

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

Кафедра инфокоммуникаций

**Отчет по лабораторной работе № 2.1
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Тема: Основы языка Python.

Цель работы: исследование процесса установки и базовых возможностей языка Python версии 3.x.

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 MrPlatynum ▾

Repository name *

ProgrammEngineering4

✔ ProgrammEngineering4 is available.

Great repository names are short and memorable. Need inspiration? How about [fantastic-carnival](#) ?

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:

☐ **Add a README file**

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

Add .gitignore

.gitignore template: **Python** ▾

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

Choose a license

License: **MIT License** ▾

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

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

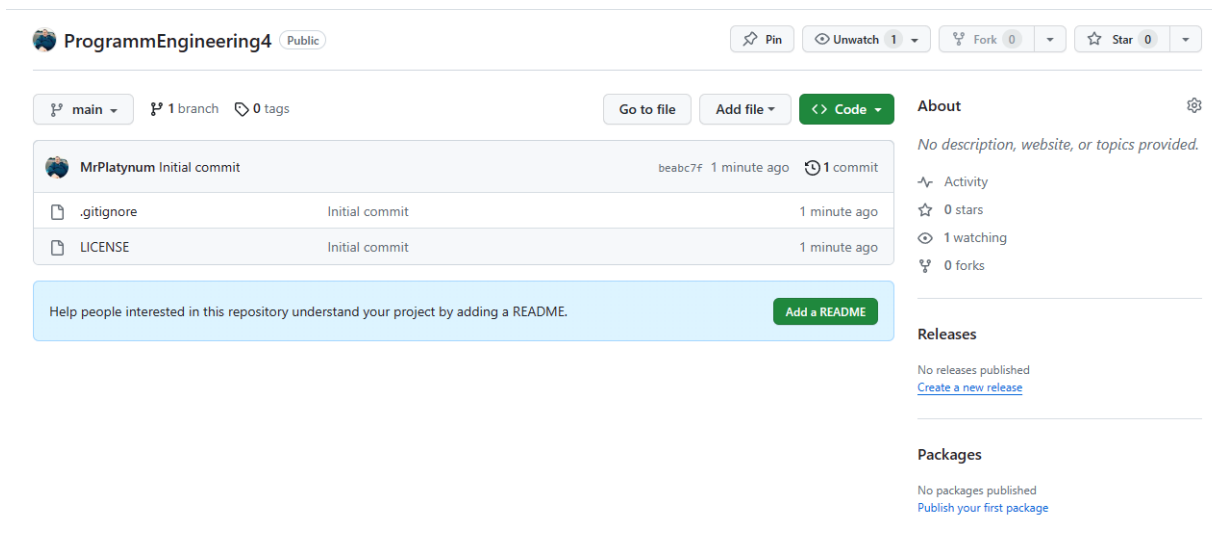


Рисунок 2 – Результат создания репозитория

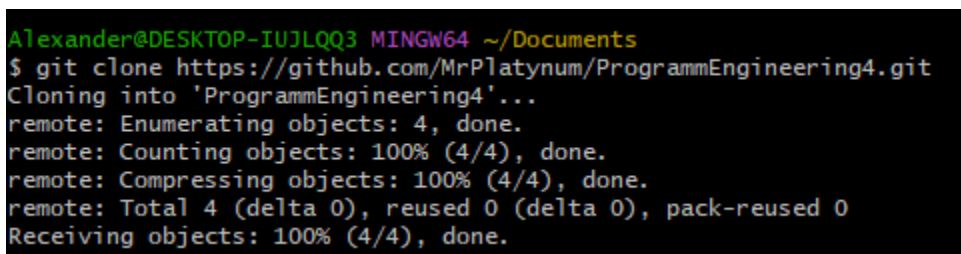


Рисунок 3 – Клонирование созданного репозитория на локальный компьютер

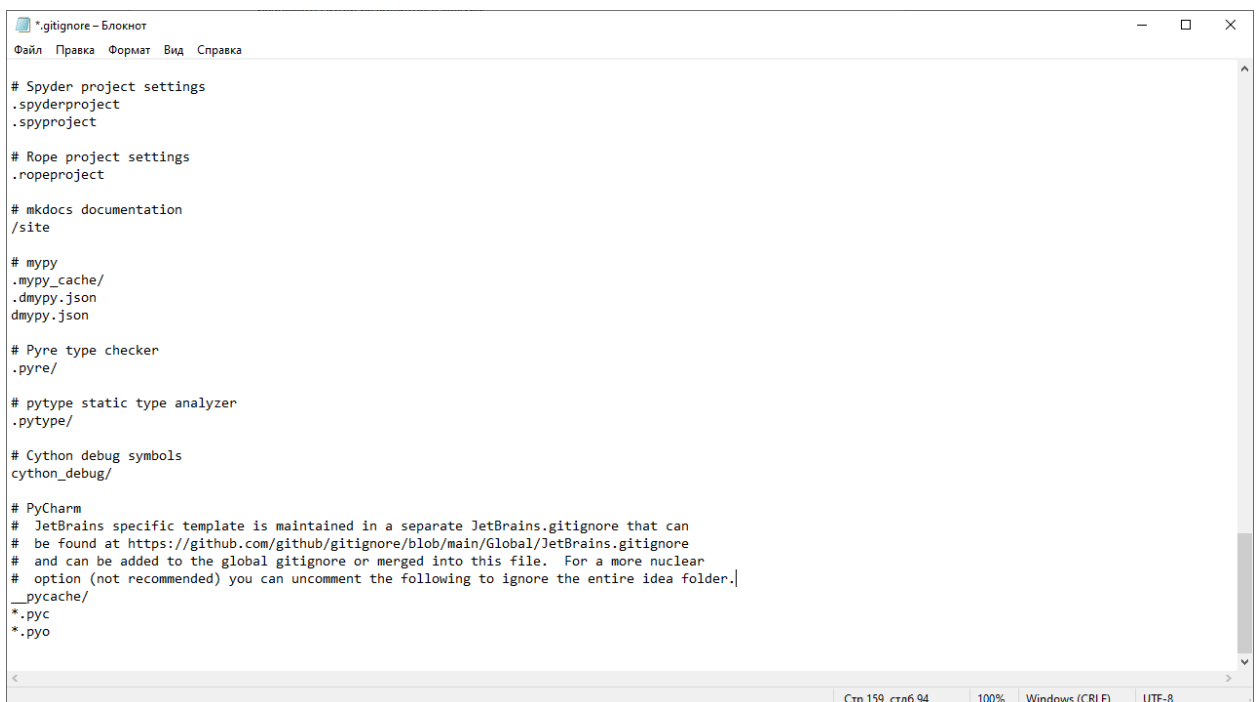


Рисунок 4 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (main)
$ git branch develop

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (main)
$ git checkout develop
Switched to branch 'develop'
M       .gitignore

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (develop)
$
```

Рисунок 5 – организация репозитория в соответствии с моделью ветвления git flow

2. Создадим проект PyCharm в репозитории:

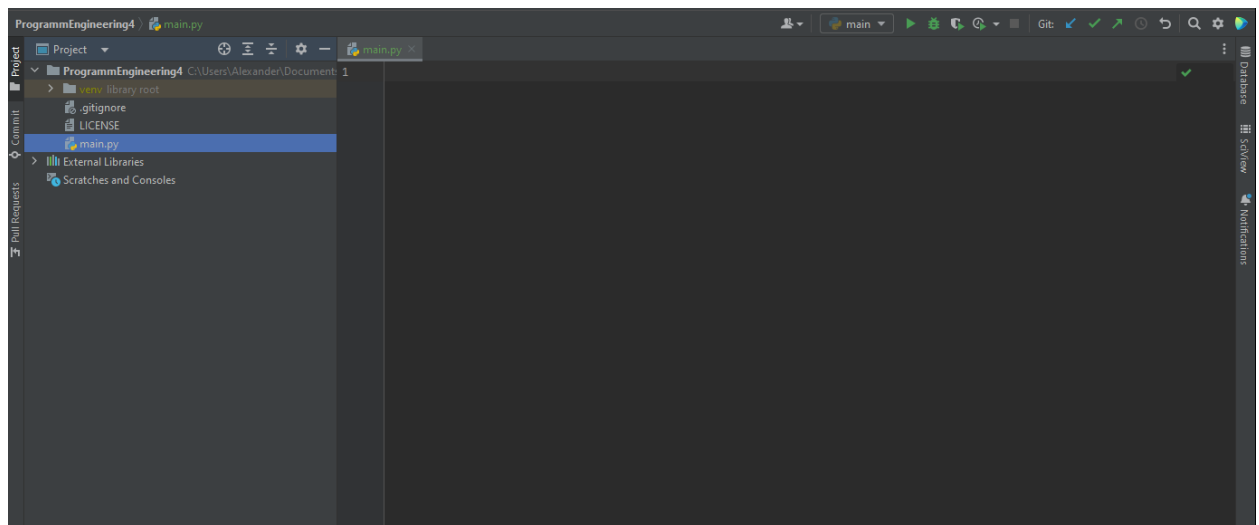
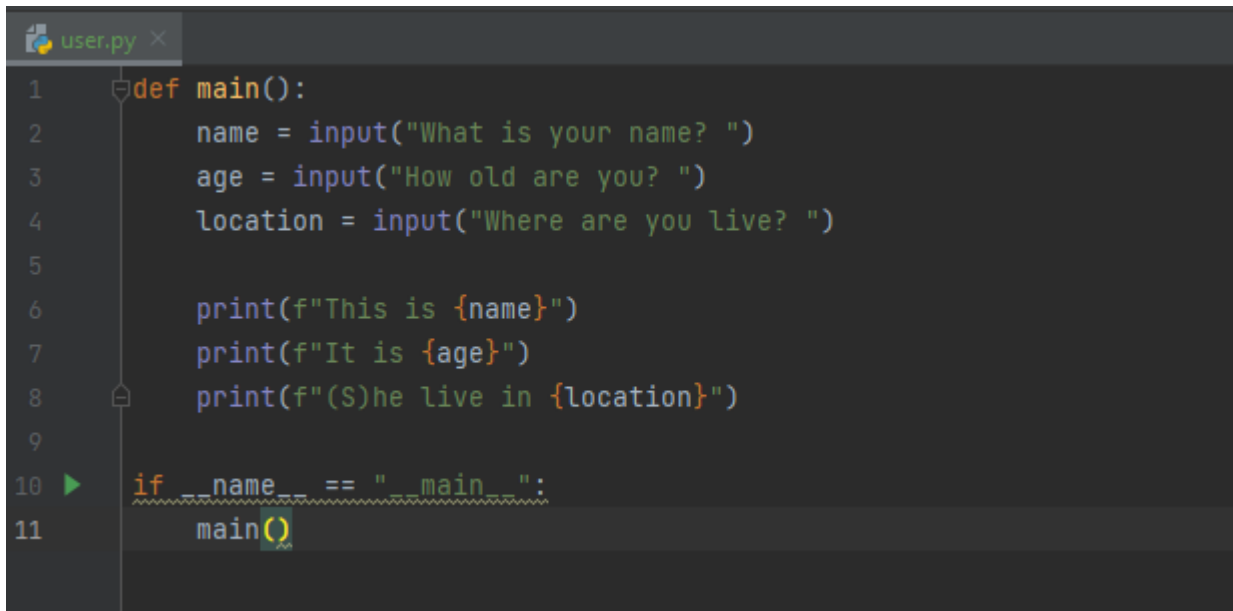


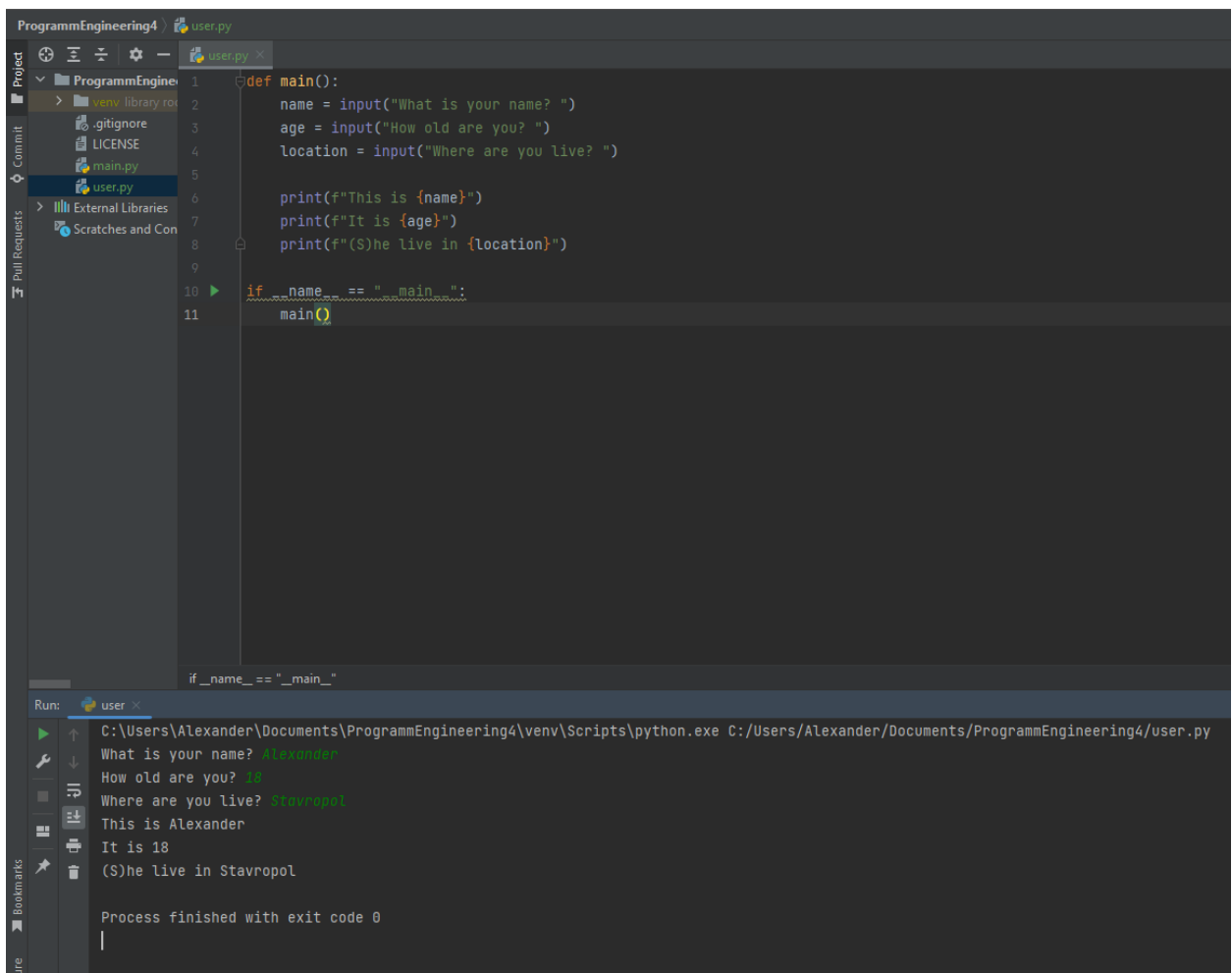
Рисунок 6 – Проект PyCharm

3. Напишите программу (user.py), которая запрашивала бы у пользователя имя, возраст, место жительства и выводила бы их на экран:



```
1 def main():
2     name = input("What is your name? ")
3     age = input("How old are you? ")
4     location = input("Where are you live? ")
5
6     print(f"This is {name}")
7     print(f"It is {age}")
8     print(f"(S)he live in {location}")
9
10 if __name__ == "__main__":
11     main()
```

Рисунок 7 – Программа вывода данных о пользователе



```
ProgrammEngineering4 > user.py
1 def main():
2     name = input("What is your name? ")
3     age = input("How old are you? ")
4     location = input("Where are you live? ")
5
6     print(f"This is {name}")
7     print(f"It is {age}")
8     print(f"(S)he live in {location}")
9
10 if __name__ == "__main__":
11     main()

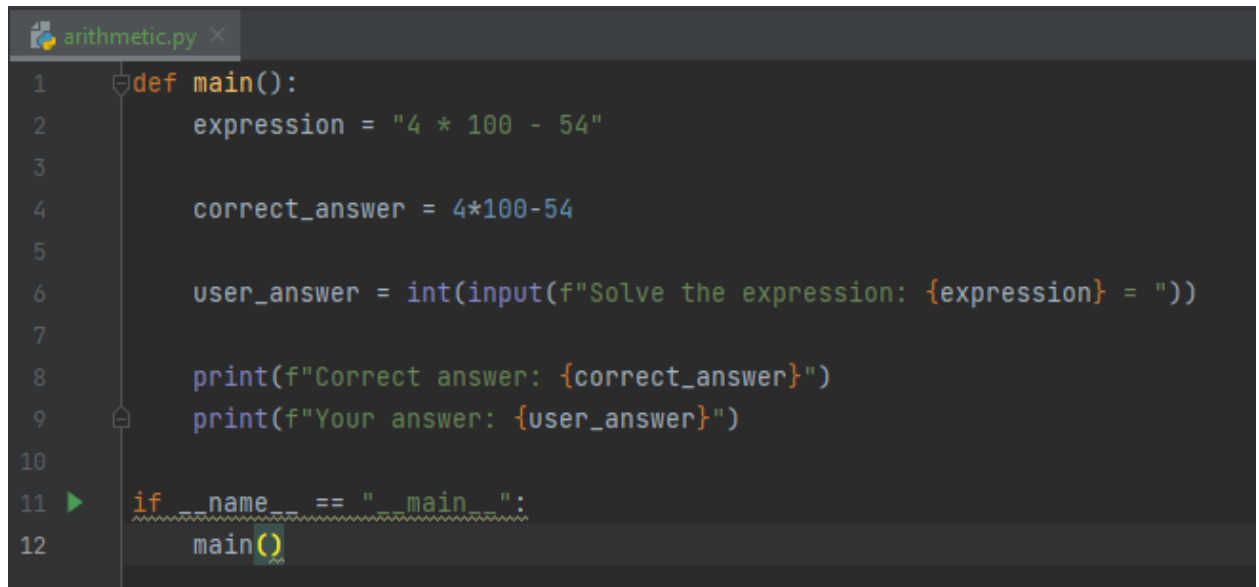
Run: user
C:\Users\Alexander\Documents\ProgrammEngineering4\venv\Scripts\python.exe C:/Users/Alexander/Documents/ProgrammEngineering4/user.py
What is your name? Alexander
How old are you? 18
Where are you live? Stavropol
This is Alexander
It is 18
(S)he live in Stavropol

Process finished with exit code 0
```

Рисунок 8 – Запуск программы

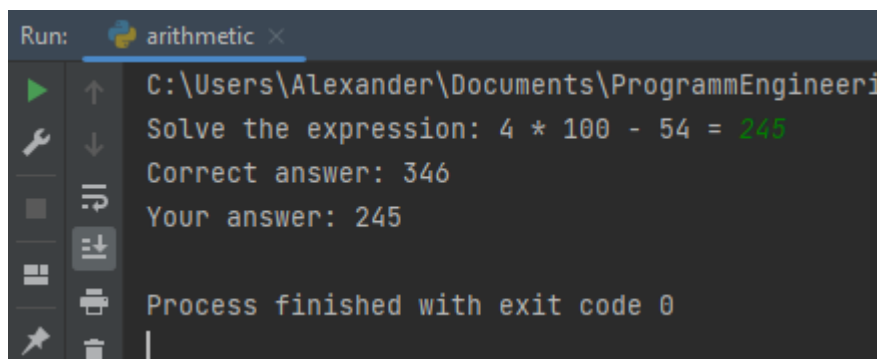
4. Напишите программу (файл arithmetic.py), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран

правильный ответ и ответ пользователя:

A screenshot of a code editor window titled 'arithmetic.py'. The code is as follows:

```
1 def main():
2     expression = "4 * 100 - 54"
3
4     correct_answer = 4*100-54
5
6     user_answer = int(input(f"Solve the expression: {expression} = "))
7
8     print(f"Correct answer: {correct_answer}")
9     print(f"Your answer: {user_answer}")
10
11 if __name__ == "__main__":
12     main()
```

Рисунок 6 – Программа проверки решения математического примера

A screenshot of a terminal window titled 'Run: arithmetic'. It shows the output of the program:

```
C:\Users\Alexander\Documents\ProgrammEngineeri
Solve the expression: 4 * 100 - 54 = 245
Correct answer: 346
Your answer: 245

Process finished with exit code 0
```

Рисунок 7 – Запуск программы

5. 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:

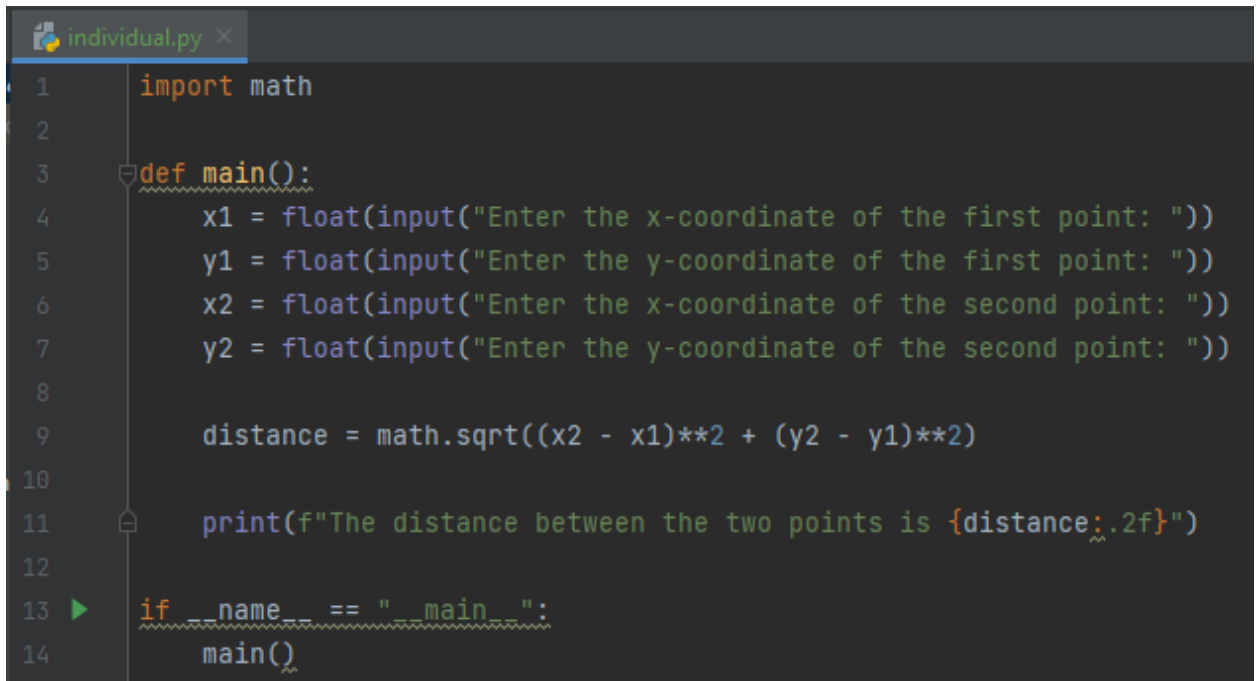
```
numbers.py x
1 def main():
2     num1 = float(input("Enter the first number: "))
3     num2 = float(input("Enter the second number: "))
4     num3 = float(input("Enter the third number: "))
5     num4 = float(input("Enter the fourth number: "))
6
7     sum1 = num1 + num2
8     sum2 = num3 + num4
9
10    result = sum1 / sum2
11    print(f"Result: {result:.2f}")
12
13 if __name__ == "__main__":
14     main()
```

Рисунок 8 – Программа, запрашивающая у пользователя числа и выводящая результат деления первой и последней сумм

```
Run: numbers x
C:\Users\Alexander\Documents\Programme
Enter the first number: 1
Enter the second number: 2
Enter the third number: 3
Enter the fourth number: 4
Result: 0.43
Process finished with exit code 0
```

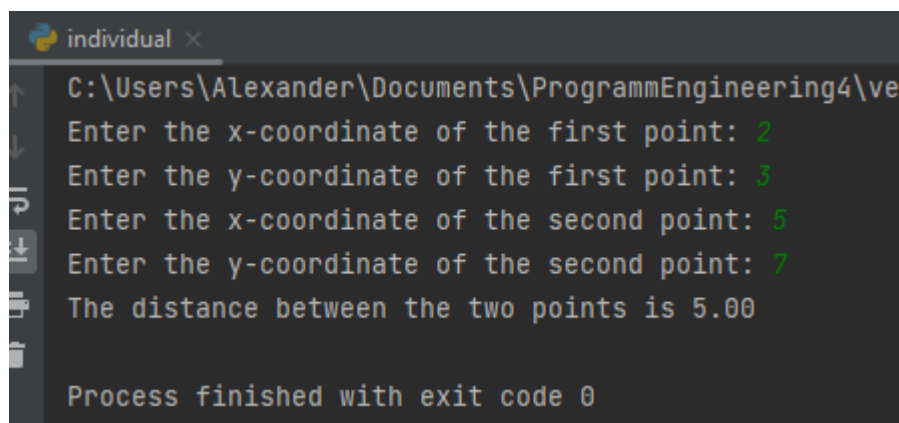
Рисунок 12 – Запуск программы

6. Напишите программу (файл individual.py) для решения индивидуального задания. Даны координаты на плоскости двух точек. Найти расстояние между этими точками.:



```
1 import math
2
3 def main():
4     x1 = float(input("Enter the x-coordinate of the first point: "))
5     y1 = float(input("Enter the y-coordinate of the first point: "))
6     x2 = float(input("Enter the x-coordinate of the second point: "))
7     y2 = float(input("Enter the y-coordinate of the second point: "))
8
9     distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
10
11     print(f"The distance between the two points is {distance:.2f}")
12
13 if __name__ == "__main__":
14     main()
```

Рисунок 9 – Программа нахождения расстояния между точками

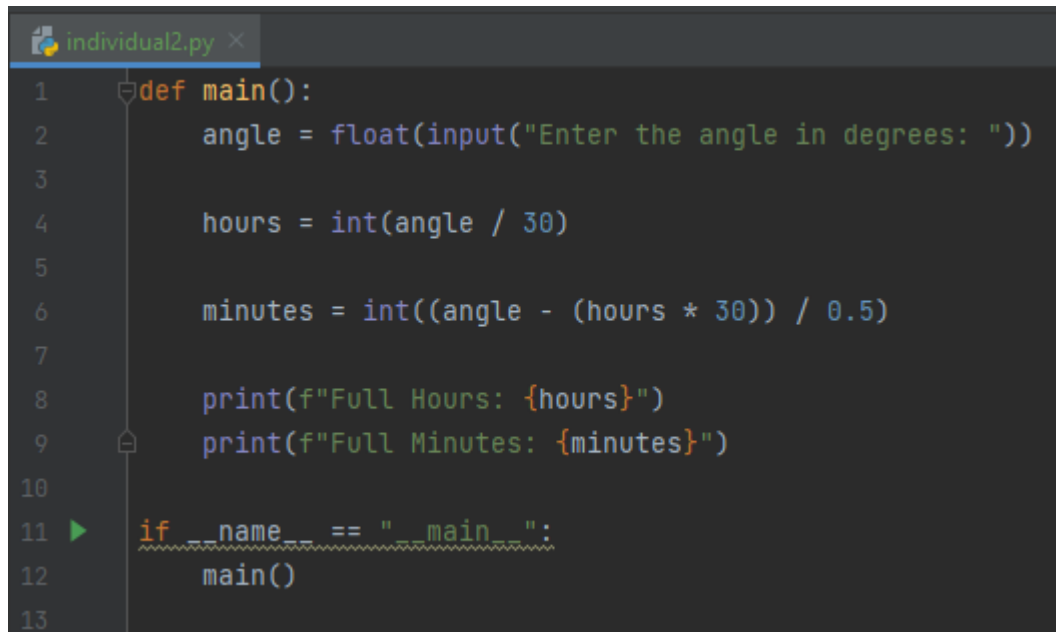


```
individual ×
C:\Users\Alexander\Documents\ProgrammEngineering4\ve
Enter the x-coordinate of the first point: 2
Enter the y-coordinate of the first point: 3
Enter the x-coordinate of the second point: 5
Enter the y-coordinate of the second point: 7
The distance between the two points is 5.00

Process finished with exit code 0
```

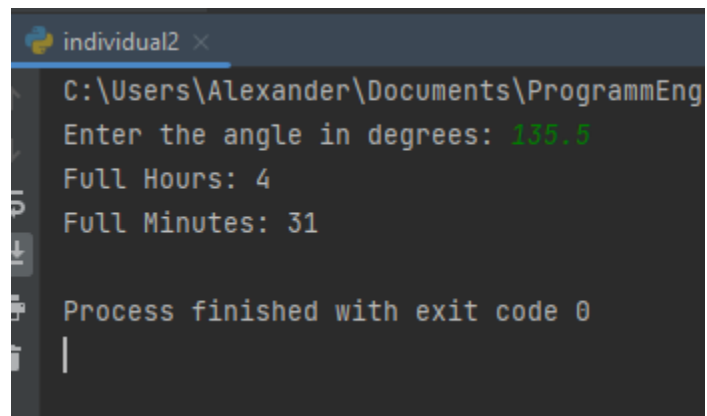
Рисунок 10 – Запуск программы

7. Задача повышенной сложности. С начала суток часовая стрелка повернулась на y градусов ($0 \leq y < 360$, y – вещественное число). Определить число полных часов и число полных минут, прошедших с начала суток. Условный оператор не использовать:

A screenshot of a code editor window titled 'individual2.py'. The code is a Python script with 13 lines. Line 1: 'def main():'. Line 2: 'angle = float(input("Enter the angle in degrees: "))'. Line 3: (empty). Line 4: 'hours = int(angle / 30)'. Line 5: (empty). Line 6: 'minutes = int((angle - (hours * 30)) / 0.5)'. Line 7: (empty). Line 8: 'print(f"Full Hours: {hours}")'. Line 9: 'print(f"Full Minutes: {minutes}")'. Line 10: (empty). Line 11: 'if __name__ == "__main__":'. Line 12: ' main()'. Line 13: (empty).

```
1 def main():
2     angle = float(input("Enter the angle in degrees: "))
3
4     hours = int(angle / 30)
5
6     minutes = int((angle - (hours * 30)) / 0.5)
7
8     print(f"Full Hours: {hours}")
9     print(f"Full Minutes: {minutes}")
10
11 if __name__ == "__main__":
12     main()
13
```

Рисунок 11 – Программа для определения числа полных часов и минут

A screenshot of a terminal window titled 'individual2'. It shows the execution of the program. The first line is the file path 'C:\Users\Alexander\Documents\ProgrammEng'. The second line is the prompt 'Enter the angle in degrees:' followed by the user input '135.5'. The third line is the output 'Full Hours: 4'. The fourth line is the output 'Full Minutes: 31'. The fifth line is the message 'Process finished with exit code 0'.

```
individual2
C:\Users\Alexander\Documents\ProgrammEng
Enter the angle in degrees: 135.5
Full Hours: 4
Full Minutes: 31
Process finished with exit code 0
|
```

Рисунок 12 – Запуск программы

8. Коммит всех созданных файлов:

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (develop)
$ git log
commit e1dfb8db8a79c6b48f1c760d601e7b28fcea5236 (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date:   Wed Nov 8 00:57:20 2023 +0300

    Добавлена программа повышенного уровня

commit d2aed41bcf171468a4be74a13200f089916e2c13
Author: MrPlatynum <dushinsasha4@gmail.com>
Date:   Wed Nov 8 00:56:28 2023 +0300

    Добавлены файлы user.py, arithmetic.py, numbers.py, individual.py в ветку для разработки

commit beabc7f2ccc788137f27a680b368ef1956f77104 (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date:   Tue Nov 7 21:33:24 2023 +0300

    Initial commit

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (develop)
$ |

```

Рисунок 17 – Коммиты ветки develop во время выполнения лабораторной работы

9. Слияние ветки develop в ветку main и отправка на удаленный репозиторий:

```

$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (main)
$ git merge develop
Updating beabc7f..e1dfb8d
Fast-forward
 arithmetic.py | 12 ++++++++
 individual.py | 14 ++++++++
 individual2.py | 15 ++++++++
 main.py       |  0
 numbers.py    | 14 ++++++++
 user.py       | 11 ++++++++
 6 files changed, 66 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 individual2.py
 create mode 100644 main.py
 create mode 100644 numbers.py
 create mode 100644 user.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (main)
$ |

```

Рисунок 18 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering4 (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 1.66 KiB | 1.66 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MrPlatynum/ProgrammEngineering4.git
   beabc7f..e1dfb8d  main -> main
```

Рисунок 19 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

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

Установка дистрибутива Python в Windows осуществляется с помощью исполняемого или архивного файла, скачанными из официального сайта Python.

Чаще всего интерпретатор Python уже входит в состав дистрибутива. Если нужно установить вручную, то можно воспользоваться командой: `sudo apt-get install python3`.

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

Anaconda включает в себя интерпретатор языка Python (есть версии 2 и 3), набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере.

3. Как осуществить проверку работоспособности пакета Anaconda?

Можно воспользоваться программой Anaconda Navigator, которая устанавливается вместе с Anaconda.

4. Как задать используемый интерпретатор языка Python в IDE PyCharm? В настройках проекта.

5. Как осуществить запуск программы с помощью IDE PyCharm?

Открыть файл или проект с помощью PyCharm. Выбрать интерпретатор и запустить файл.

6. В чем суть интерактивного и пакетного режимов работы Python? Интерактивным режимом можно воспользоваться из командной строки.

Пакетный режим использует файлы с расширением `.py`.

7. Почему язык программирования Python называется языком динамической типизации?

Потому что тип инициализированных объектов может меняться в процессе выполнения программы.

8. Какие существуют основные типы в языке программирования Python?

Численные – int, float, complex. Строковые – str. Логические – bool. Списки – list, tuple, range. Бинарные списки – bytes, bytearray, memoryview. Множества – set, frozenset. Словари – dict. None.

9. Как создаются объекты в памяти? Каково их устройство? В чем заключается процесс объявления новых переменных и работа операции присваивания?

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 объект (можно представить, что в этот момент создается ячейка и значение кладется в эту ячейку); данный объект имеет некоторый идентификатор, значение и тип; посредством оператора “=” создается ссылка между переменной и объектом.

10. Как получить список ключевых слов в Python?

Для этого нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

11. Каково назначение функций id() и type()?

Для того, чтобы посмотреть на объект с каким идентификатором ссылается данная переменная, можно использовать функцию id().

Тип переменной можно определить с помощью функции type().

12. Что такое изменяемые и неизменяемые типы в Python.

К неизменяемым (immutable) типам относятся: целые числа (int), числа с плавающей точкой (float), комплексные числа (complex), логические переменные (bool), кортежи (tuple), строки (str) и неизменяемые множества (frozenset). К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

13. Чем отличаются операции деления и целочисленного деления?

Целочисленное деление возвращает целую часть от деления.

14. Какие имеются средства в языке Python для работы с комплексными числами?

Для создания комплексного числа можно использовать функцию `complex(a, b)`, в которую, в качестве первого аргумента, передается действительная часть, в качестве второго – мнимая. Либо записать число в виде $a + bj$. Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень. Для получения комплексносопряженного числа необходимо использовать метод `conjugate()`. `.real` – действительная часть, `.imag` – мнимая часть.

15. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`.

В стандартную поставку Python входит библиотека `math`, в которой содержится большое количество часто используемых математических функций. В языке программирования Python для работы с комплексными числами используется модуль `cmath`. Модуль содержит набор функций для обработки комплексных чисел. `cmath.phase(x)` — возвращает фазу от аргумента `x` в виде числа типа `float`; `cmath.polar()` — возвращает представление `x` в полярных координатах; `cmath.rect()` — возвращает комплексное число из полярных координат; `cmath.exp(x)` — возвращает экспоненту `e`, возведенную в степень `x`, где `x` может быть комплексным числом. Экспонента `e` является основой натурального логарифма; `cmath.atan(x)` — определяет арктангенс от аргумента `x` и т. д.

16. Каково назначение именованных параметров `sep` и `end` в функции `print()`? Параметр `end` позволяет указывать, что делать, после вывода строки. По умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку. Через параметр `sep` можно указать отличный от пробела разделитель строк.

17. Каково назначение метода `format()`? Какие еще существуют средства для форматирования строк в Python? Примечание: в дополнение к

рассмотренным средствам изучите самостоятельно работу с f-строками в Python.

Метод `format()` в Python используется для вставки значений переменных в строку, заменяя плейсхолдеры (обычно фигурные скобки `{}`) соответствующими значениями. Это позволяет создавать динамические строки, где значения переменных могут быть динамически вставлены в определенные позиции в строке. Помимо метода `format()`, в Python существуют другие способы форматирования строк, включая использование f-строк (f-strings), которые предоставляют более удобный и читаемый способ вставки значений переменных в строки. F-строки доступны в Python 3.6 и более новых версиях и позволяют вставлять значения переменных непосредственно в строку с использованием префикса `f` перед строкой и фигурных скобок `{}` для обозначения плейсхолдеров.

18. Каким образом осуществить ввод с консоли значения целочисленной вещественной переменной в языке Python?

`int(input())` или `float(input())`.