

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Функции с переменным числом параметров в Python»

ОТЧЕТ
по лабораторной работе №13
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

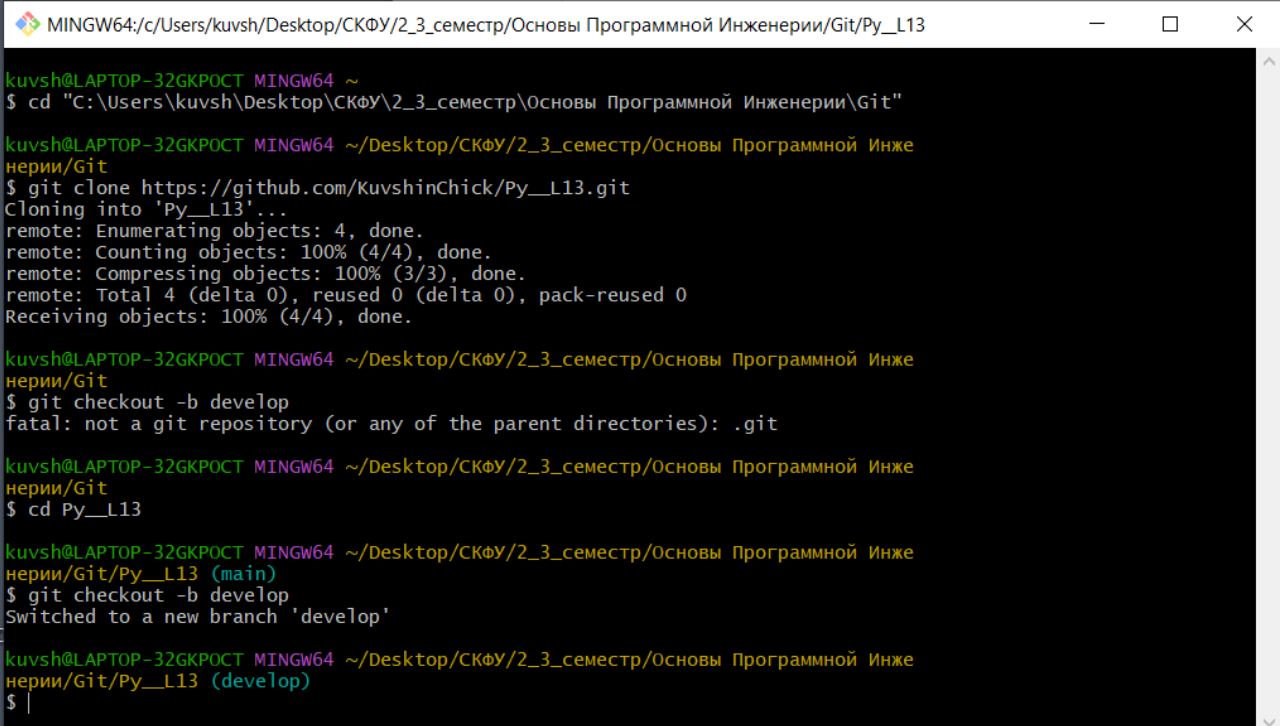
Ставрополь, 2022 г.

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

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L13.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L13
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L13.git
Cloning into 'Py__L13'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git checkout -b develop
fatal: not a git repository (or any of the parent directories): .git
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L13
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L13 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L13 (develop)
$
```

Рисунок 13.1 – Клонирование репозитория и создание ветки develop

```

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L13 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L13 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L13 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L13 (develop)
$ git commit -m "modified .gitignore & readme"
[develop deal210] modified .gitignore & readme
2 files changed, 132 insertions(+), 1 deletion(-)
create mode 100644 .gitignore

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L13 (develop)
$ |

```

Рисунок 13.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы.
8. Решить поставленную задачу: написать функцию, вычисляющую

среднее геометрическое своих аргументов a_1, a_2, \dots, a_n

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

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

```

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

import math

def geome_mean(*args):
    if args:
        nums = [int(arg) for arg in args]
        return math.prod(nums) ** (1/len(nums))
    else:
        return None

```

```

if __name__ == '__main__':
    arguments = [i for i in input("Enter the arguments: ").split()]
    print(f"The geometric mean of these arguments is:
{geome_mean(*arguments)}")

```

Рисунок 13.3 – Код программы

