

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.10

Дисциплина: «Программирование на Python»

Тема:

«Функции с переменным числом параметров в
Python»

Выполнил: студент 2 курса
группы ИВТ-б-о-21-1
Богдашов Артём
Владимирович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.10» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

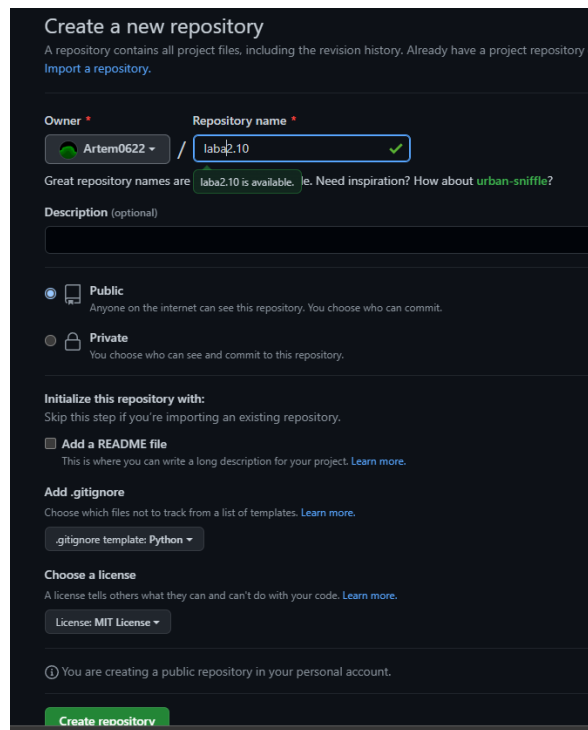


Рисунок 1.1 Создание репозитория

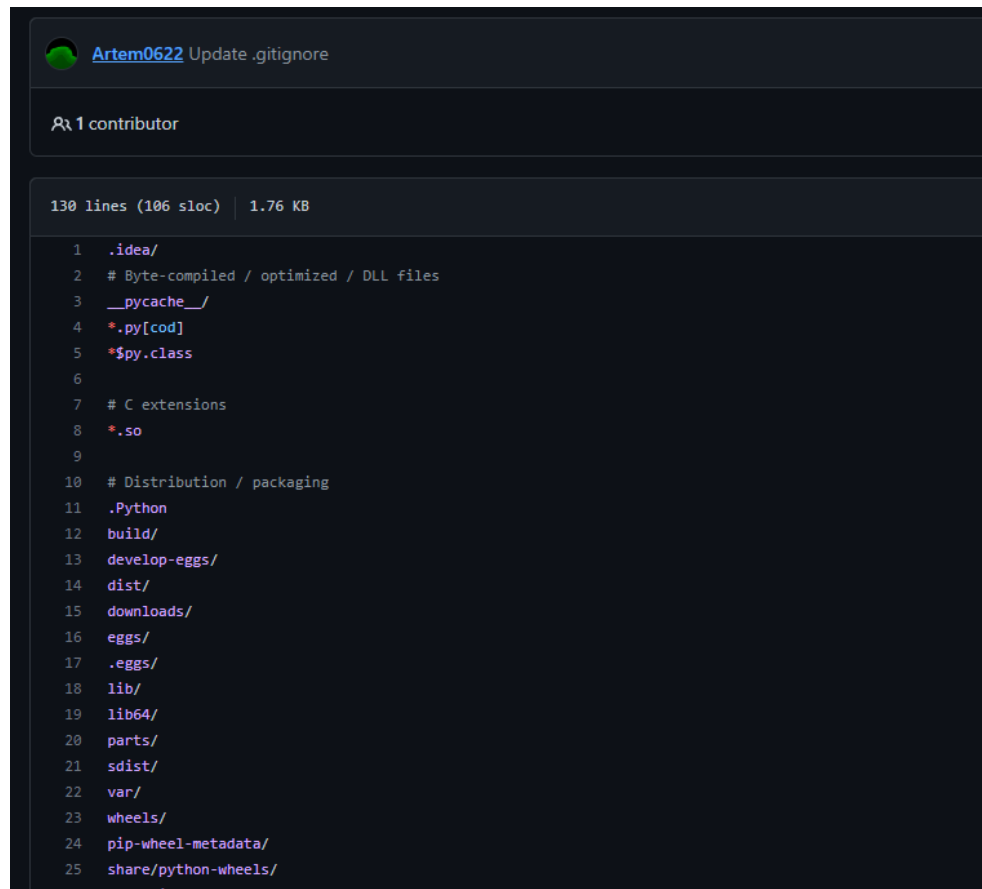
```
C:\Games\Программы\СУ\1.3\Программирование на Python\2 10>git clone https://github.com/Artem0622/laba2.10.git
Cloning into 'laba2.10'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
C:\Games\Программы\СУ\1.3\Программирование на Python\2 10>
```

Рисунок 1.2 Клонирование репозитория

```
C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

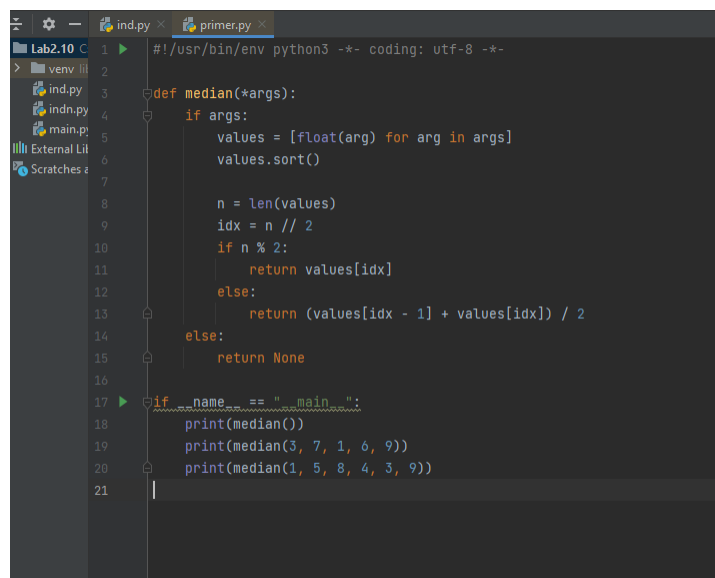
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Games/Программы/СУ/1.3/Программирование на Python/2 10/laba2.10/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления
git-flow



```
1 .idea/
2 # Byte-compiled / optimized / DLL files
3 __pycache__/
4 *.py[cod]
5 *.py.class
6
7 # C extensions
8 *.so
9
10 # Distribution / packaging
11 .Python
12 build/
13 develop-eggs/
14 dist/
15 downloads/
16 eggs/
17 .eggs/
18 lib/
19 lib64/
20 parts/
21 sdist/
22 var/
23 wheels/
24 pip-wheel-metadata/
25 share/python-wheels/
```

Рисунок 1.4 Изменение .gitignore



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

Рисунок 2.1 Результат выполнения примера 1

3.(3 вариант). Выполнил индивидуальное задание.

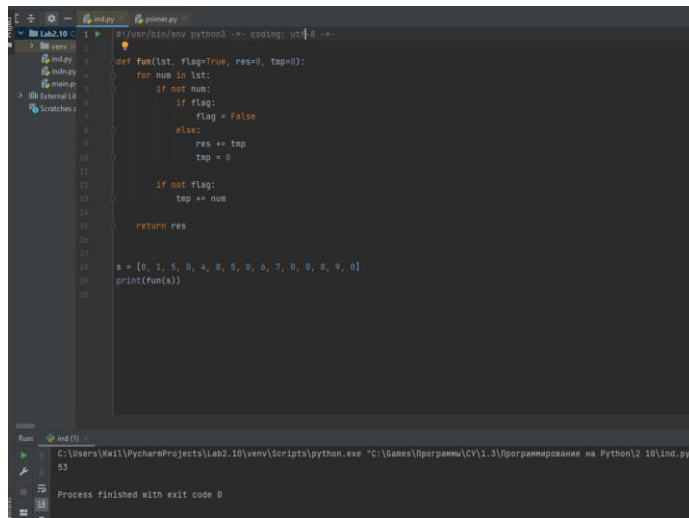


Рисунок 3.1 Вывод программы индивидуального задания

