нет положительных аргументов, то функция возвращает None
        if test_positive == False:
            return None

    else:
        return sum

else:
    return None

if __name__ == "__main__":
    print(f(-1, -2, -3))
    print(f())
    print(f(-1, -2, 3))
    print(f(-1, -2, 3, 4, 5, 6, 7))
    print(f(1, 2, 3))
```

Результат работы программы:



```
None
None
0.0
22.0
5.0
```

Рисунок 5 – Результат работы программы

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы – аргументы, значение которых передаётся переменным путём сопоставления позиций переданных аргументов и параметров.

2. Какие аргументы называются именованными в Python?

Именованными называются аргументы, значение которых назначается по именам при вызове функции, а их порядок может быть любым. В результате функция будет всегда выводить одно и то же значение независимо от позиций переданных ей аргументов.

3. Для чего используется оператор *?

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

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

*args — это сокращение от «arguments» (аргументы), а **kwargs — сокращение от «keyword arguments» (именованные аргументы).

Первая конструкция позволяет «распаковывать» списки обычных аргументов, а вторая позволяет делать то же самое с именованными аргументами, создавая словарь.

Вывод

В ходе выполнения работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.