

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

Институт цифрового развития  
Кафедра инфокоммуникаций

Отчет по лабораторной работе №2

По дисциплине основы кроссплатформенного программирования  
«Основы языка 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.

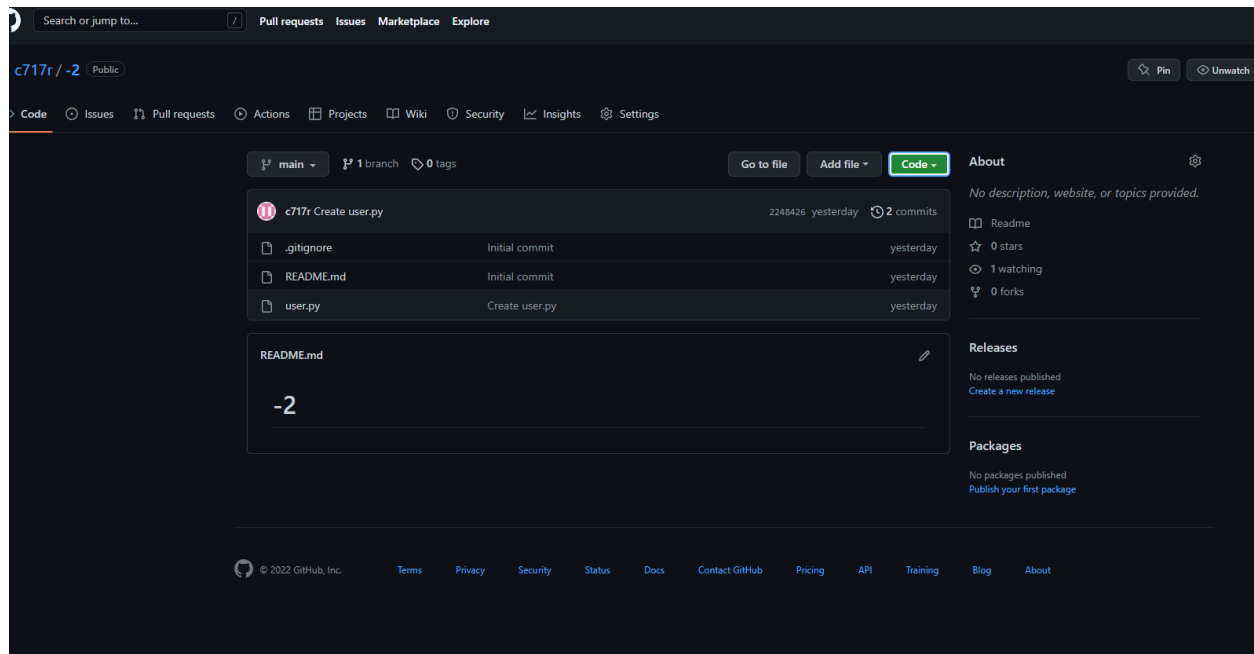
Работа защищена с оценкой:

---

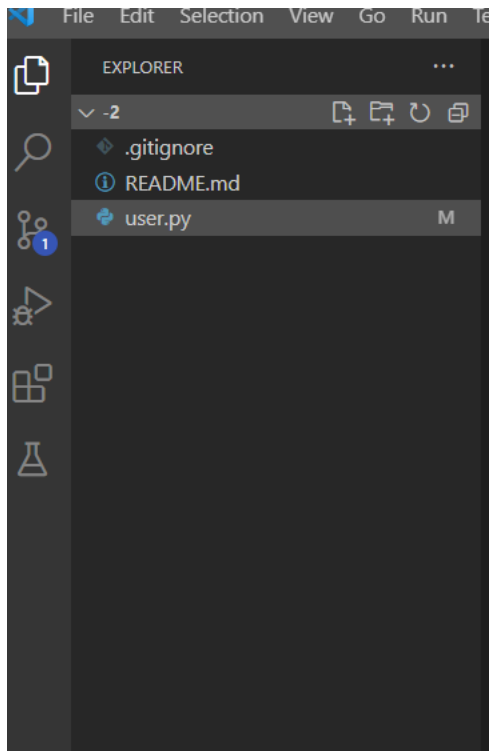
(подпись)

