

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Работа с функциями в языке Python»**

**Отчет по лабораторной работе № 2.8  
по дисциплине «Программирование на Python»**

Выполнил студент группы ИВТ-б-о-21-1

Сумин Никита Сергеевич.

« » \_\_\_\_\_ 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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


**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.


---

Owner \*

Repository name \*

 BrandooDi


 / 

mywork2.8 


Great repository names are short and memorable. Need inspiration? How about [probable-octo-memory?](#)

Description (optional)

---

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

---

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python


Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License

This will set  main as the default branch. Change the default name in your [settings](#).

---

 You are creating a public repository in your personal account.

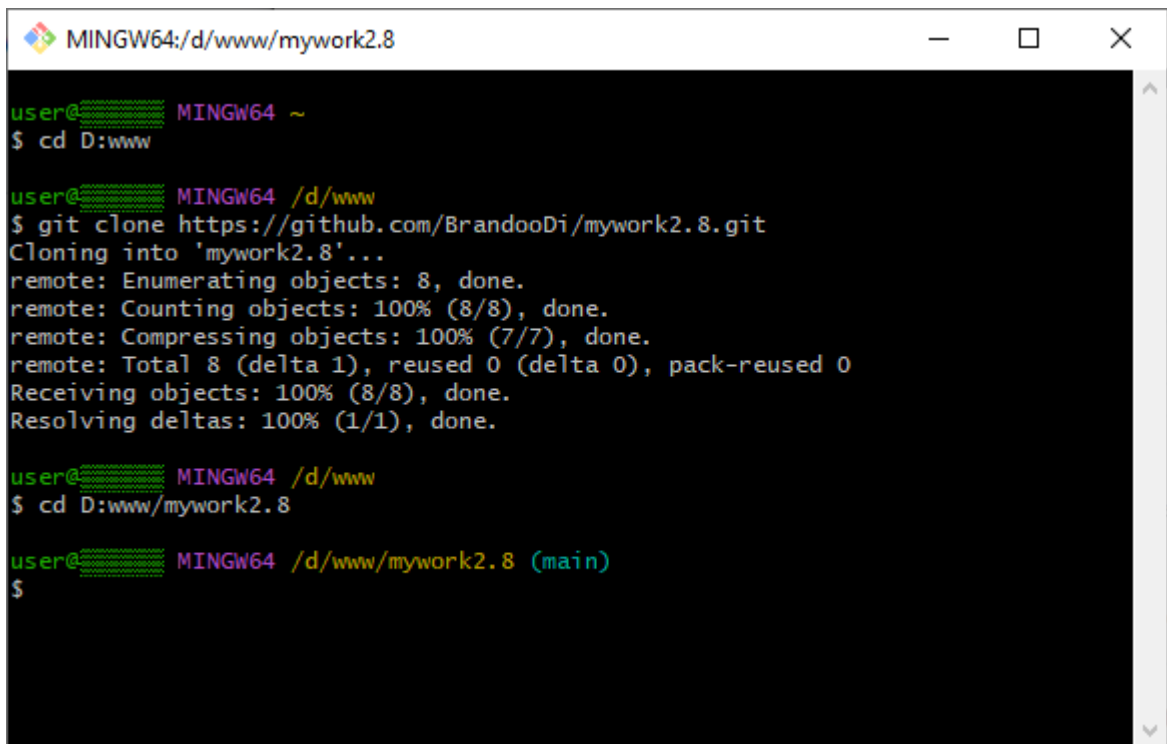
---

Create repository

---

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.



```
MINGW64:/d/www/mywork2.8

user@MINGW64 ~
$ cd D:www

user@MINGW64 /d/www
$ git clone https://github.com/BrandooDi/mywork2.8.git
Cloning into 'mywork2.8'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

user@MINGW64 /d/www
$ cd D:www/mywork2.8

user@MINGW64 /d/www/mywork2.8 (main)
$
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
user@MINGW64 /d/www/mywork2.8 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 3 - Ветвление по модели git-flow 4.

Создайте проект PyCharm в папке репозитория.

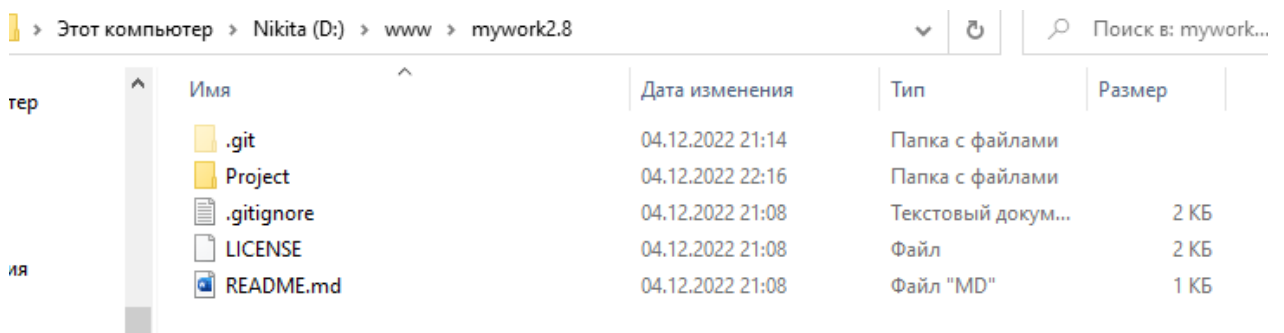


Рисунок 4 - Проект PyCharm

5. Проработайте пример лабораторной работы. Создайте для него отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

