

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

Отчет по лабораторной работе №8

По дисциплине основы кроссплатформенного программирования
«Работа с функциями в языке Python»

Выполнила:

студентк группы ИТС-б-о-21-1

Аллаёров Жамшид Хасан угли

(подпись)

Проверил: Доцент, к.т.н, доцент
кафедры

инфокоммуникаций

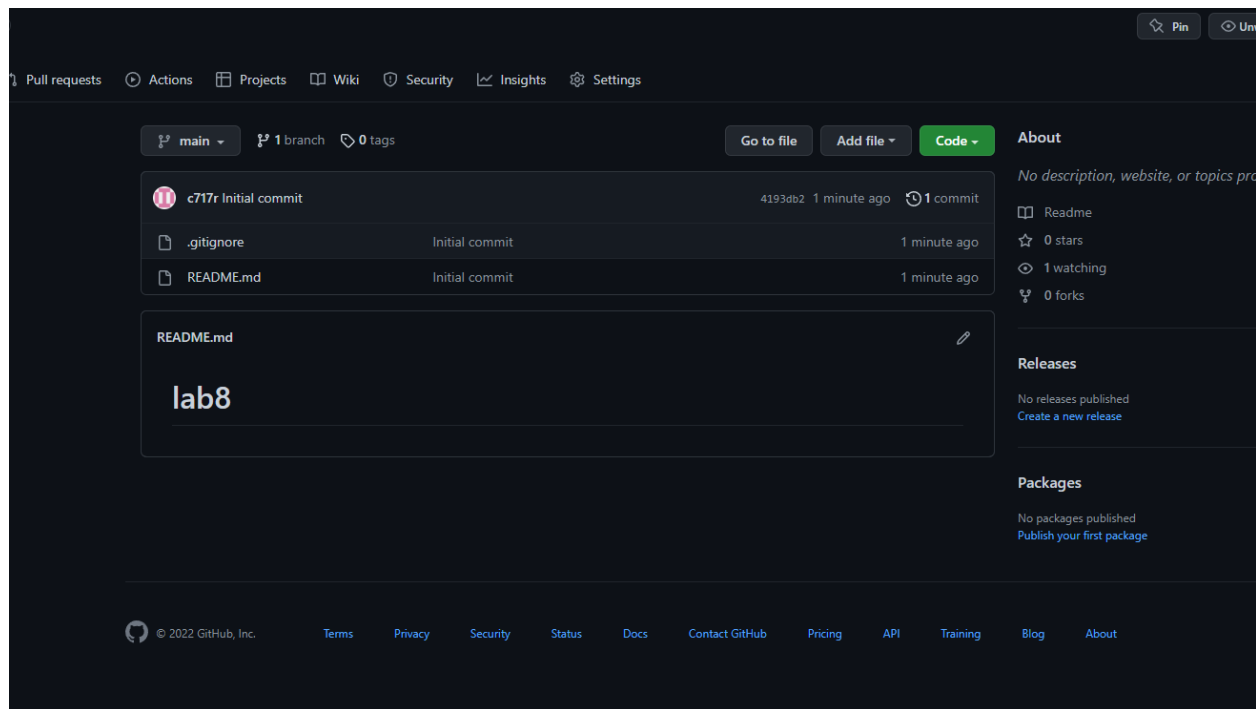
Воронкин Р. А.

Работа защищена с оценкой:

(подпись)

Ставрополь, 2022

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка **Add .gitignore**)



Решить следующую 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".
else:

```
print(f'Неизвестная команда {command}', file=sys.stderr)
```

```
if __name__ == '__main__':
```

```
main()
```

Понятно, что вызов *test()* должен следовать после определения функций.

Однако имеет ли

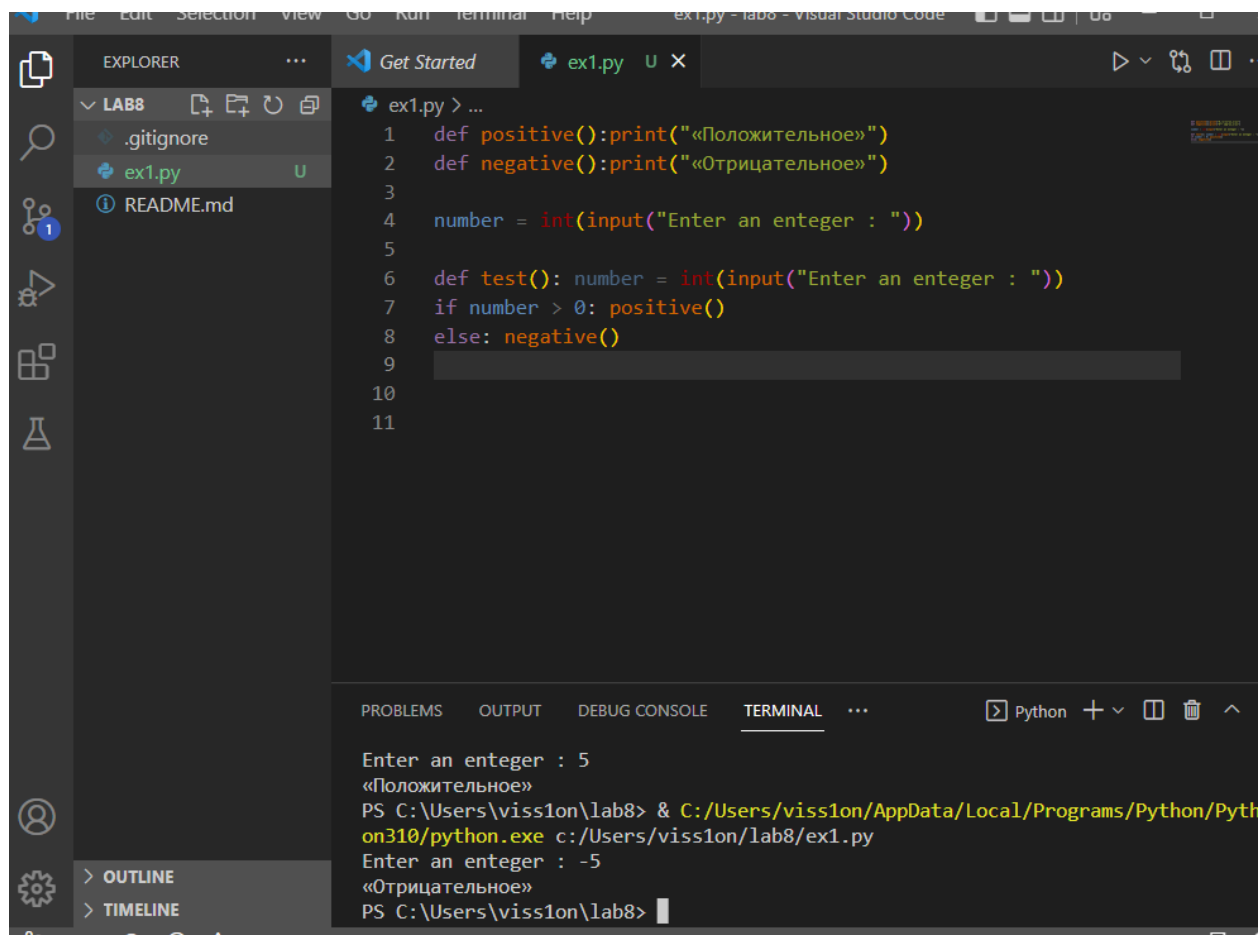
значение порядок определения самих функций? То есть должны ли

определения *positive()* и

negative() предшествовать *test()* или могут следовать после него? Проверьте

вашу гипотезу,

поменяв объявления функций местами. Попробуйте объяснить результат.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a file named `ex1.py`. The main editor displays the following Python code:

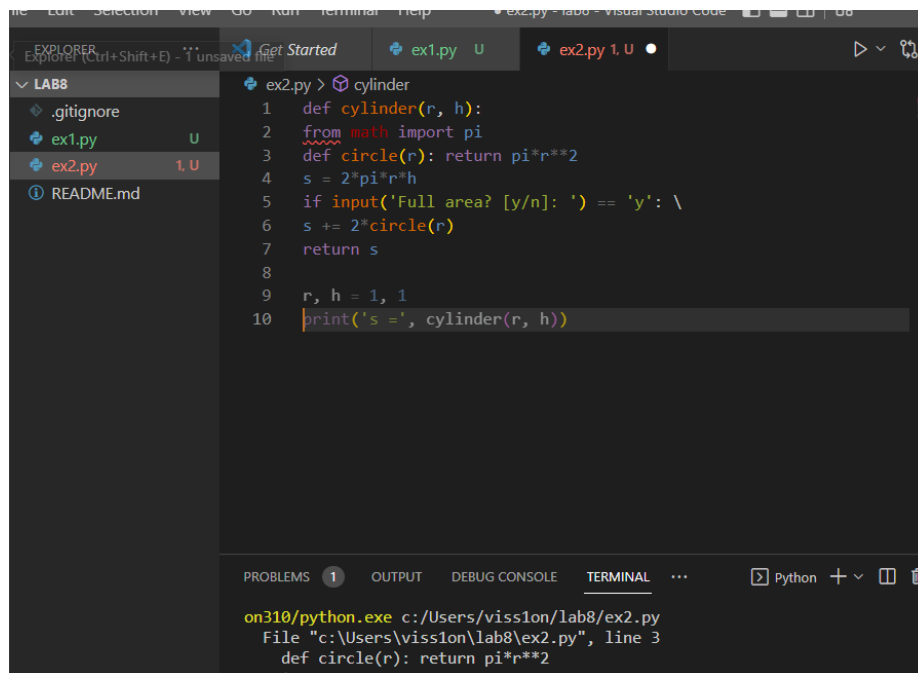
```
1 def positive():print("«Положительное»")
2 def negative():print("«Отрицательное»")
3
4 number = int(input("Enter an integer : "))
5
6 def test(): number = int(input("Enter an integer : "))
7 if number > 0: positive()
8 else: negative()
9
10
11
```

The TERMINAL pane at the bottom shows the execution of the script:

```
Enter an integer : 5
«Положительное»
PS C:\Users\visson\lab8> & C:/Users/visson/AppData/Local/Programs/Python/Python310/python.exe c:/Users/visson/lab8/ex1.py
Enter an integer : -5
«Отрицательное»
PS C:\Users\visson\lab8>
```

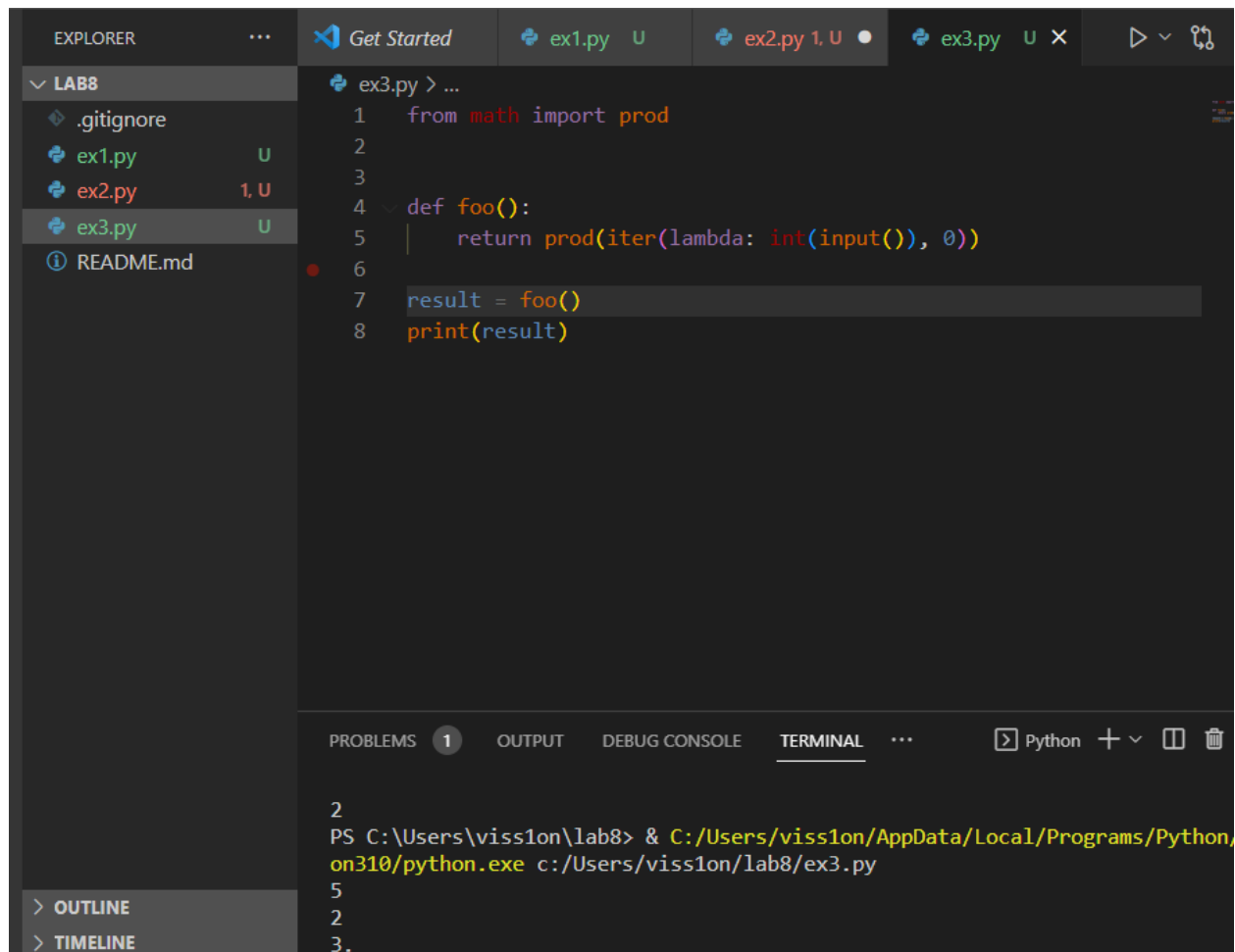
в основной ветке программы вызывается функция *cylinder()*, которая вычисляет площадь цилиндра. В теле *cylinder()* определена функция *circle()*, вычисляющая площадь круга по формуле . В теле *cylinder()* у пользователя

спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле $S_{\text{бок}} = 2\pi rh$, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции *circle()*.



```
1 def cylinder(r, h):
2     from math import pi
3     def circle(r): return pi*r**2
4     s = 2*pi*r*h
5     if input('Full area? [y/n]: ') == 'y': \
6         s += 2*circle(r)
7     return s
8
9 r, h = 1, 1
10 print('s =', cylinder(r, h))
```

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



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project named 'LAB8' containing files: '.gitignore', 'ex1.py', 'ex2.py', 'ex3.py', and 'README.md'. The main editor window shows the code in 'ex3.py':

```
1 from math import prod
2
3
4 def foo():
5     return prod(iter(lambda: int(input()), 0))
6
7 result = foo()
8 print(result)
```

At the bottom, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\viss1on\lab8> & C:/Users/viss1on/AppData/Local/Programs/Python/Python310/python.exe c:/Users/viss1on/lab8/ex3.py
5
2
3.
```

напишите программу, в которой определены следующие четыре функции:

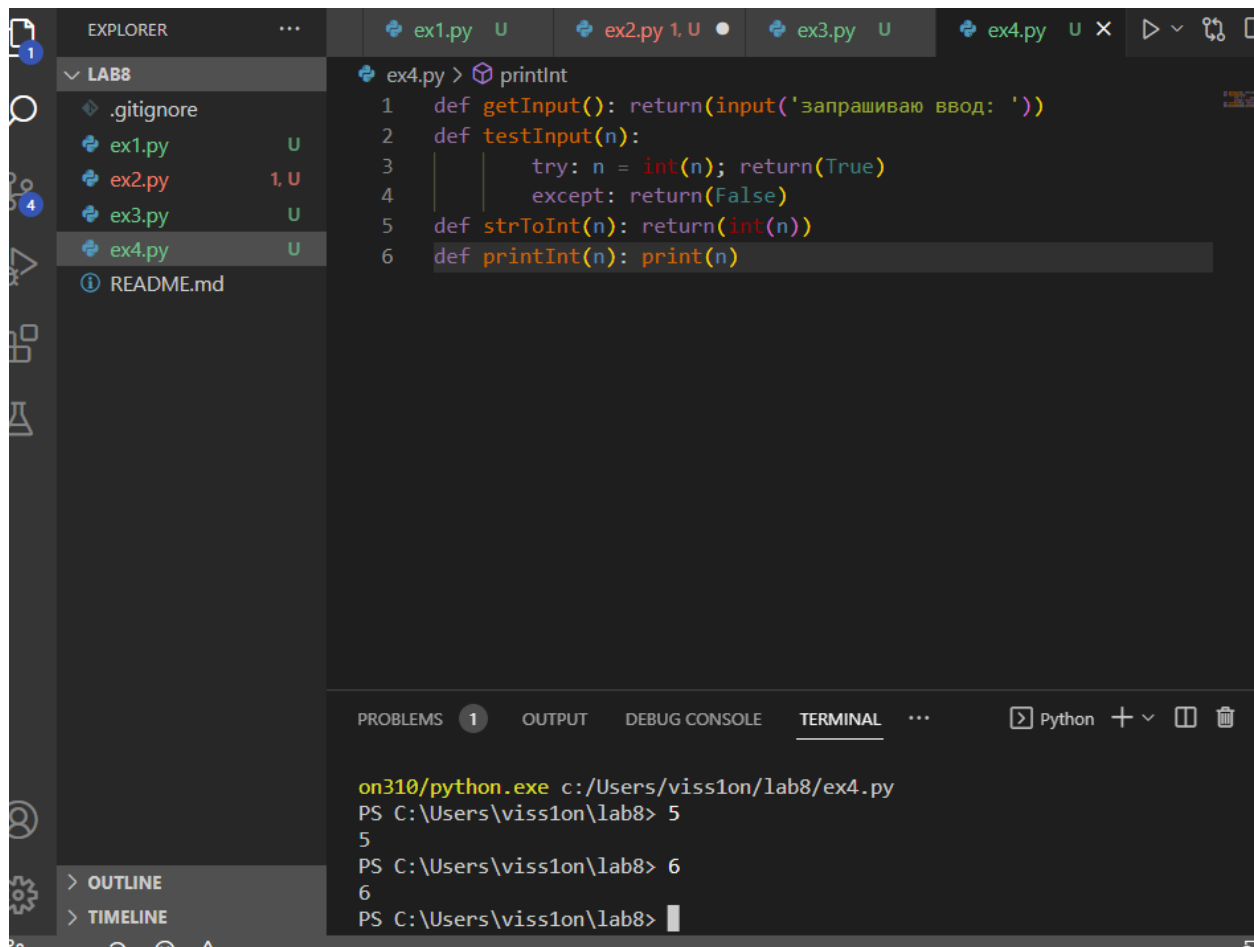
1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.
2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое *True*. 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



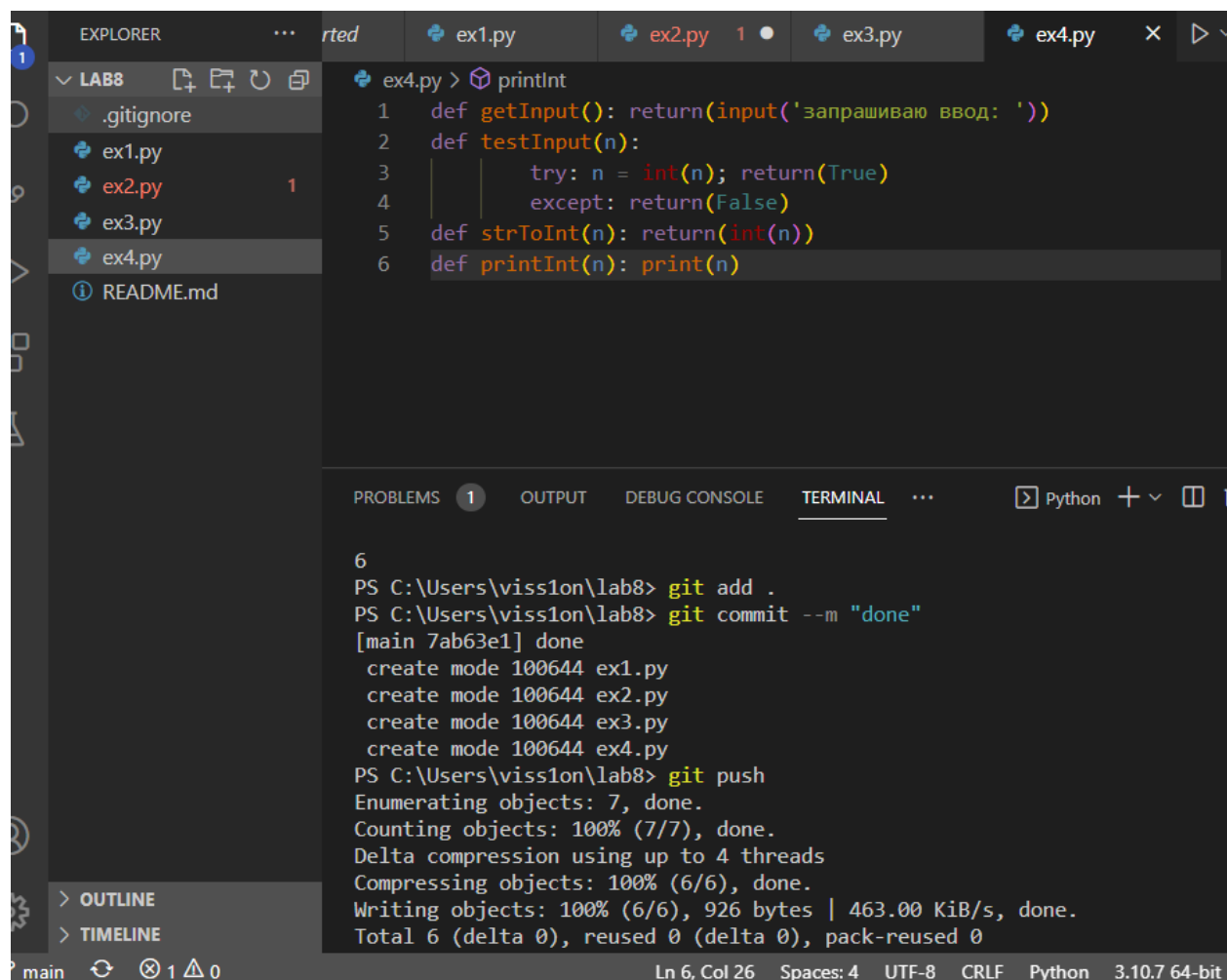
The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a folder named 'LAB8' containing files: '.gitignore', 'ex1.py', 'ex2.py', 'ex3.py', 'ex4.py', and 'README.md'. The file 'ex4.py' is selected. The main editor area displays the code for 'ex4.py':