Ставрополь, 2022

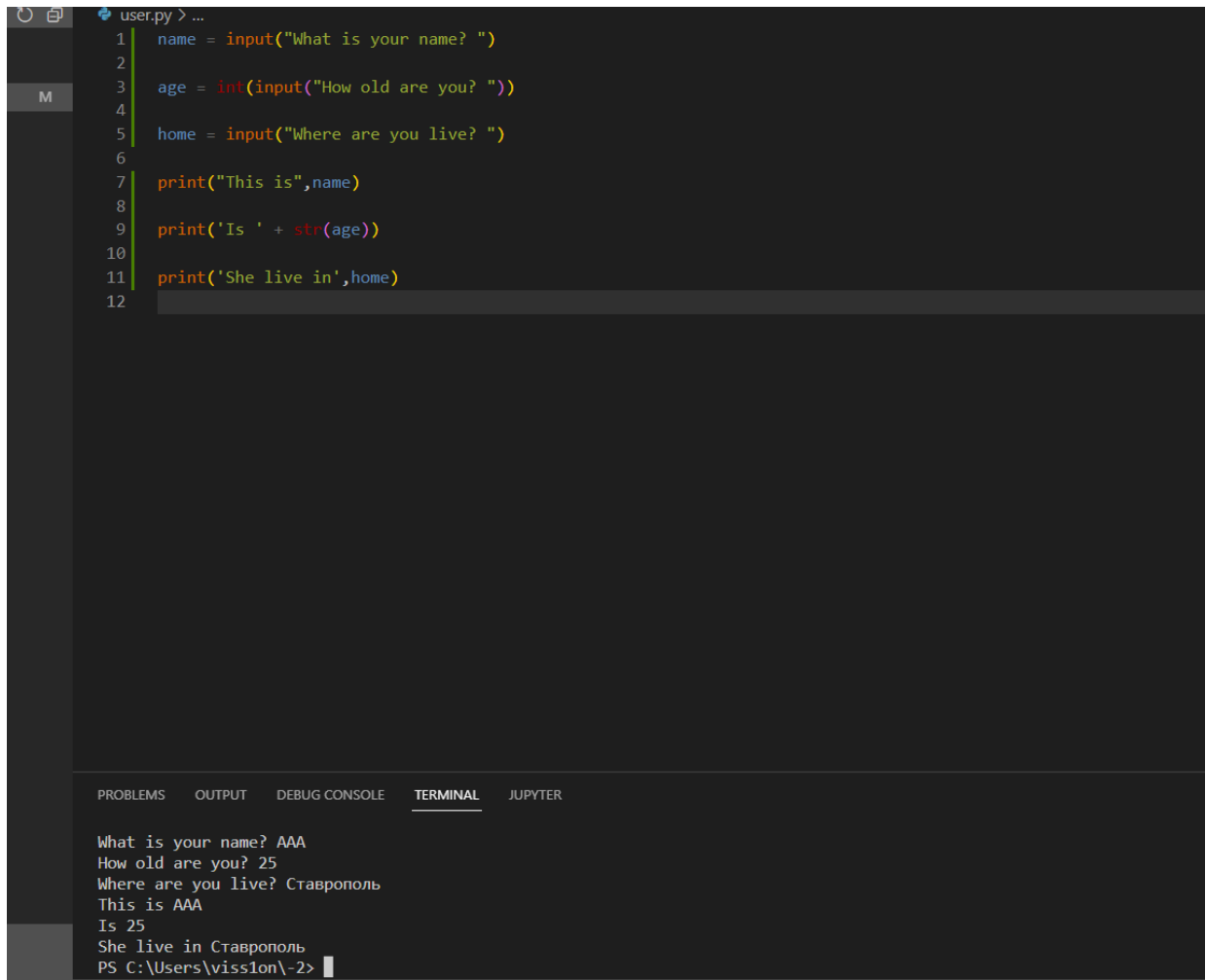
Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



Выполните клонирование созданного репозитория.



. Напишите программу (файл *user.py*), которая запрашивала бы у пользователя:  
его имя (например, "What is your name?")  
возраст ("How old are you?")  
место жительства ("Where are you live?")  
После этого выводила бы три строки:



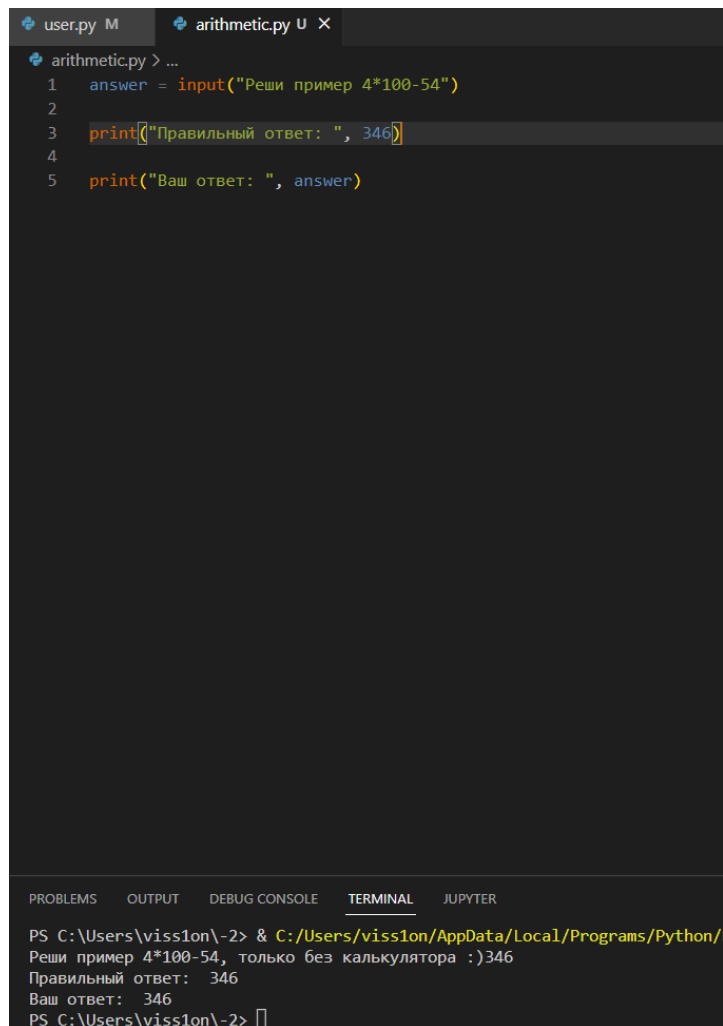
The image shows a Jupyter Notebook interface with a dark theme. The top part displays a Python script in a code editor. The script prompts the user for their name, age, and location, then prints the results. The bottom part shows the terminal output of the script, where the user has entered 'AAA' for the name, '25' for the age, and 'Ставрополь' for the location. The output shows the program's response for each input.

```
user.py > ...
1  name = input("What is your name? ")
2
3  age = int(input("How old are you? "))
4
5  home = input("Where are you live? ")
6
7  print("This is",name)
8
9  print('Is ' + str(age))
10
11 print('She live in',home)
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
What is your name? AAA
How old are you? 25
Where are you live? Ставрополь
This is AAA
Is 25
She live in Ставрополь
PS C:\Users\viisslon\>
```

Напишите программу (файл *arithmetic.py*), которая предлагала бы пользователю решить пример  $4 * 100 - 54$ . Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число.

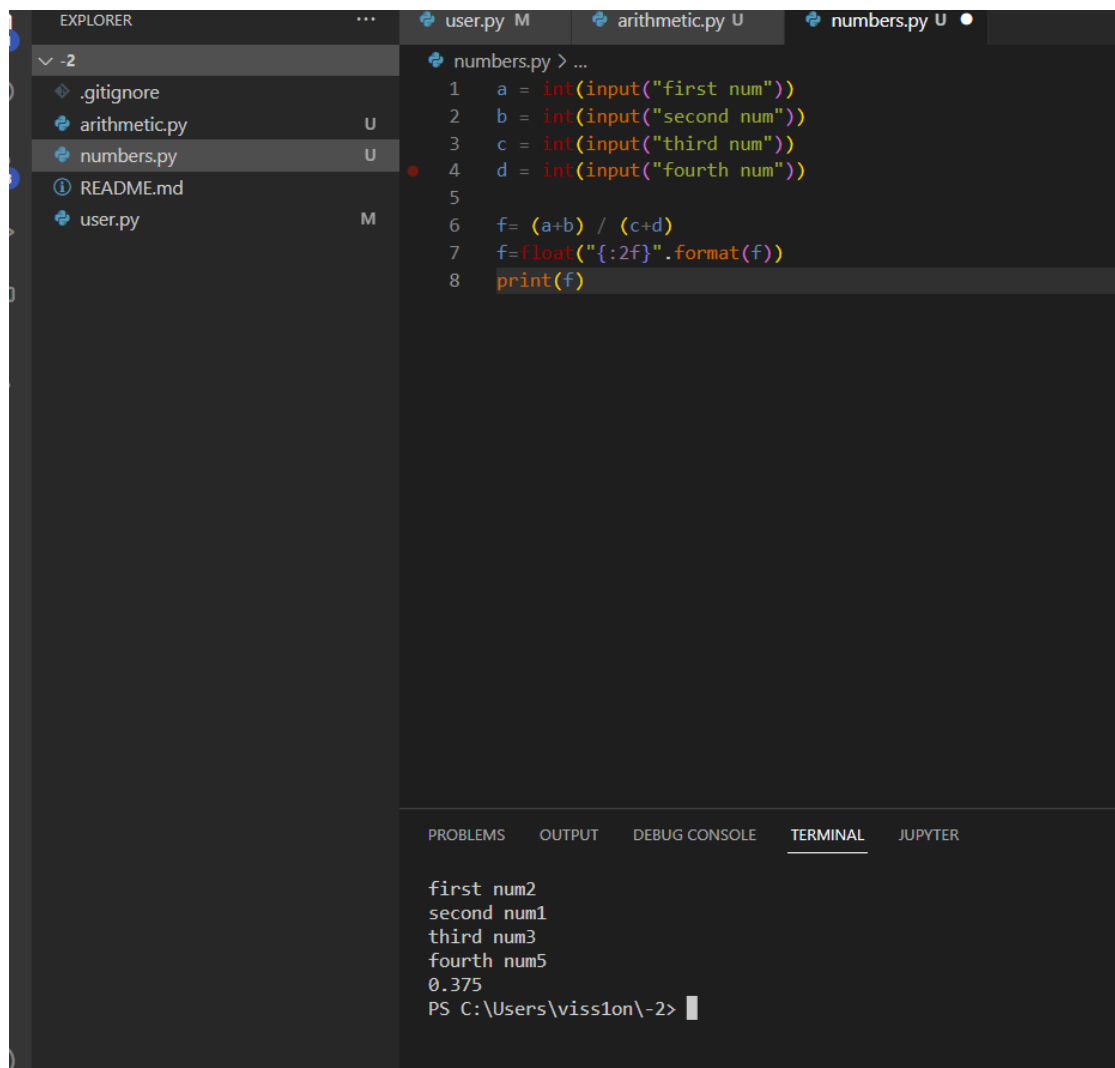


```
user.py M arithmetic.py U X
arithmetic.py > ...
1  answer = input("Реши пример 4*100-54")
2
3  print("Правильный ответ: ", 346)
4
5  print("Ваш ответ: ", answer)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS C:\Users\viiss1on\> & C:/Users/viiss1on/AppData/Local/Programs/Python/Python38-32/Python.exe C:\Users\viiss1on\arithmetic.py
Реши пример 4*100-54, только без калькулятора :)346
Правильный ответ:  346
Ваш ответ:  346
PS C:\Users\viiss1on\>
```

Запросите у пользователя четыре числа (файл *numbers.py*). Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named '-2' containing files: .gitignore, arithmetic.py, numbers.py, README.md, and user.py. The code editor shows the content of numbers.py, which is a Python script that prompts the user for four numbers, calculates the sum of the first two, the sum of the last two, and then divides the first sum by the second sum, formatting the result to two decimal places. The terminal at the bottom shows the output of the script, which is 0.375.

```
EXPLORER
-2
  .gitignore
  arithmetic.py
  numbers.py
  README.md
  user.py

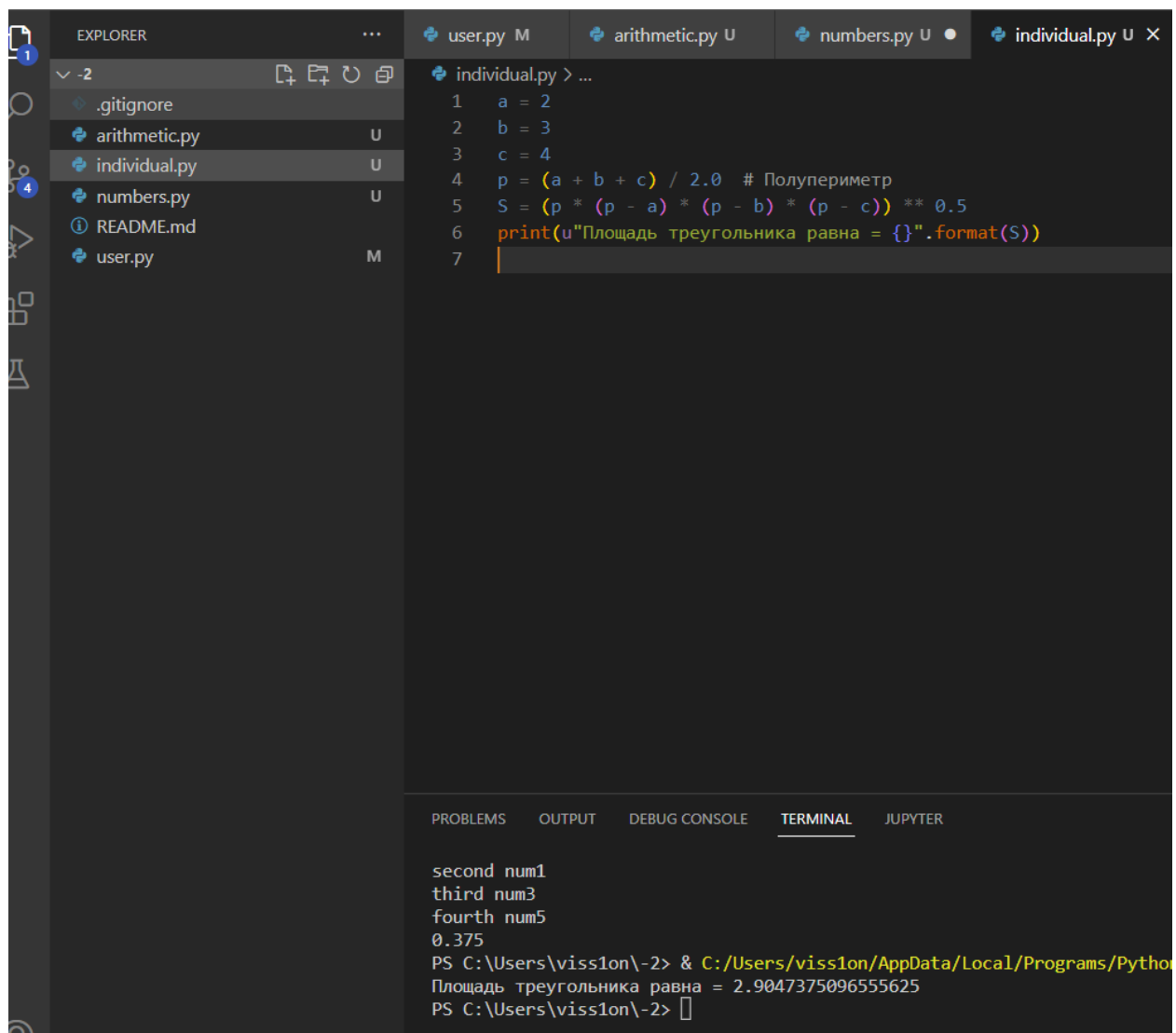
numbers.py
1 a = int(input("first num"))
2 b = int(input("second num"))
3 c = int(input("third num"))
4 d = int(input("fourth num"))
5
6 f= (a+b) / (c+d)
7 f=float("{:2f}".format(f))
8 print(f)

TERMINAL
first num2
second num1
third num3
fourth num5
0.375
PS C:\Users\viss1on\>
```

Напишите программу (файл *individual.py*) для решения индивидуального задания.

Вариант

индивидуального задания уточните у преподавателя.

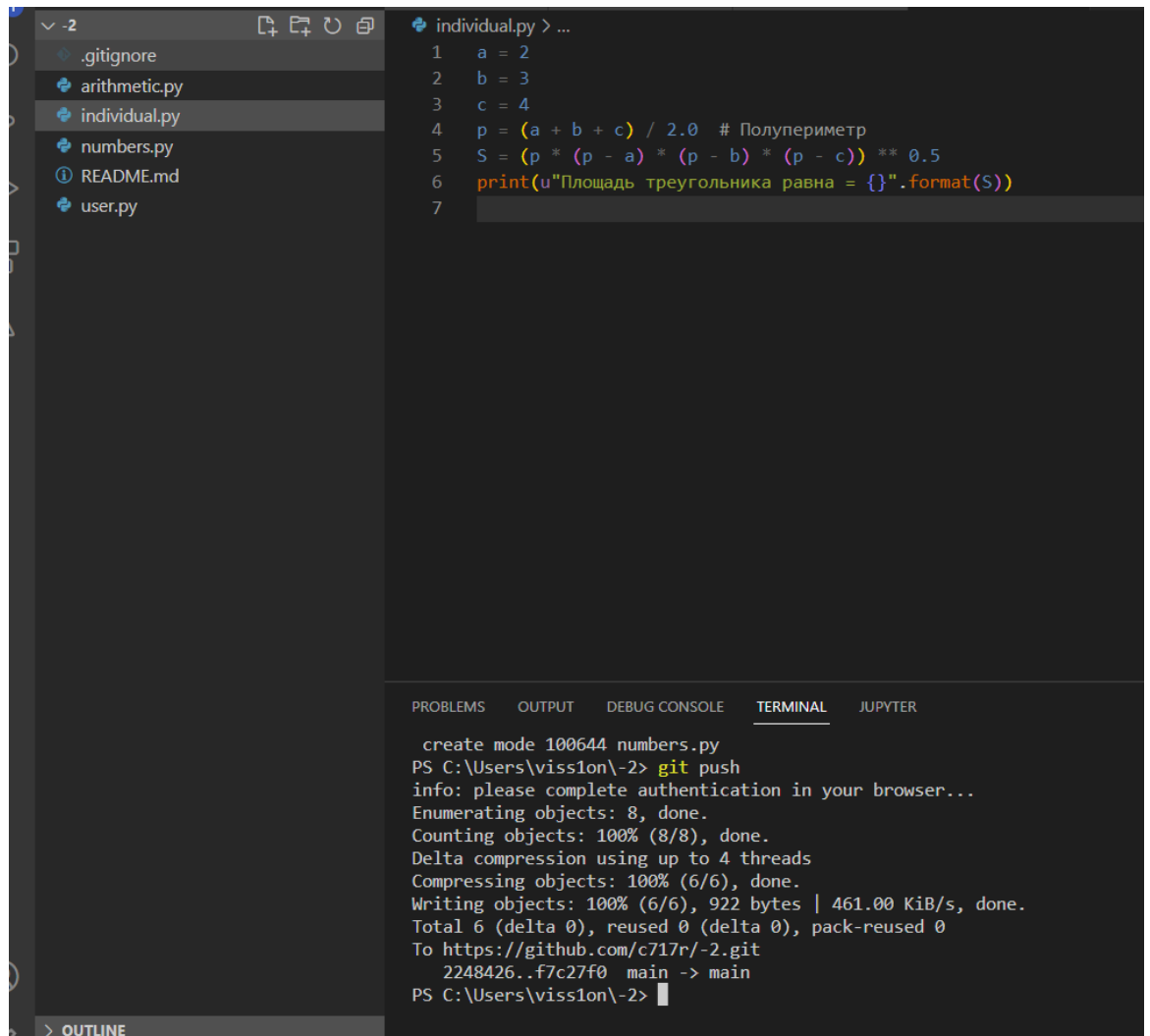


The screenshot shows a Visual Studio Code editor with a dark theme. The Explorer sidebar on the left displays a file tree for a project named '-2', containing files: .gitignore, arithmetic.py, individual.py, numbers.py, README.md, and user.py. The 'individual.py' file is selected and its content is visible in the main editor. The code defines variables a=2, b=3, c=4, calculates the semi-perimeter p, and uses Heron's formula to calculate the area S. The output is printed as 'Площадь треугольника равна = {}'. The bottom panel shows the TERMINAL with the command 'python individual.py' executed, resulting in the output 'Площадь треугольника равна = 2.9047375096555625'.

```
1 a = 2
2 b = 3
3 c = 4
4 p = (a + b + c) / 2.0 # Полупериметр
5 S = (p * (p - a) * (p - b) * (p - c)) ** 0.5
6 print("Площадь треугольника равна = {}".format(S))
7
```

```
second num1
third num3
fourth num5
0.375
PS C:\Users\vision\> & C:/Users/vis1on/AppData/Local/Programs/Python
Площадь треугольника равна = 2.9047375096555625
PS C:\Users\vision\>
```

Выполните коммит файлов *user.py*, *arithmetic.py*, *numbers.py* и *individual.py* в репозиторий *git* в ветку для разработки.



```
individual.py > ...
1  a = 2
2  b = 3
3  c = 4
4  p = (a + b + c) / 2.0 # Полупериметр
5  S = (p * (p - a) * (p - b) * (p - c)) ** 0.5
6  print("Площадь треугольника равна = {}".format(S))
7
```

```
create mode 100644 numbers.py
PS C:\Users\vision\> git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 922 bytes | 461.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/c717r/-2.git
 2248426..f7c27f0  main -> main
PS C:\Users\vision\>
```

## Вопросы для защиты работы

### 1. Опишите основные этапы установки Python в Windows и Linux

Для того чтобы приступить к работе с Python 3, следует сначала обеспечить доступ к интерпретатору «Пайтон». Есть несколько способов это выполнить: • для Windows рекомендуется скачать новую версию Python на официальном сайте **python.org**. Речь идёт о загрузке нужного установочного файла (file) с учётом операционной системы компьютера и последующего запуска этого файла; • для Linux можно воспользоваться специальным **менеджером пакетов**, который предоставляется операционной системой, включая Linux (пакетный менеджер обеспечит запуск и установку Python); • для ОС macOS используют менеджер пакетов Homebrew; • для мобильных ОС (Android, iOS) устанавливают приложения, предоставляющие среду разработки Python (неплохой способ попрактиковать навыки, что называется, на ходу).

### 2. В чем отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта?

Вкратце, это менеджер пакетов, который входит в стандартную комплектацию Anaconda / Miniconda и представляет собой систему управления окружающей средой.

Anaconda - это платформа, которая предоставляет пакеты для науки о данных (если вы собираетесь узнать разницу между conda и pip, вы уже знаете). Вы можете одновременно установить языки программирования для науки о данных, такие как Python и R, и пакеты, необходимые для статистического анализа и машинного обучения, и сразу же создать среду, использующую Python.

Еще есть Miniconda в минимальной комплектации.

conda работает только при установке установщиком Anaconda или установщиком miniconda. Даже если вы установите conda в среде python + pip, вы не сможете использовать его, как в дистрибутиве Anaconda.

### 3. Как осуществить проверку работоспособности пакета Anaconda?

```
conda --version
```

В результате вы увидите версию Conda, установленную на вашем ПК. А это значит, что у нас все готово для работы. Далее мы рассмотрим важнейшие команды для управления средами в большинстве случаев.

#### 1. Проверка доступных сред



Для начала выясним, какие среды находятся в нашем распоряжении. Выполняем следующую команду:

```
conda env list
```

Предположим, вы только что установили Conda, тогда непременно увидите вот такой результат:

```
# среды conda
#
base * /opt/anaconda3
```

#### 4. Как задать используемый интерпретатор языка Python в IDE PyCharm?

В интегрированной среде от JetBrains поддерживаются основные современные Python-фреймворки для веб-разработки, в ней вы сможете работать с Jupyter-notebook, подключать Anaconda, а также прочие библиотеки для научных вычислений и Data Science. Но, как говорится — "И это ещё не всё!". Помимо, собственно, питона, PyCharm отлично ладит и с другими языками программирования — JS, TypeScript-a, SQL или шаблонизаторами.

#### 5. Как осуществить запуск программы с помощью IDE PyCharm?

Для этого перейдем на [страницу загрузки](#) и загрузим установочный файл PyCharm Community.

После загрузки выполним его установку.

После завершения установки запустим программу. При первом запуске открывается начальное окно:

Создадим проект и для этого выберем пункт **New Project**.

Далее нам откроется окно для настройки проекта. В поле **Location** необходимо указать путь к проекту. В моем случае проект будет помещаться в папку HelloApp. Собственно название папки и будет названием проекта.

Кроме пути к проекту все остальные настройки оставим **по умолчанию** и нажмем на кнопку Create для создания проекта.

После этого будет создан пустой проект:

#### 6. В чем суть интерактивного и пакетного режимов работы Python?

Пожалуй, самый простой способ запускать программы на языке Python - это вводить инструкции непосредственно в командной строке интерпретатора,

которая иногда называется интерактивной оболочкой. Запустить эту командную строку можно разными способами - в интегрированной среде разработки,

в системной консоли и так далее. Предположим, что интерпретатор установлен в вашей системе как выполняемая программа, тогда самый универсальный способ запустить интерактивный сеанс работы с интерпретатором заключается в том, чтобы ввести команду python без аргументов в командной строке вашей операционной системы. Например:

**% python**

**Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (Intel)] ...**

**Type "help", "copyright", "credits" or "license" for more information.**

**>>>**

После ввода слова «python» командная оболочка вашей операционной системы запустит интерактивный сеанс работы с интерпретатором Python (символ «%» здесь означает строку приглашения к вводу, он не должен вводиться вами).

7. Почему язык программирования Python называется языком динамической типизации?

**Динамическая типизация** — приём, используемый в [языках программирования](#) и [языках спецификации](#), при котором [переменная](#) связывается с [типом](#) в момент [присваивания значения](#), а не в момент объявления переменной. Таким образом, в различных участках [программы](#) одна и та же переменная может принимать [значения](#) разных [типов](#). Примеры языков с динамической типизацией — [Smalltalk](#), [Python](#), [Objective-C](#), [Ruby](#), [PHP](#), [Perl](#), [JavaScript](#), [Лисп](#).

Динамическая типизация упрощает написание программ для работы с меняющимся окружением, при работе с данными переменных типов; при этом отсутствие информации о типе на этапе [компиляции](#) повышает вероятность ошибок в исполняемых модулях.

8. Какие существуют основные типы в языке программирования Python?

**К основным встроенным типам относятся:**

- None (неопределенное значение переменной)
- Логические переменные (Boolean Type)

- Числа (Numeric Type) int – целое число ...
- Списки (Sequence Type) list – список ...
- Строки (Text Sequence Type) str.
- Бинарные списки (Binary Sequence Types) ...
- Множества (Set Types) ...
- Словари (Mapping Types)

9. Как создаются объекты в памяти? Каково их устройство? В чем заключается процесс

объявления новых переменных и работа операции присваивания?

Рассмотрим как создаются объекты в памяти, их устройство, процесс объявления новых переменных и работу операции присваивания.

Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана. Например строка:

```
b = 5
```

объявляет переменную b и присваивает ей значение 5.

Целочисленное значение 5 в рамках языка Python по сути своей является объектом. Объект, в данном случае – это абстракция для представления данных, данные – это числа, списки, строки и т.п. При этом, под данными следует понимать как непосредственно сами объекты, так и отношения между ними (об этом чуть позже). Каждый объект имеет три атрибута – это идентификатор, значение и тип. Идентификатор – это уникальный признак объекта, позволяющий отличать объекты друг от друга, а значение – непосредственно информация, хранящаяся в памяти, которой управляет интерпретатор.

При инициализации переменной, на уровне интерпретатора, происходит следующее:

- создается целочисленный объект 5 (можно представить, что в этот момент создается ячейка и 5 кладется в эту ячейку);
- данный объект имеет некоторый идентификатор, значение: 5, и тип: целое число;
- посредством оператора “=” создается ссылка между переменной b и целочисленным объектом 5 (переменная b ссылается на объект 5).

10. Как получить список ключевых слов в Python?

```
>>> help()
```

**help > ключевые слова**

Вот список ключевых слов Python. Введите любое ключевое слово, чтобы получить дополнительную помощь.

False def if raise

None del import return

True elif in try

and else is while

as except lambda with

assert finally nonlocal yield

break for not

class from or

continue global pass