

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 практической работе №2.10
Дисциплины «Программирование на Python»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и
вычислительная техника (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

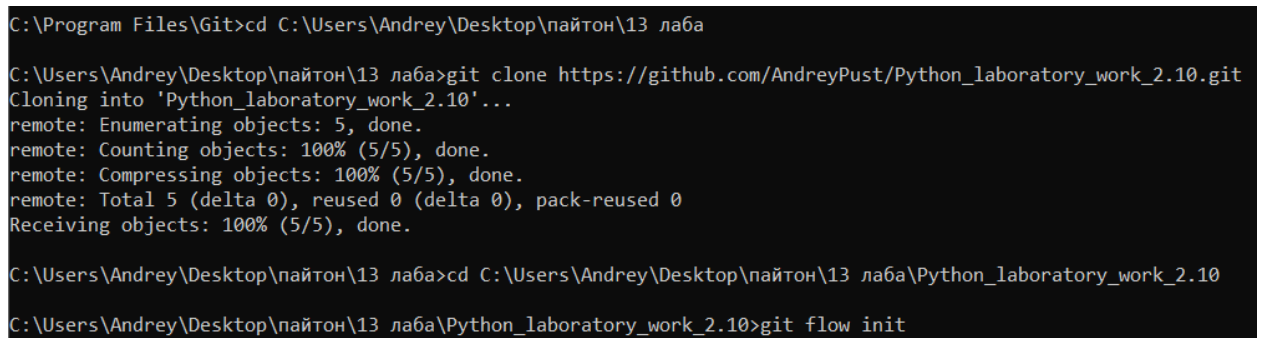
Ставрополь, 2023 г.

Тема: Функции с переменным числом параметров в Python.

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

Ход работы:

Создание общедоступного репозитория на «GitHub», клонирование репозитория, редактирование файла «.gitignore», организация репозитория согласно модели ветвления «git-flow» (рис. 1).



```
C:\Program Files\Git>cd C:\Users\Andrey\Desktop\пайтон\13 лаба
C:\Users\Andrey\Desktop\пайтон\13 лаба>git clone https://github.com/AndreyPust/Python_laboratory_work_2.10.git
Cloning into 'Python_laboratory_work_2.10'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\Andrey\Desktop\пайтон\13 лаба>cd C:\Users\Andrey\Desktop\пайтон\13 лаба\Python_laboratory_work_2.10
C:\Users\Andrey\Desktop\пайтон\13 лаба\Python_laboratory_work_2.10>git flow init
```

Рисунок 1 – Организация модели ветвления «git-flow».

Проработка примеров лабораторной работы:

Пример 1.

Необходимо разработать функцию для определения медианы значений аргументов функции. В случае, когда функции передается пустой список аргументов, функция должна возвращать значение «None».

Код программы примера №1 и результаты работы программы (рис. 2, 3).

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  3 usages  new *
6  def median(*args):
7      if args:
8          values = [float(arg) for arg in args]
9          values.sort()
10
11         n = len(values)
12         idx = n // 2
13         if n % 2:
14             return values[idx]
15         else:
16             return (values[idx - 1] + values[idx]) / 2
17     else:
18         return None
19
20 if __name__ == "__main__":
21     print(median())
22     print(median(*args: 3, 7, 1, 6, 9))
23     print(median(*args: 1, 5, 8, 4, 3, 9))
24

```

Рисунок 2 – Код программы примера №1.

```

C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\
None
6.0
4.5

Process finished with exit code 0

```

Рисунок 3 – Результаты работы программы примера №1.

Необходимо решить задачу: Необходимо разработать функцию, которая вычисляет среднее геометрическое всех переданных ей аргументов $a_1, a_2, a_3, \dots, a_n$. В случае, когда функции передается пустой список аргументов, она должна возвращать значение «None».

Код программы решения данной задачи и результаты работы программы при (рис. 4, 5).

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

def geometric_mean(*args):
    """
    Функция вычисления среднего геометрического всех переданных ей
    аргументов.
    В случае, если функции передается пустой список аргументов, она
    возвращает "None".
    """
    if args:
        composition = 1.0
        for i in args:
            composition *= i
        return round(pow(composition, 1 / len(args)), 2)
    else:
        return None

if __name__ == "__main__":
    # Необходимо разработать функцию, которая вычисляет среднее
    # геометрическое всех
    # переданных ей аргументов a1, a2, a3, ... , an. В случае, когда функции
    # передается
    # пустой список аргументов, она должна возвращать значение «None».

    print(geometric_mean())
    print(geometric_mean(1, 2, 3, 4, 5))
    print(geometric_mean(1.1, 5.6, 8.9, 4, 3, 9.1))
```