```

D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/primer.py
>>> add
Фамилия и инициалы? Иванов П.А
Должность? Куратор
Год поступления? 2000
>>> add
Фамилия и инициалы? Степаненко Д.А
Должность? Следователь
Год поступления? 2005
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Иванов П.А              |      Куратор        |      2000     |
|  2 | Степаненко Д.А         |      Следователь    |      2005     |
+-----+-----+-----+-----+
>>> exit

Process finished with exit code 0

```

Рисунок 5 - Результат выполнения примера

6. Решите задачу: задачу: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное".

Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

```
D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/zd1.py
Введите число: 5
Положительное

Process finished with exit code 0
|
```

Рисунок 6 - Результат выполнения задания 1

7. Решите следующую задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

```
D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/zd2.py
Какую площадь вы хотите вычислить?
Боковой поверхности(1)
Полную площадь(2)
> 2
Введите радиус: 3.5
Введите высоту: 2.7
Полная площадь s = 136.34512116579702

Process finished with exit code 0
|
```

Рисунок 7 - Результат выполнения задания 2

8. Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

```
D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/zd3.py
Введите числа:
3
4
5
6
7
2
5
0
Произведение: 25200.0

Process finished with exit code 0
|
```

Рисунок 8 - Результат выполнения задания 3

9. Решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.

3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

```
D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/zd4.py
Введите строку: 5
5

Process finished with exit code 0
|
```

Рисунок 9 - Результат выполнения задания 2

10. Решите индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```
D:\www\mywork2.8\Project\venv\Scripts\python.exe D:/www/mywork2.8/Project/ind1.py
>>> add
Расчётный счёт платильщика: A
Расчётный счет получателя: Й
Перечисляемая сумма в руб: 344
>>> add
Расчётный счёт платильщика: Б
Расчётный счет получателя: Е
Перечисляемая сумма в руб: 365
>>> list
+-----+-----+-----+
| No | Расчётный счёт платильщика | Расчётный счет получателя | Перечисляемая сумма в руб |
+-----+-----+-----+
| 1 | A | Й | 344 |
| 2 | Б | Е | 365 |
+-----+-----+-----+
>>> exit

Process finished with exit code 0
|
```

Рисунок 10 - Результат выполнения индивидуального задания

### Контрольные вопросы:

#### 1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых

отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные.

## **2. Каково назначение операторов `def` и `return` ?**

В языке программирования Python функции определяются с помощью оператора `def`.

В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`.

## **3. Каково назначение локальных и глобальных переменных при написании функций в Python?**

Локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции.

## **4. Как вернуть несколько значений из функции Python?**

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.

Когда из функции возвращается несколько значений, на самом деле из нее возвращается один объект класса `tuple`.

## **5. Какие существуют способы передачи значений в функцию?**

По ссылке и по значению.

## **6. Как задать значение аргументов функции по умолчанию?**



При передаче параметров присвоить им значения.

## **7. Каково назначение lambda-выражений в языке Python?**

lambda – это выражение, а не инструкция. По этой причине ключевое слово lambda может появляться там, где синтаксис языка Python не позволяет использовать инструкцию def , – внутри литералов или в вызовах функций.

## **8. Как осуществляется документирование кода согласно PEP257?**

Для согласованности, всегда используйте `"""triple double quotes"""` для строк документации. Используйте `r"""raw triple double quotes"""` , если вы будете использовать обратную косую черту в строке документации.

## **9. В чем особенность однострочных и многострочных форм строк документации?**

Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции).

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием.

**Вывод:** были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.