```
1 def getInput(): return(input('запрашиваю ввод: '))
2 def testInput(n):
3     try: n = int(n); return(True)
4     except: return(False)
5 def strToInt(n): return(int(n))
6 def printInt(n): print(n)
```

Below the editor, the TERMINAL pane is active, showing the command prompt output for running 'ex4.py' using 'python.exe':

```
on310/python.exe c:/Users/viss1on/lab8/ex4.py
PS C:\Users\viss1on\lab8> 5
5
PS C:\Users\viss1on\lab8> 6
6
PS C:\Users\viss1on\lab8>
```

Отправьте сделанные изменения на сервер GitHub.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays a project named 'LAB8' with files: '.gitignore', 'ex1.py', 'ex2.py' (with a red error indicator), 'ex3.py', 'ex4.py' (selected), and 'README.md'. The Editor panel shows the content of 'ex4.py', which contains the following Python code:

```
ex4.py > printInt
1 def getInput(): return(input('запрашиваю ввод: '))
2 def testInput(n):
3     |     try: n = int(n); return(True)
4     |     except: return(False)
5 def strToInt(n): return(int(n))
6 def printInt(n): print(n)
```

The TERMINAL panel at the bottom shows the execution of the script in a PowerShell prompt:

```
6
PS C:\Users\viiss1on\lab8> git add .
PS C:\Users\viiss1on\lab8> git commit -m "done"
[main 7ab63e1] done
create mode 100644 ex1.py
create mode 100644 ex2.py
create mode 100644 ex3.py
create mode 100644 ex4.py
PS C:\Users\viiss1on\lab8> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 926 bytes | 463.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
```

The status bar at the bottom indicates the current file is 'main', with 1 error and 0 warnings. The cursor is at line 6, column 26, using 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.10.7 64-bit.

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, то есть 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. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

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

Функция в `python` - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции **`def`**.

Определим простейшую функцию:

```
def add(x, y):  
    return x + y
```

Инструкция **`return`** говорит, что нужно вернуть значение. В нашем случае функция возвращает сумму `x` и `y`.

Теперь мы ее можем вызвать:

```
>>>
```

```
>>> add(1, 10)  
11  
>>> add('abc', 'def')  
'abcdef'
```


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

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, 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. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

4. Как вернуть несколько значений из функции Python?

Словари содержат комбинации элементов, которые представляют собой пары «ключ — значение» (key:value), заключенные в фигурные скобки ({}).

Словари, на мой взгляд, это оптимальный вариант для работы, если вы знаете ключ для доступа к значениям. Далее представлен словарь, где ключом является имя человека, а соответствующим значением — возраст.

```
people={  
    'Robin': 24,  
    'Odin': 26,  
    'David': 25  
}
```

А теперь перейдем к функции, которая возвращает словарь с парами «ключ — значение».

```
# A Python program to return multiple values using dictionary  
# This function returns a dictionary  
def people_age():  
    d = dict();  
    d['Jack'] = 30  
    d['Kim'] = 28  
    d['Bob'] = 27  
    return d  
d = people_age()  
print(d)  
# {'Bob': 27, 'Jack': 30, 'Kim': 28}
```

5. Какие существуют способы передачи значений в функцию?

В общем, условно(для упрощения понимания) есть 2 способа передачи аргументов в функцию в Си : по значению, по указателю.

6. Как задать значение аргументов функции по умолчанию?

Параметры по умолчанию позволяют задавать формальным параметрам функции значения по умолчанию в случае, если функция вызвана без

аргументов, или если параметру явным образом передано значение `undefined`.

7. Каково назначение `lambda`-выражений в языке Python?

Лямбда-функции в Python являются анонимными. Это означает, что функция безымянна. Как известно, ключевое слово `def` используется в Python для определения обычной функции. В свою очередь, ключевое слово `lambda` используется для определения анонимной функции.

8. Как осуществляется документирование кода согласно PEP257?

Целью данного PEP является стандартизация высокоуровневой структуры документационных строк: описать, что именно они должны содержать и объяснять (При этом мы не будем обговаривать фактический синтаксис разметки). PEP содержит не жесткие предписания, а рекомендации:

«Общепринятые соглашения обеспечивают ясность, логичность, удобство сопровождения и воспитывают хорошие программистские привычки. Но они не заставляют вас действовать против своей воли. Это Python!»

Тим Питерс на `comp.lang.python`, 2001-06-16

Если вы избегаете общепринятых соглашений, в худшем случае на вас посмотрят искоса. Но существуют некоторые приложения (Например, системы документирования, подобные Docutils), которые позволят вам добиться более качественного результата, если вы знаете о договорённостях и будете им следовать.

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

Однострочные строки документации должны помещаться на одной строке.

Стандартные соглашения для написания однострочных строк документации:

- Несмотря на то, что они однострочные, мы по-прежнему используем тройные кавычки вокруг этих строк документации, тогда их можно будет легко расширить позже.
- Закрывающие кавычки находятся на той же строке, что и открывающие кавычки.
- Не нужно добавлять пустую строку ни перед, ни после строки документации.
- Они не должны быть описательными, скорее они должны следовать структуре «Делает это, возвращает это», заканчивающейся точкой.
- Многострочные строки документации состоят из резюмирующей однострочной строки документации, за которой следует пустая строка, а затем более подробное описание.
- Документ [PEP 257](#) предоставляет стандартные соглашения для написания многострочных строк документации для различных объектов.