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 9

Выполнил:
Дудкин Константин Александрович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»
направление «Программное
обеспечение средств вычислительной
техники и автоматизированных
систем»,
очная форма обучения

_____
(подпись)

Руководитель практики:
Кандидат технических наук, доцент
кафедры инфокомуникаций, доцент
Воронкин Роман Александрович

_____
(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. Проработал пример: Разработать функцию для определения медианы значений аргументов функции. Если в функции передается пустой список аргументов, то она должна возвращать значение None

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

3 usages  new *
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( *args: 3, 7, 1, 6, 9))
    print(median( *args: 1, 5, 8, 4, 3, 9))
```

Рисунок 1. Код программы примера

```
/usr/bin/python3.11 /home/code_ra1der/git/Python_LW13/Python Programs/Example.py
None
6.0
4.5

Process finished with exit code 0
```

Рисунок 2. Результат работы программы

2. Выполнил основное задание №1: Написать функцию, вычисляющую среднее арифметическое своих аргументов a1, a2, …, an

$$G = \sqrt[n]{\prod_{k=1}^{n} a_k}.$$

Если функции передается пустой список аргументов, то она должна возвращать значение None

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

1 usage  new *
def function(*args):
    if args:
        nums = list(map(float, args))
        result = 1.0
        for num in nums:
            result *= num
        return round(pow(result, 1 / len(nums)), 4)
    else:
        return None


if __name__ == "__main__":
    input_data = input("Введите числа через пробел: ").split()
    result = function(*input_data)
    print("Результат:", result)
```

Рисунок 3. Код программы задания

```
/usr/bin/python3.11 /home/code_ra1der/git/Python_LW13/Python Programs/Task1.py
Введите числа через пробел: 2 7 9 3 6 8
Результат: 5.1261

Process finished with exit code 0
```

Рисунок 4. Результат работы программы

3. Выполнил основное задание №2: Написать функцию, вычисляющую среднее гармоническое своих аргументов a1, a2, …, an

$$\frac{n}{H} = \sum_{k=1}^{n} \frac{1}{a_k}.$$

Если функции передается пустой список аргументов, то она должна возвращать значение None

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

1 usage  new *
def function(*args):
    if args:
        nums = list(map(float, args))
        function_sum = sum(1 / num for num in nums)
        return round(len(nums) / function_sum, 4)
    else:
        return None


if __name__ == "__main__":
    input_data = input("Введите числа через пробел: ").split()
    result = function(*input_data)
    print("Результат:", result)
```

Рисунок 5. Код программы задачи

```
/usr/bin/python3.11 /home/code_ra1der/git/Python_LW13/Python Programs/Task2.py
Введите числа через пробел: 3 6 1 7
Результат: 2.4348

Process finished with exit code 0
```

Рисунок 6. Результат выполнения задачи

4. Выполнил индивидуальное задание: Написать функцию, принимающую произвольное кол-во аргументов и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение None. В процессе решения не использовать преобразования конструкции *args в список или иную структуру данных

Вариант 9 — Сумму модулей аргументов, расположенных после первого отрицательного аргумента

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

1 usage  new *
def function(*args):
    negative_num = False
    result = 0
    if args:
        for num in args:
            num = int(num)
            if negative_num == True:
                result += abs(num)
            elif num < 0 and negative_num == False:
                negative_num = True
        if negative_num == False:
            return 0
        return result
    else:
        return None

if __name__ == "__main__":
    input_args = input("Введите значения (для корректного выполнения добавьте отрицательное число): ").split()

    print("Результат:", function(*input_args))
```

Рисунок 7. Код программы задачи

```
/usr/bin/python3.11 /home/code_ra1der/git/Python_LW13/Python Programs/Individual.py
Введите значения (для корректного выполнения добавьте отрицательное число): 1 5 2 -5 3 2 -6 1
Результат: 12

Process finished with exit code 0
```

Рисунок 8. Результат выполнения программы

Ответы на вопросы

1. В Python аргументы считаются позиционными, когда они передаются функции в том порядке, в котором они определены в самой функции. Возможно также передавать переменное количество позиционных аргументов, используя оператор * перед именем аргумента в определении функции

2. Именованными аргументами в Python называются те, которые передаются функции с указанием имени аргумента, за которым следует значение. Также, как и с позиционными аргументами, можно передавать пременное количество именованных аргументов с использованием оператора ** перед именем аргумента в определении функции

3. Оператор * в Python не только ассоциируется с операцией умножения, но также используется для «распаковки» объектов, содержащих некие элементы. Этот оператор позволяет извлекать элементы из объектов внутри определенных структур данных

4. *args позволяет передавать переменное кол-во позиционных аргументов в функцию. Объект, переданный с использованием *args, может быть распакован, позволяя вызывать функции с различным числом аргументов

**kwargs позволяет передавать переменное кол-во именованных аргументов в функцию. Аналогично *args, он позволяет распаковывать словари с аргументами, что облегчает вызов функции с различными именованными параметрами

Вывод: В ходе выполения работы были получены навыки по работе с функциями, поддерживающими переменное кол-во параметров в Python версии 3.x