Рисунок 4 – Код программы первой задачи.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\Des
None
2.61
4.26

Process finished with exit code 0
```

Рисунок 5 – Результаты работы программы первой задачи.

Необходимо решить задачу: Необходимо написать функцию, вычисляющую среднее гармоническое переданных ей аргументов $a_1, a_2, a_3, \dots, a_n$. В случае, когда функции передается пустой список аргументов, она должна возвращать значение «None».

Код программы решения задачи №2 и результаты работы программы (рис. 6, 7).

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

def harmonic_mean(*args):
    """
    Функция вычисления среднего гармонического всех переданных ей аргументов.
    В случае, если функции передается пустой список аргументов, она
    возвращает "None".
    """
    if args:
        harmonic = 0
        for i in args:
            harmonic += 1 / i
        return round(len(args) / harmonic, 2)
    else:
        return None

if __name__ == "__main__":
    # Необходимо написать функцию, вычисляющую среднее гармоническое
    # переданных
    # ей аргументов a1, a2, a3, ... , an. В случае, когда функции передается
    # пустой
    # список аргументов, она должна возвращать значение «None».

    print(harmonic_mean())
    print(harmonic_mean(1, 2, 3, 4, 5))
    print(harmonic_mean(1.1, 5.6, 8.9, 4, 3, 9.1))
```

Рисунок 6 – Код программы второй задачи.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe"
None
2.19
3.17

Process finished with exit code 0
```

Рисунок 7 – Результаты работы программы второй задачи.

Необходимо решить задачу: необходимо самостоятельно придумать задачу, для которой необходимо написать функцию с переменным числом именованных аргументов.

Найти среднее арифметическое всех переданных функции аргументов, найти и вывести информацию о среднем отклонении этих аргументов в квадрате и вычислить среднеквадратическое отклонение от среднего значения (вычислить случайную ошибку в измерениях).

Код программы решения задачи №3 и результаты работы программы (рис. 8, 9).

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

import math

def accidental_error(*args):
    """
    Функция вычисления среднего арифметического всех переданных ей
    аргументов,
    вычисления отклонения этих аргументов от среднеарифметического в квадрате
    и
    расчета среднеквадратического отклонения от среднего значения.
    В случае, если функции передается пустой список аргументов, она
    возвращает "None".
    """
    if args:
        # Найдем среднеарифметическое всех аргументов.
        arithmetic_mean = 0
        for i in args:
            arithmetic_mean += i
        arithmetic_mean /= len(args)
        print("Среднеарифметическое аргументов =", arithmetic_mean)

        # Найдем и выведем отклонение от среднеарифметического в квадрате.
        for i in args:
            i = (i - arithmetic_mean)**2
            print("Отклонение аргумента от среднеарифметического в квадрате
            =", i)

        # Найдем среднеквадратическое отклонение от среднего значения
        (случайную ошибку)
        accidental = 0
        for i in args:
            accidental += i
        return math.sqrt(accidental/(len(args) * (len(args) - 1)))

    else:
        return None

if __name__ == "__main__":
    # Необходимо самостоятельно придумать задачу, для которой необходимо
    написать
    # функцию с переменным числом именованных аргументов.

    # Найти среднее арифметическое всех переданных функции аргументов, найти
    и вывести
    # информацию о среднем отклонении этих аргументов в квадрате и вычислить
    # среднеквадратическое отклонение от среднего значения.

    # В случае, когда функции передается пустой список аргументов, она должна
    # возвращать значение «None».

    print("Среднеквадратическое отклонение от среднего значения (случайная
    ошибка) =",
          accidental_error(), "\n")
    print("Среднеквадратическое отклонение от среднего значения (случайная
    ошибка) =",
          accidental_error(1, 2, 3, 4, 5), "\n")
```

```
print("Среднеквадратическое отклонение от среднего значения (случайная
ошибка) =",
      accidental_error(1.1, 5.6, 8.9, 4, 3, 9.1))
```

Рисунок 8 – Код программы произвольной задачи.

```
Среднеквадратическое отклонение от среднего значения (случайная ошибка) = None

Среднеарифметическое аргументов = 3.0
Отклонение аргумента от среднеарифметического в квадрате = 4.0
Отклонение аргумента от среднеарифметического в квадрате = 1.0
Отклонение аргумента от среднеарифметического в квадрате = 0.0
Отклонение аргумента от среднеарифметического в квадрате = 1.0
Отклонение аргумента от среднеарифметического в квадрате = 4.0
Среднеквадратическое отклонение от среднего значения (случайная ошибка) = 0.8660254037844386

Среднеарифметическое аргументов = 5.2833333333333334
Отклонение аргумента от среднеарифметического в квадрате = 17.500277777777778
Отклонение аргумента от среднеарифметического в квадрате = 0.10027777777777706
Отклонение аргумента от среднеарифметического в квадрате = 13.080277777777775
Отклонение аргумента от среднеарифметического в квадрате = 1.6469444444444463
Отклонение аргумента от среднеарифметического в квадрате = 5.213611111111114
Отклонение аргумента от среднеарифметического в квадрате = 14.566944444444436
Среднеквадратическое отклонение от среднего значения (случайная ошибка) = 1.0279429296739517
```

Рисунок 9 – Результаты работы программы произвольной задачи.

Выполнение индивидуального задания:

Необходимо написать функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение «None». Необходимо найти сумму аргументов, расположенных между первым и вторым положительными аргументами (Вариант 26 (8)).

Код программы индивидуального задания и результаты работы программы (рис. 10, 11).

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

def first_sum_second(*args):
    """
    Функция вычисления суммы переданных ей аргументов, расположенных между
    первым и
    вторым положительными аргументами.
    В случае, если функции передается пустой список аргументов, она
    возвращает "None".
    """
    if args:
```

```

positive_count = 0
sum_args = float(0)
count = 0

for arg in args:
    if arg > 0:
        positive_count += 1

    if positive_count == 1 and args[count+1] > 0:
        break

    if positive_count == 1:
        sum_args += args[count+1]

    count += 1

return sum_args

else:
    return None

if __name__ == "__main__":
    # Необходимо найти сумму аргументов, расположенных между первым и вторым
    # положительными аргументами (Вариант 26 (8)). Если функции передается
    # пустой список аргументов, то она должна возвращать значение «None».

    print(first_sum_second())
    print(first_sum_second(10, -10, -20, -30, -50, 1))
    print(first_sum_second(-1, 10, -273, -3749, -372.1, -40, 1, 23, 50))

```

Рисунок 10 – Код программы индивидуального задания.

```

C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\l
None
-110.0
-4434.1

Process finished with exit code 0

```

Рисунок 11 – Результаты работы программы индивидуального задания.

Ответы на контрольные вопросы:

1. Позиционные аргументы – это аргументы, которые определяются порядком, в котором они передаются функции. Позиция, в которой находятся аргументы при их передаче функции важна для позиционных аргументов.
2. Именованные аргументы передаются с указанием имени параметра. `def example(arg1, arg2, arg3)` или `example(arg1=11.1, arg2=11, arg3='a')` – пример вызова функции с именованными аргументами.

3. Оператор «*» помимо того, что он может указывать на операцию умножения, также он позволяет «распаковывать» объекты, внутри которых хранятся некие элементы, например при передачи неопределенного количества переменных функции.

4. «*args» –сокращение от «arguments» (аргументы), а «**kwargs» – сокращение от «keyword arguments» (именованные аргументы). Обе конструкции используются для распаковки аргументов соответствующего типа (аргументов типа и именованных аргументов), позволяя вызывать функции со списком аргументов переменной длины.

Вывод: в ходе лабораторной работы была изучена работа с функциями с переменным числом параметров, были более подробно изучены понятия позиционных и именованных аргументов, а также способы передачи переменного числа параметров функции и работа с этими параметрами, а также способы проверки на наличие переданных параметров функции.