```

task_8 (1) x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программирования"
Enter the arguments: 3 3 3
The geometric mean of these arguments is: 3.0
Process finished with exit code 0

```

```

task_8 (1) x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программирования"
Enter the arguments:
The geometric mean of these arguments is: None
Process finished with exit code 0

```

Рисунок 13.4 – Результат работы программы

9. Решить поставленную задачу: написать функцию, вычисляющую

среднее гармоническое своих аргументов a_1, a_2, \dots, a_n

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

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

```

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

def harmon_mean(*args):
    if args:
        # Сразу получаю перевернутые дроби 1/n
        nums = [1/int(arg) for arg in args]

```

```

        return len(nums)/sum(nums)
    else:
        return None

if __name__ == '__main__':
    arguments = [i for i in input("Enter the arguments: ").split()]
    print(f"The harmonic mean of these arguments is:
{harmon_mean(*arguments)}")

```

Рисунок 13.5 – Код программы

```

task_9 x
"C:\Users\kuvsh\Desktop\СКОУ\2_3_семестр\Основы Программной Ин
Enter the arguments: 2 3 5
The harmonic mean of these arguments is: 2.9032258064516134
Process finished with exit code 0

```

```

task_9 x
"C:\Users\kuvsh\Desktop\СКОУ\2_3_семестр\Основы
Enter the arguments:
The harmonic mean of these arguments is: None
Process finished with exit code 0

```

Рисунок 13.6 – Результат работы программы

10. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

11. Зафиксируйте изменения в репозитории.

12. Решите индивидуальное задание согласно своего варианта.

15. Сумму модулей аргументов, расположенных после минимального по модулю аргумента.

```

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

def sum_of_modules(*args):
    if args:
        nums = [abs(int(arg)) for arg in args]
        return sum(nums[nums.index(min(nums)) + 1::])
    else:

```

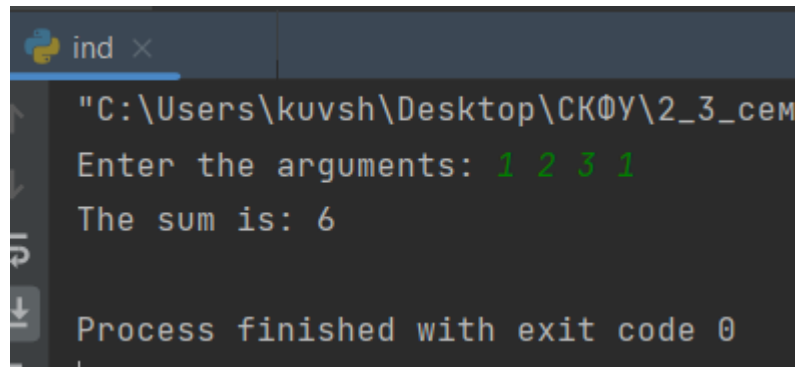
```

return None

if __name__ == '__main__':
    arguments = [i for i in input("Enter the arguments: ").split()]
    print(f"The sum is: {sum_of_modules(*arguments)}")

```

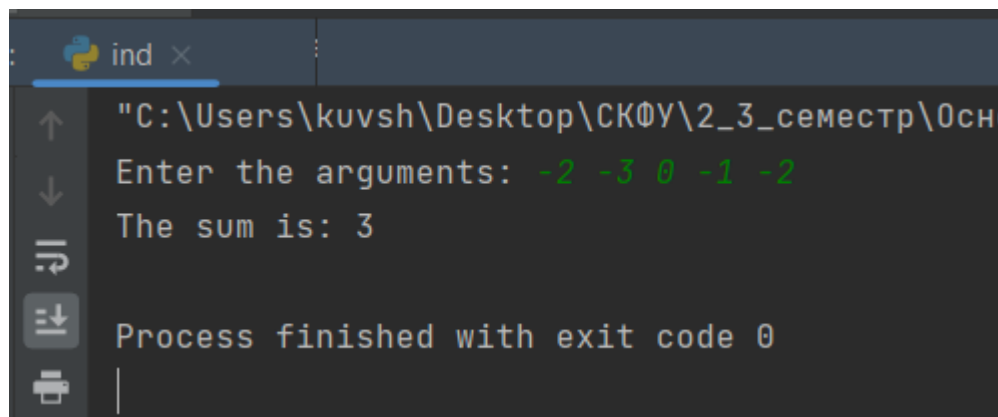
Рисунок 13.7 – Код программы



