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 31

Выполнил:
Репкин Александр Павлович
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. техн. наук,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Функции с переменным числом параметров в языке Python.

Цель: приобрести навыки работы с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Выполнен первый пример. В нём была создана функция для вычисления медианы значений аргументов функции. Также, было условие – при передаче функции пустого списка аргументов, возвращается значение None.

```
None
6.0
4.5
```

Рисунок 1. Полученный результат выполнения примера.

2. Выполнено задание №8. В нём требовалось создать функцию, вычисляющую среднее геометрическое переданных ей аргументов. Также, было условие – при передаче функции пустого списка аргументов, возвращается значение None.

```
Elements are: 16.78 20.46 5.76 24.76
Answer is -  14.88
None
```

Рисунок 2. Полученный результат задания №8.

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

import random


def calculating_multiply(*args):
    """
    Вычисление среднего геометрического полученных аргументов.
    Если аргументов нет, то выводится None.
    """

    if args:
        multiply = 1.0
        for value in args:
            multiply *= value
        return round(pow(multiply, 1/len(args)), 2)
    else:
        return "None"


if __name__ == "__main__":
    elements = [round(random.uniform(1, 25), 2)
                for _ in range(random.randint(2, 7))]
    print("Elements are:", *elements)
    print("Answer is - ", calculating_multiply(*elements))
    print(calculating_multiply())
```

Рисунок 3. Полученный код задания №8.

3. Выполнено задание №9. В нём требовалось создать функцию, вычисляющую среднее гармоническое переданных ей аргументов. Также, было условие – при передаче функции пустого списка аргументов, возвращается значение None.

```
Elements are: 16.43 23.41 16.8 10.98 13.79 9.33
Answer is -  13.83
None
```

Рисунок 4. Полученный результат задания №9.

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

import random


def calculating_garmonical(*args):
    """
    Вычисление среднего гармонического полученных аргументов.
    Если аргументов нет, то выводится None.
    """

    if args:
        garmony = 0
        for value in args:
            garmony += 1 / value
        return round(len(args)/garmony, 2)
    else:
        return "None"


if __name__ == "__main__":
    elements = [round(random.uniform(1, 25), 2)
                for _ in range(random.randint(2, 7))]
    print("Elements are:", *elements)
    print("Answer is - ", calculating_garmonical(*elements))
    print(calculating_garmonical())
```

Рисунок 5. Полученный код задания №9.

4. Выполнено задание №13. В нём требовалось самостоятельно придумать функцию, принимающую и обрабатывающую переменное число именованных аргументов. Исходя из задания решено сделать функцию, выводящую определённые символы на основе полученных чисел. Также, добавлено условие – при передаче функции пустого списка аргументов, возвращается значение None.

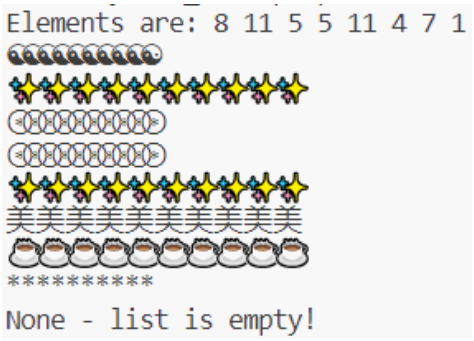

Рисунок 6. Полученный результат задания №13.

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

import random


def summ_after_min(*args):
    """Вывод рисунка, исходя из полученных чисел"""

    if args:
        patterns = {
            1: "*"*10,
            2: "☼"*10,
            3: "๑"*10,
            4: "美"*10,  # Означает "Красота", согласно словарю
            5: "⊙"*10,
            6: "⌛"*10,
            7: "🍲"*10,
            8: "☯"*10,
            9: "⛏"*10,
            10: "✿"*10,
            11: "🐈"*10
        }
        for i in args:
            print(patterns.get(i))
    else:
        print("None - list is empty!")


if __name__ == "__main__":
    elements = [random.randint(1, 11) for _ in range(random.randint(2, 30))]
    print("Elements are:", *elements)
    summ_after_min(*elements)
    summ_after_min()
```

Рисунок 7. Полученный код задания №13.

5. Выполнено индивидуальное задание. В нём требовалось написать функцию, принимающую произвольное количество аргументов, и возвращающую сумму аргументов, расположенных после минимального аргумента. Если функции передается пустой список аргументов, то она возвращает значение None. В процессе решения не использованы преобразования конструкции *args в список или иную структуру данных.

```
Elements are: 40 69 80 32 16 7 80 87
('Summ of elements after', 7, 'is', 167)
Summ of elements after 7 is 167
Result for empty list: None
```

Рисунок 8. Полученный результат индивидуального задания.

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

import random


def summ_after_min(*args):
    """
    Вычисление суммы аргументов после минимального.
    Если аргументов нет, то выводится None.
    """

    if args:
        summ = 0
        found = False
        minimal = min(args)
        for value in args:
            if found:
                summ += value
            elif minimal == value:
                found = True
        return "Summ of elements after",  minimal, "is", summ
    else:
        return "None"


if __name__ == "__main__":
    elements = [random.randint(1, 100) for _ in range(random.randint(2, 20))]
    print("Elements are:", *elements)
    print(summ_after_min(*elements))
    print(*summ_after_min(*elements))  # Экспериментирование с распаковкой
    print("Result for empty list:", summ_after_min())

```

Рисунок 9. Полученный код индивидуального задания.

6. Ответы на вопросы.

1) Какие аргументы называются позиционными в Python?

Ответ: позиционные аргументы определяются порядком, в котором они передаются функции. def smth(arg1, arg2, arg3):... smth(“a”, 235, “?!”) - пример вызова функции с позиционными аргументами.

2) Какие аргументы называются именованными в Python?

Ответ: ключевые аргументы передаются с указанием имени параметра. def smth(arg1, arg2, arg3):... smth(arg1="a", arg2=235, arg3="?!") - пример вызова функции с именованными аргументами.

3) Для чего используется оператор * ?

Ответ: оператор * позволяет “распаковывать” объекты, внутри которых хранятся некие элементы. Так, если a = [1, 2, 3], после чего добавить a в b - b =

[*a, 4, 5, 6], то благодаря оператору *, b = [1, 2, 3, 4, 5, 6], а не b = [[1, 2, 3], 4, 5, 6].

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

Ответ: *args –сокращение от "arguments" (аргументы), а **kwargs – сокращение от "keyword arguments" (именованные аргументы). Обе конструкции используются для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки работы функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.