```

C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git add .
C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git commit -m "program"
[develop 3db45f0] program
2 files changed, 39 insertions(+)
create mode 100644 ind.py
create mode 100644 primer.py

```

Рисунок 4.1 Коммит изменений

```

C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

```

Рисунок 4.2 Переход на ветку main

```

C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git merge develop
Updating 00fb721..3db45f0
Fast-forward
 ind.py | 19 ++++++
 primer.py | 20 ++++++
2 files changed, 39 insertions(+)
create mode 100644 ind.py
create mode 100644 primer.py

```

Рисунок 4.3 Слияние ветки main с develop

```

C:\Games\Программы\СУ\1.3\Программирование на Python\2 10\laba2.10>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.11 KiB | 1.11 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Artem0622/laba2.10.git
58fbf28..a6952c4 main -> main

```

Рисунок 4.4 Пуш изменений

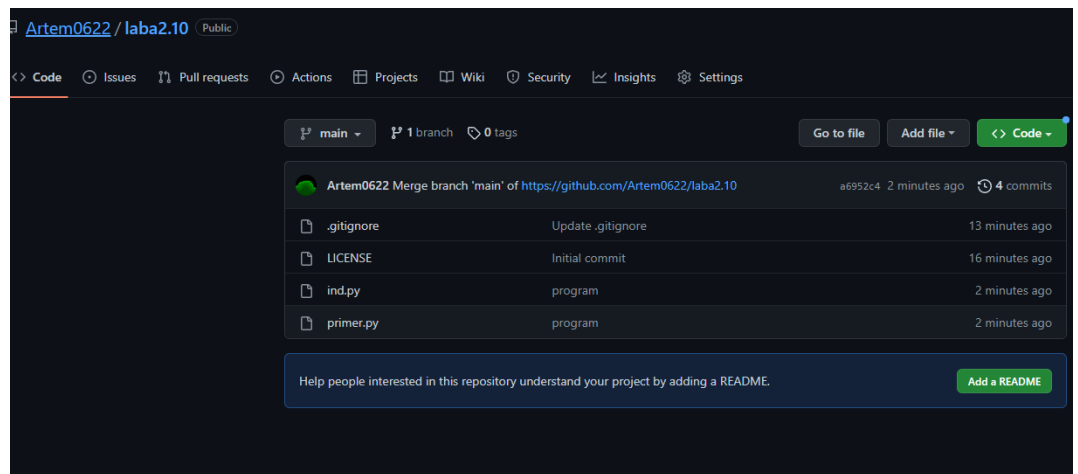


Рисунок 4.5 Изменение на уд сервере

Ответы на контрольные вопросы:

1. Какие аргументы называются позиционными в Python?

Аргументы, которые передаются без указания имен называются позиционными, потому что именно по позиции, расположению аргумента, функция понимает, какому параметру он соответствует.

2. Какие аргументы называются именованными в Python?

Аргументы, передаваемые с именами, называются именованными. При вызове функции можно использовать имена параметров из ее определения.

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.

Важно помнить, что «args» — это всего лишь набор символов, которым принято обозначать аргументы. Самое главное тут — это оператор `*`. А то, что именно идёт после него, особой роли не играет. Благодаря использованию `*` мы создали список позиционных аргументов на основе того, что было передано функции при вызове.

После того, как мы разобрались с `*args`, с пониманием `**kwargs`

проблембыть уже не должно.

Имя, опять же, значения не имеет. Главное — это два символа **. Благодаря им создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.