```

"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\0сн
Enter the arguments: 1 2 3 1
The sum is: 6
Process finished with exit code 0

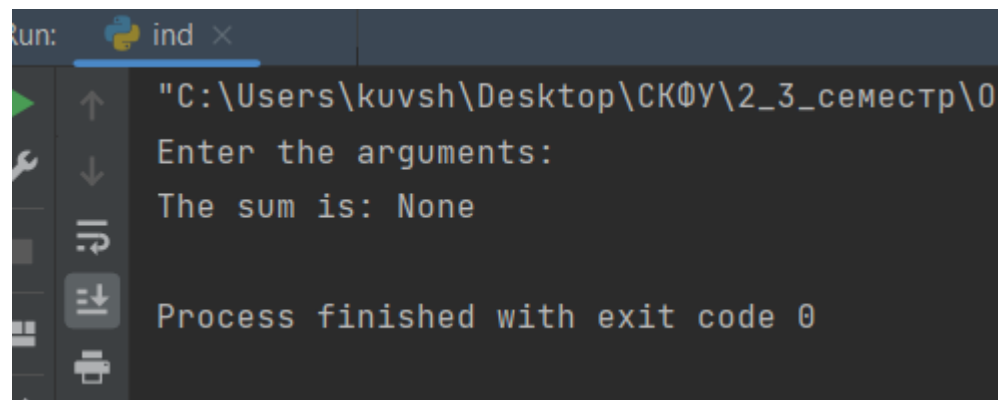
```



```

"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\0сн
Enter the arguments: -2 -3 0 -1 -2
The sum is: 3
Process finished with exit code 0

```



```

Run: "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\0сн
Enter the arguments:
The sum is: None
Process finished with exit code 0

```

Рисунок 13.8 – Результат работы программы

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

2. Сумму аргументов, расположенных между первым и последним нулевыми аргументами.

```

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

def sum_of_modules(*args):
    if args:
        nums = [int(arg) for arg in args]

```

```

        return sum(nums[nums.index(0):len(nums) - 1 - nums[::-1].index(0):])
    else:
        return None

if __name__ == '__main__':
    arguments = [i for i in input("Enter the arguments: ").split()]
    print(f"The sum is: {sum_of_modules(*arguments)}")

```

Рисунок 13.9 – Код программы

```

task_13 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы П
Enter the arguments: 1 0 2 3 0 1 2
The sum is: 5

Process finished with exit code 0

```

```

task_13 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы
Enter the arguments: 1 0 1 1 0 2 1 0 1
The sum is: 5

Process finished with exit code 0

```

```

task_13 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\
Enter the arguments:
The sum is: None

Process finished with exit code 0

```

Рисунок 13.10 – Результат работы программы

Контрольные вопросы

1. Какие аргументы называются позиционными в Python?
2. Какие аргументы называются именованными в Python?

Функция `print` принимает много разных аргументов. Например, можно передать ей две строки:

```
print("Hello" , "World") # Выведет Hello World
```

Аргументы "Hello" и "World" — позиционные. Позиционными они называются потому, что идут по порядку. Сначала первый, потом второй. Порядок их вывода в терминал зависит от позиции. Сначала выведется "Hello", а потом уже "World".

Но у функции `print` есть и другие аргументы, именованные:

```
print("Вася" , "Петя", sep=" и ") # Выведет Вася и Петя
print("Вася" , "Петя", "Маша", sep=" и ") # Выведет Вася и Петя и Маша
```

`sep=" и "` — именованный аргумент. У него есть имя — `sep`. Он говорит функции `print`, что разделять позиционные аргументы надо не пробелом, а буквой "и". При создании функций разработчики сами закладывают в них именованные аргументы. Они могут сильно менять поведение функции.

3. Для чего используется оператор `*` ?

Оператор `*` чаще всего ассоциируется у людей с операцией умножения, но в Python он имеет и другой смысл.

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

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

print(b) # [1, 2, 3, 4, 5, 6]
```

Тут берётся содержимое списка `a`, распаковывается, и помещается в список `b`.

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

Итак, мы знаем о том, что оператор «звёздочка» в Python способен «вытаскивать» из объектов составляющие их элементы. Знаем мы и о том, что существует два вида параметров функций. А именно, `*args` — это сокращение от «arguments» (аргументы), а `**kwargs` — сокращение от «keyword arguments» (именованные аргументы). Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины.