

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.1
дисциплины «Основы программной инженерии»

Выполнил:
Соколов Михаил Романович
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Основы языка Python.

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

Ход выполнения работы:

1. Установить Python версии 3.10.6:

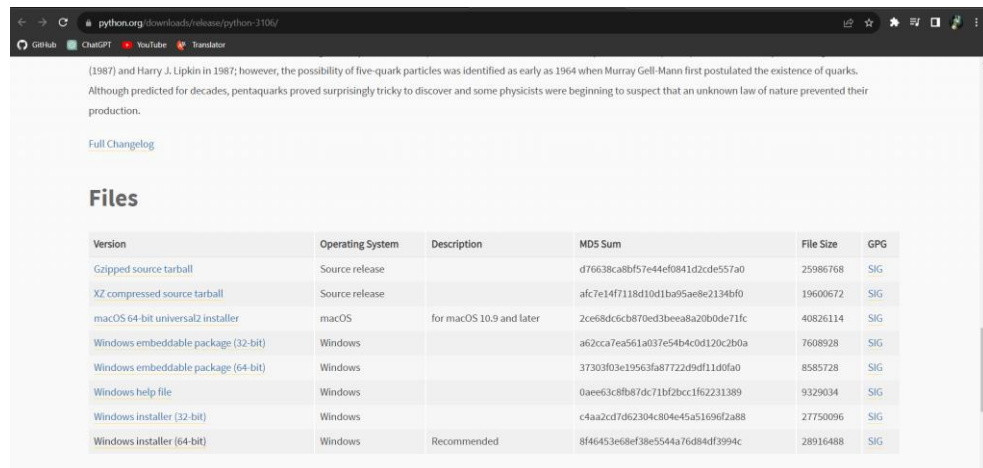


Рисунок 1 – Сайт для загрузки установщика Python 3.10.6
(<https://www.python.org/downloads/release/python-3106/>)



Рисунок 2 – Окно установки Python 3.10.6

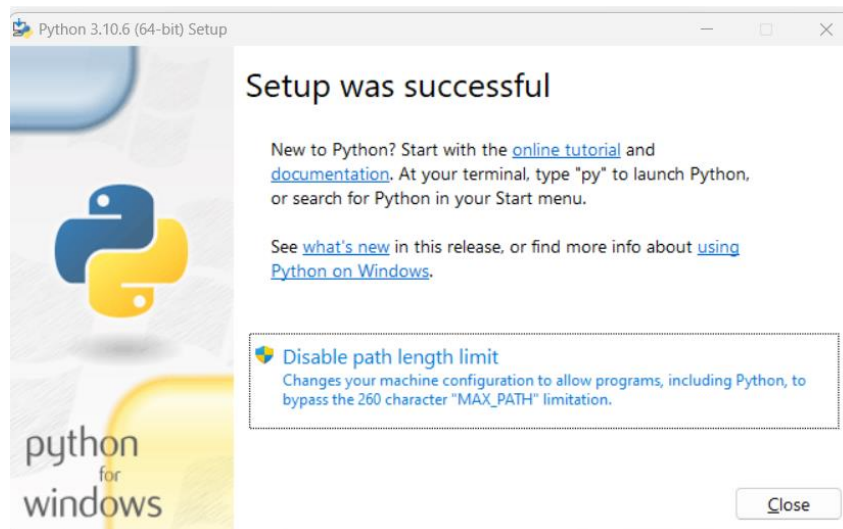


Рисунок 3 – Окно, уведомляющее о успешной установке Python 3.10.6

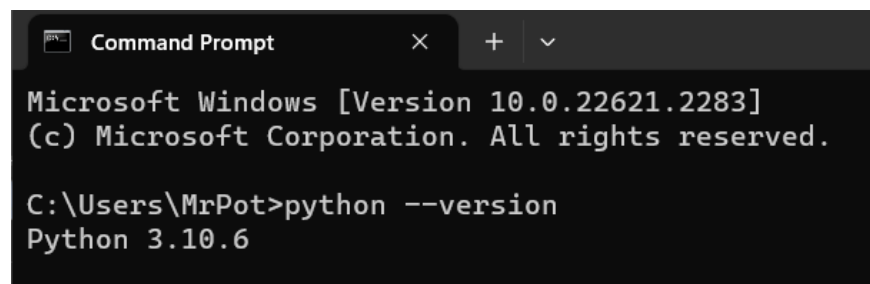


Рисунок 4 – Проверка установки

2. Установить Anaconda:

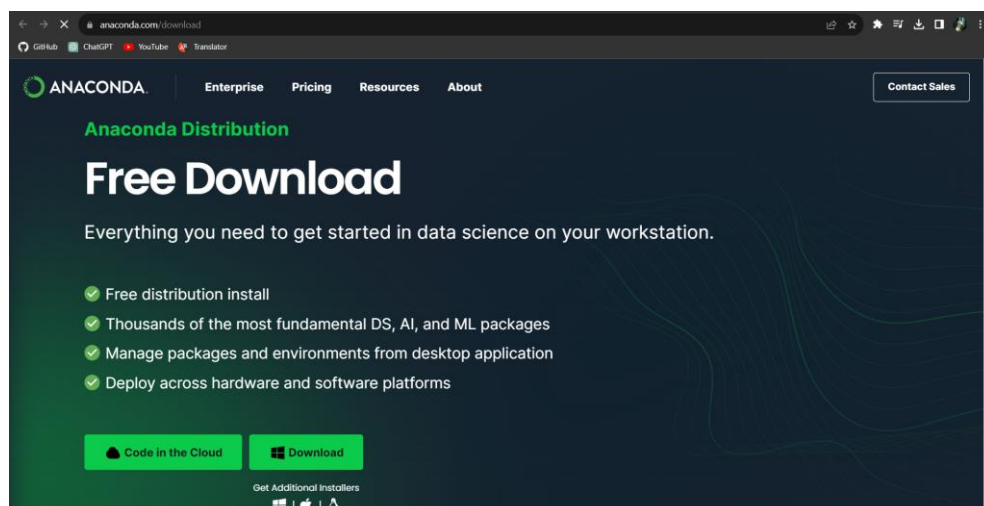


Рисунок 5 – Сайт установки Anaconda (<https://www.anaconda.com/download>)

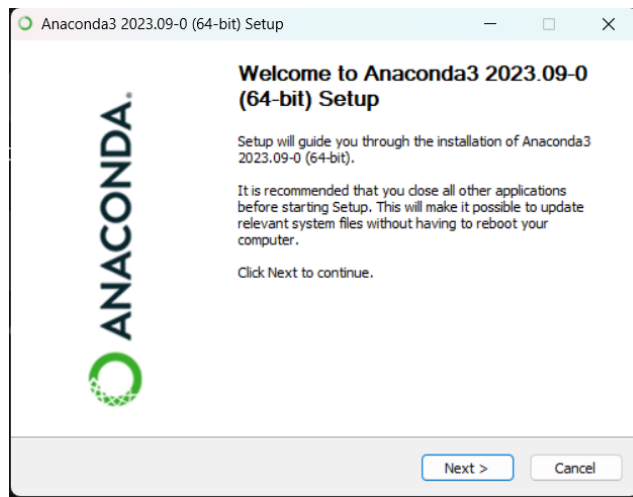


Рисунок 6 – Окно установки Anaconda

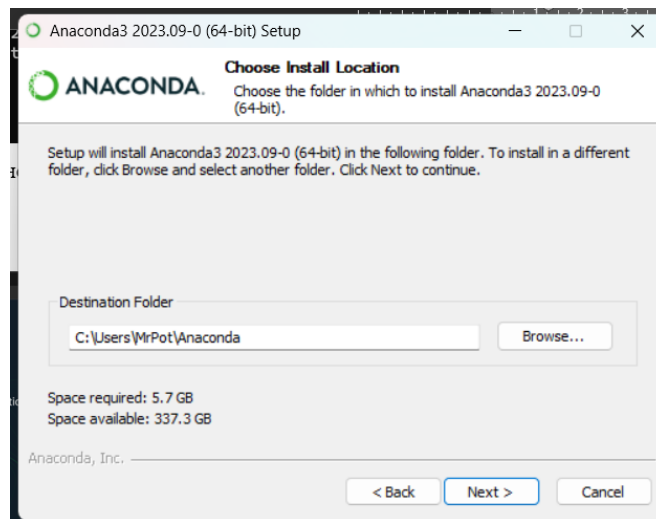


Рисунок 7 – Выбор папки установки Anaconda

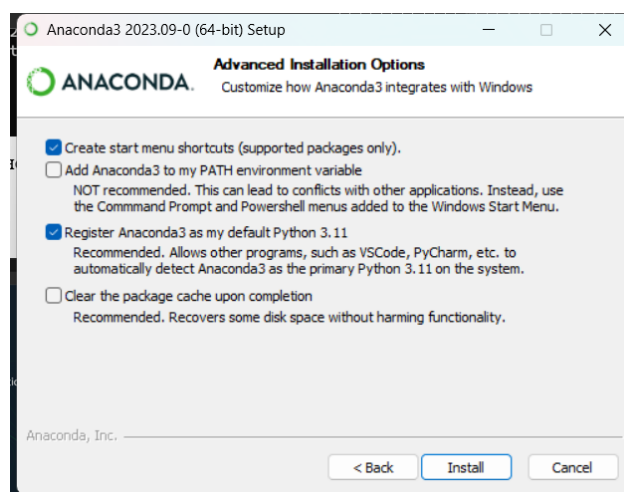


Рисунок 8 – Выбор опциональный настроек установки Anaconda

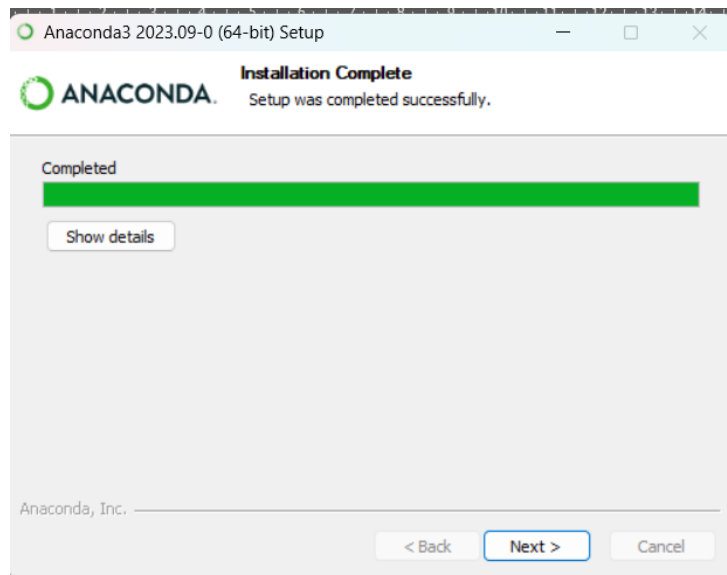


Рисунок 9 – Окно завершения установки Anaconda

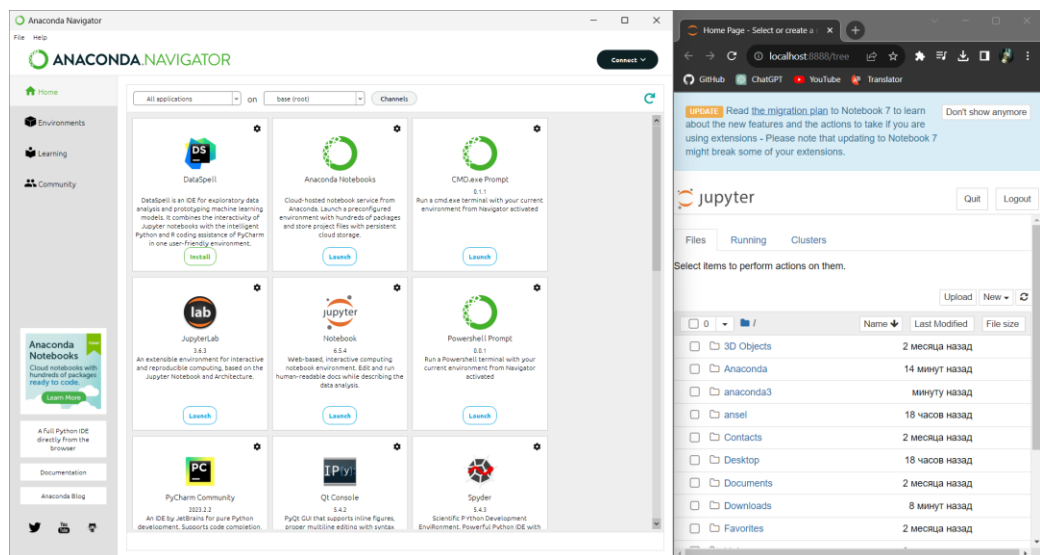


Рисунок 10 – Проверка работоспособности Anaconda

3. Установить PyCharm:

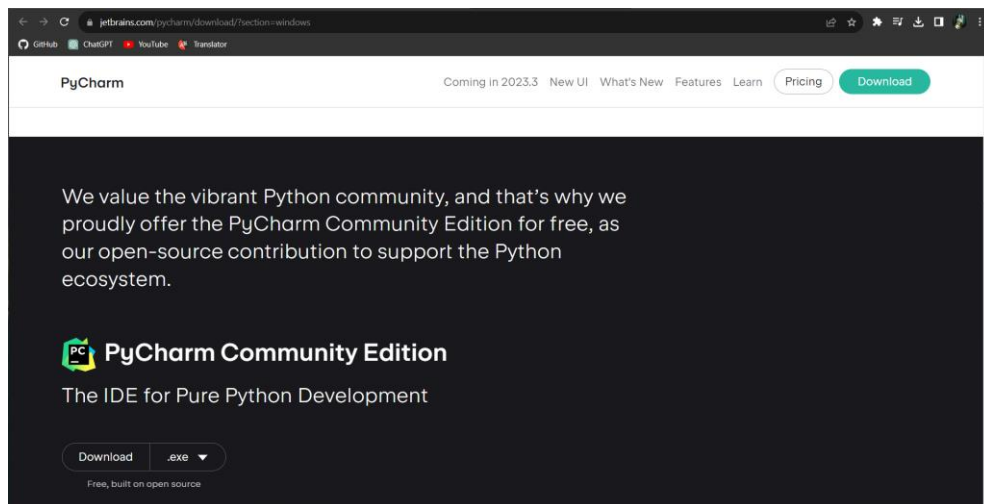


Рисунок 11 – Сайт установки PyCharm

(<https://www.jetbrains.com/pycharm/download/?section=windows>)

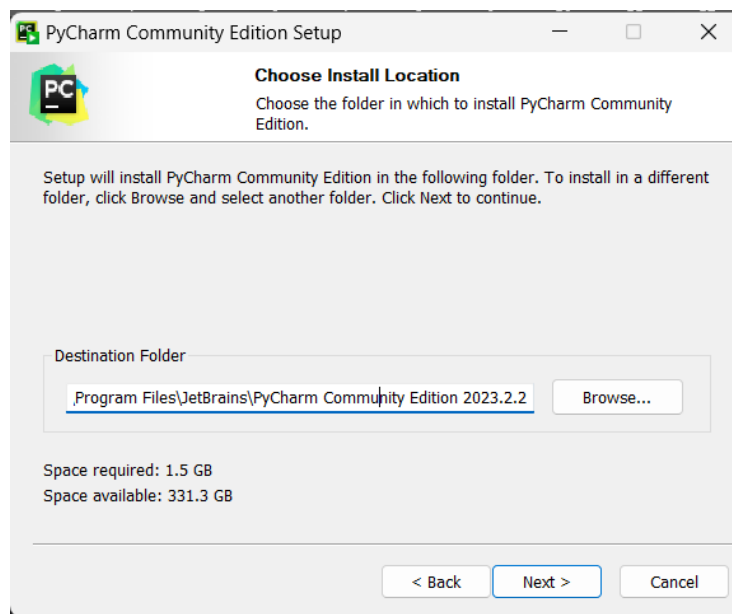


Рисунок 12 – Окно установки PyCharm

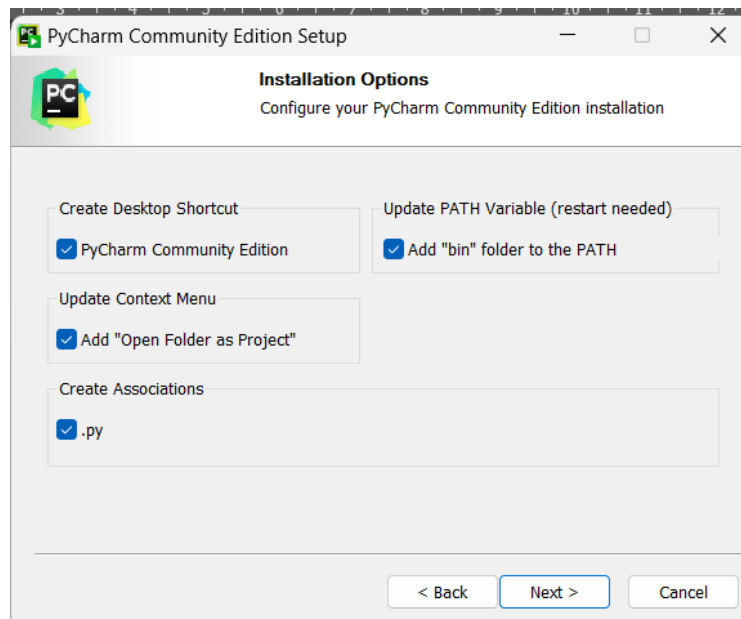


Рисунок 13 – Дополнительные опции установки PyCharm

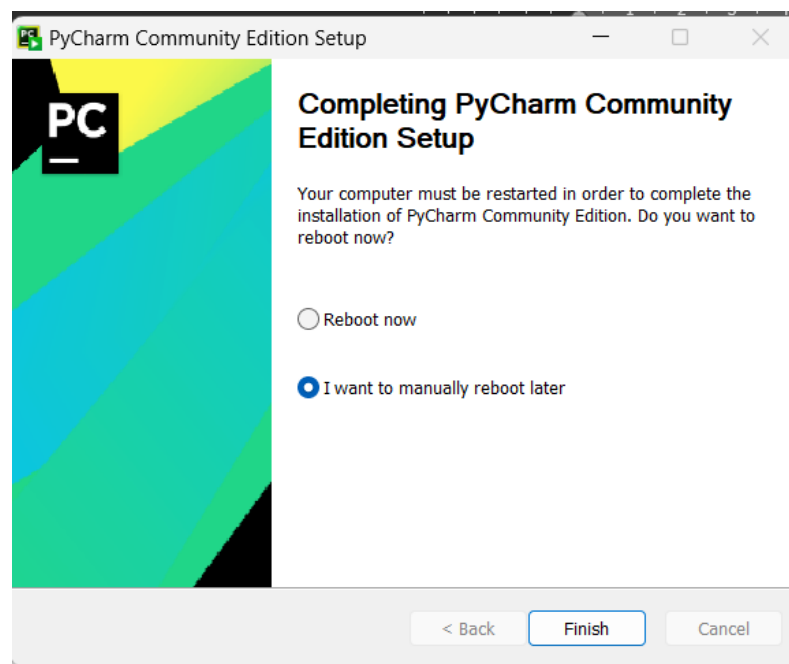


Рисунок 14 – Окно завершения установки PyCharm

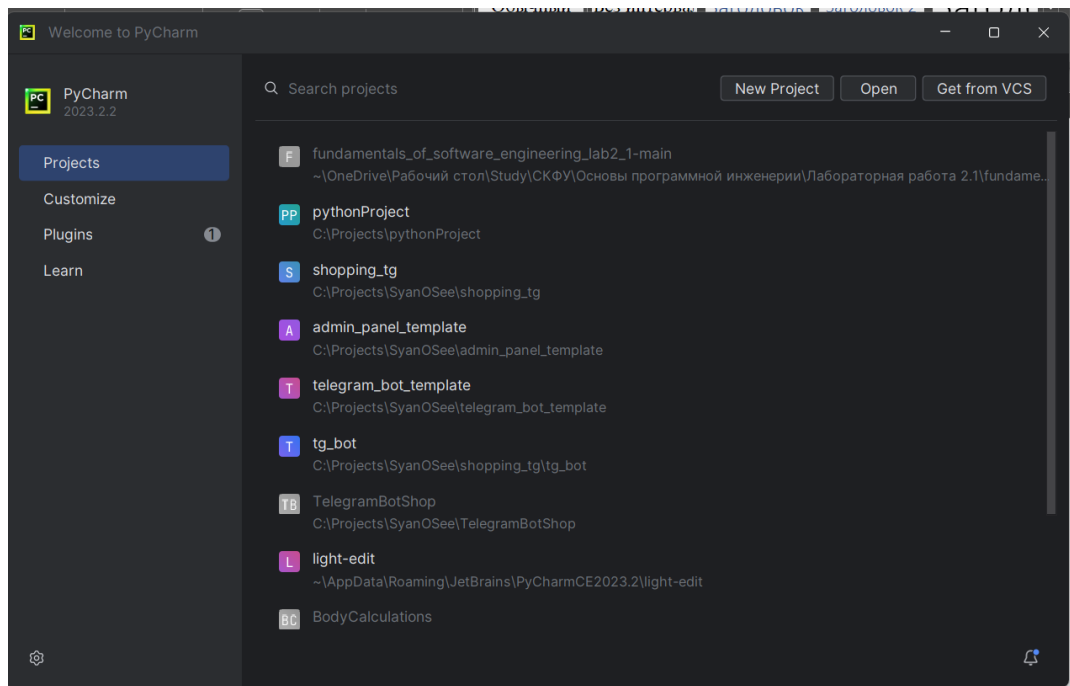


Рисунок 15 – Проверка работоспособности PyCharm

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

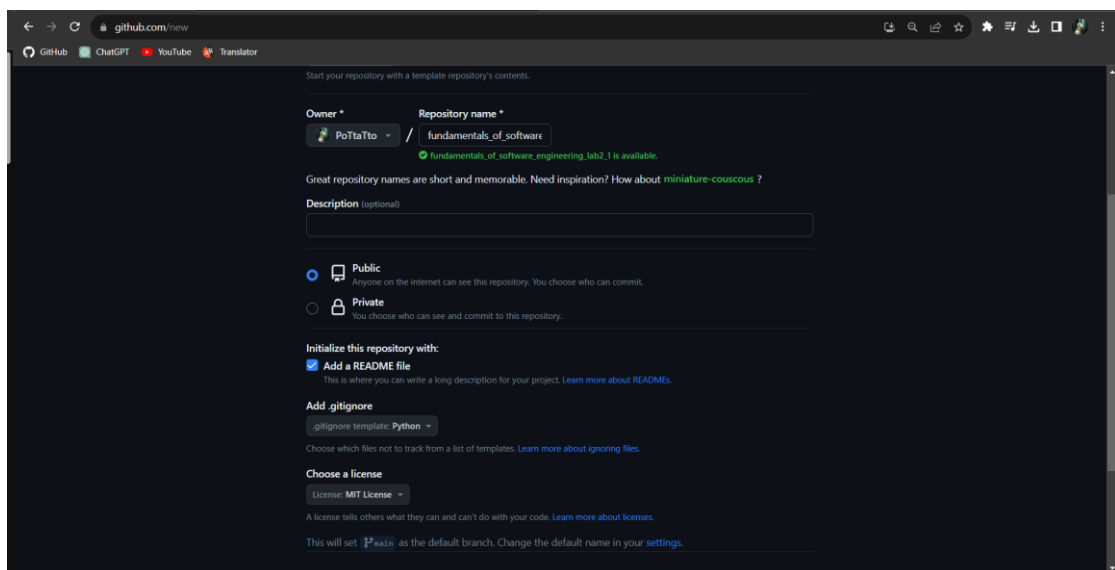


Рисунок 16 – Создание репозитория с заданными настройками

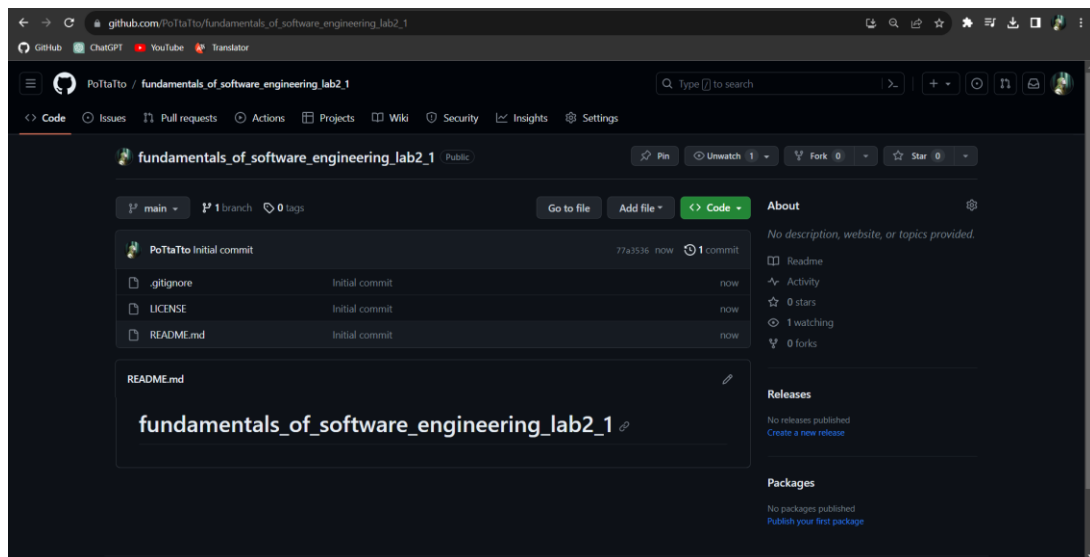


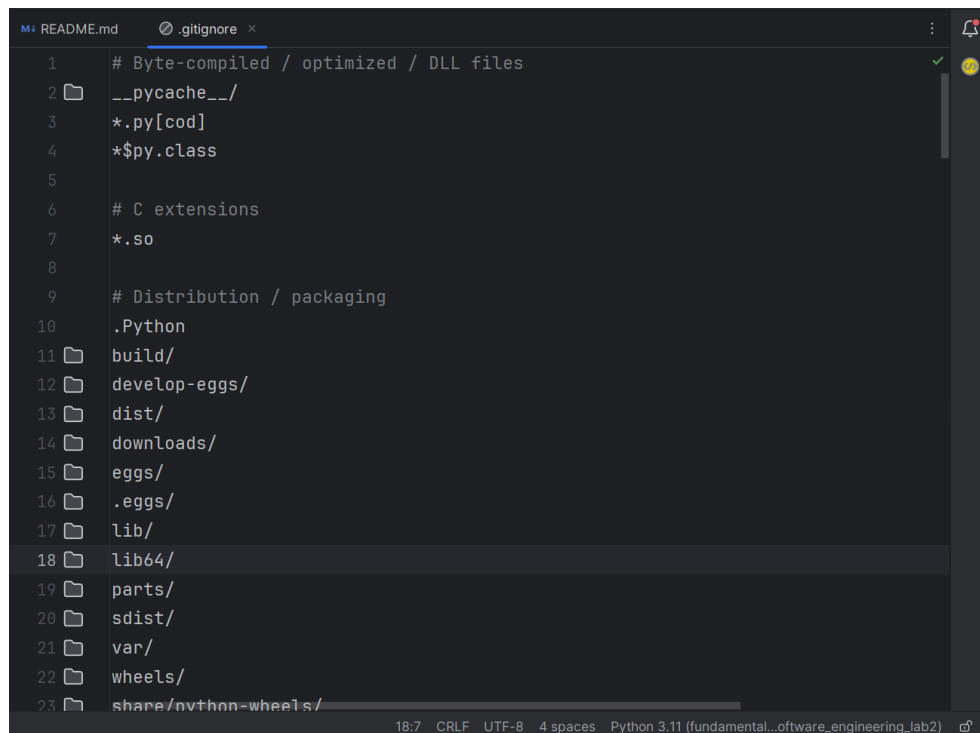
Рисунок 17 – Созданный репозиторий

```
PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2_1
Cloning into 'fundamentals_of_software_engineering_lab2_1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1> |
```

Рисунок 18 – Клонирование репозитория

```
PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1> |
```

Рисунок 19 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

A screenshot of a code editor showing a .gitignore file. The file contains a list of patterns to be ignored by Git, including compiled Python files, C extensions, and various build and distribution directories. The editor has a dark theme and a sidebar on the right with icons for search, run, and other functions.

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

Рисунок 20 – Часть файла .gitignore, созданный в GitHub

5. Создадим проект PyCharm в репозитории:

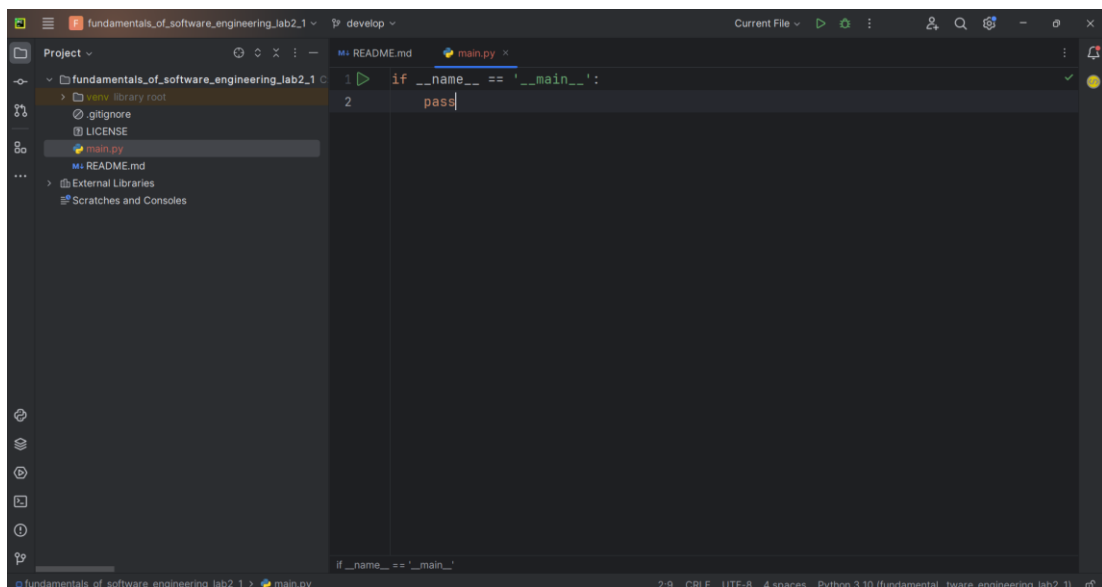


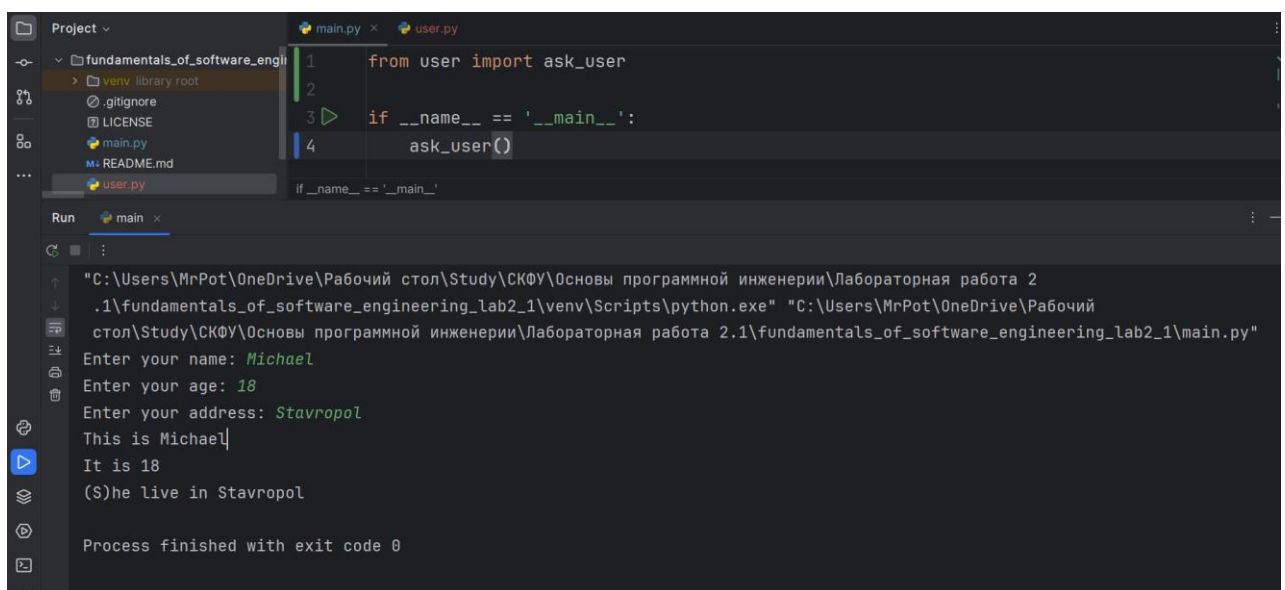
Рисунок 21 – Проект PyCharm

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



```
main.py user.py x
2 usages
1 def ask_user():
2     name = input('Enter your name: ')
3     age = input('Enter your age: ')
4     address = input('Enter your address: ')
5
6     print(f'This is {name}\nIt is {age}\n(S)he live in {address}')
7
```

Рисунок 22 – Функция вывода данных о пользователе



```
Project
fundamentals_of_software_engi
venv library root
.gitignore
LICENSE
main.py
README.md
user.py

main.py x user.py
1 from user import ask_user
2
3 if __name__ == '__main__':
4     ask_user()

if __name__ == '__main__':

Run main x
"C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2
.1\fundamentals_of_software_engineering_lab2_1\venv\Scripts\python.exe" "C:\Users\MrPot\OneDrive\Рабочий
стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\main.py"
Enter your name: Michael
Enter your age: 18
Enter your address: Stavropol
This is Michael\
It is 18
(S)he live in Stavropol

Process finished with exit code 0
```

Рисунок 23 – Запуск программы

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

```
main.py arithmetic.py x
1 usage MrPotatsus *
2 def solve_math_problem():
3     print('Task: 4 * 100 - 54')
4     user_answer = int(input('Enter your answer: '))
5     right_answer = 4 * 100 - 54
6     print(f'Your answer: {user_answer}\nRight answer: {right_answer}')
7     if user_answer == right_answer:
8         print('You\'re right!')
9     else:
10        print('Your answer is wrong!')
```

Рисунок 24 – Функция проверки решения математического примера

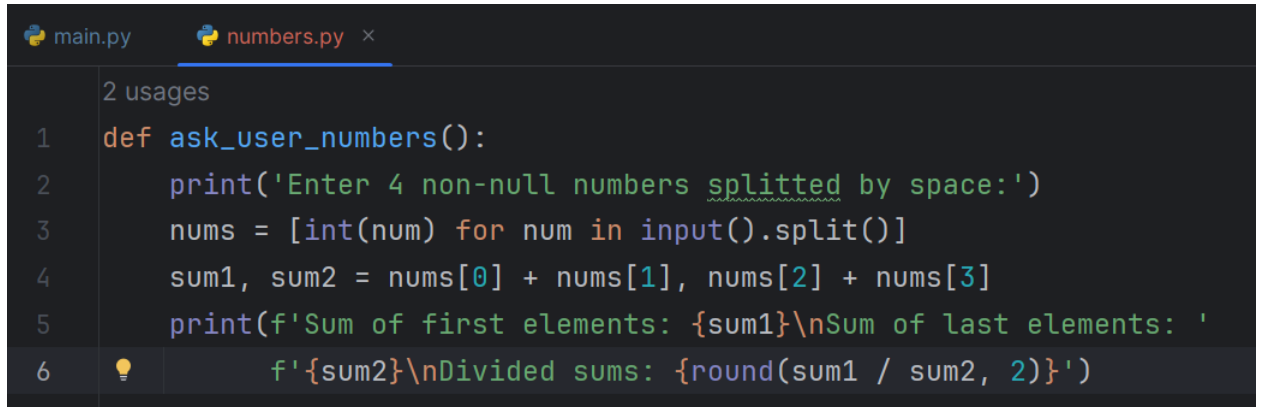
```
Project - fundamentals_of_software_engineering_lab2_1 - develop
main.py arithmetic.py
1 from user import ask_user
2 from arithmetic import solve_math_problem
3
4 if __name__ == '__main__':
5     # ask_user()
6     solve_math_problem()
7
8 if __name__ == '__main__':
9
Run - main
Task: 4 * 100 - 54
Enter your answer: 12323
You're answer: 12323
Right answer: 346
You're answer is wrong!
Process finished with exit code 0
```

Рисунок 25 – Запуск программы (1)

```
Project - fundamentals_of_software_engineering_lab2_1 - develop
main.py arithmetic.py
1 from user import ask_user
2 from arithmetic import solve_math_problem
3
4 if __name__ == '__main__':
5     # ask_user()
6     solve_math_problem()
7
8 if __name__ == '__main__':
9
Run - main
Task: 4 * 100 - 54
Enter your answer: 346
You're answer: 346
Right answer: 346
You're right!
Process finished with exit code 0
```

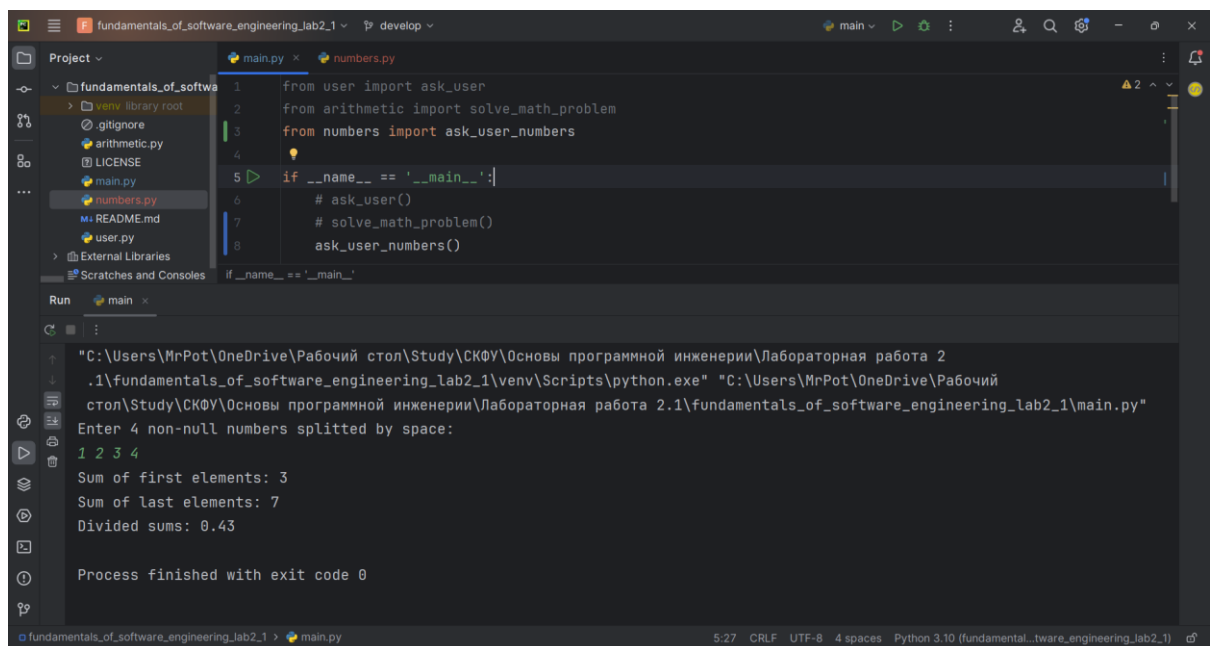
Рисунок 26 – Запуск программы (2)

8. 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:



```
2 usages
1 def ask_user_numbers():
2     print('Enter 4 non-null numbers splitted by space:')
3     nums = [int(num) for num in input().split()]
4     sum1, sum2 = nums[0] + nums[1], nums[2] + nums[3]
5     print(f'Sum of first elements: {sum1}\nSum of last elements: '
6         f'{sum2}\nDivided sums: {round(sum1 / sum2, 2)}')
```

Рисунок 27 – Функция, запрашивающая у пользователя числа и выводящая результат деления первой и последней сумм



```
1 from user import ask_user
2 from arithmetic import solve_math_problem
3 from numbers import ask_user_numbers
4
5 if __name__ == '__main__':
6     # ask_user()
7     # solve_math_problem()
8     ask_user_numbers()
```

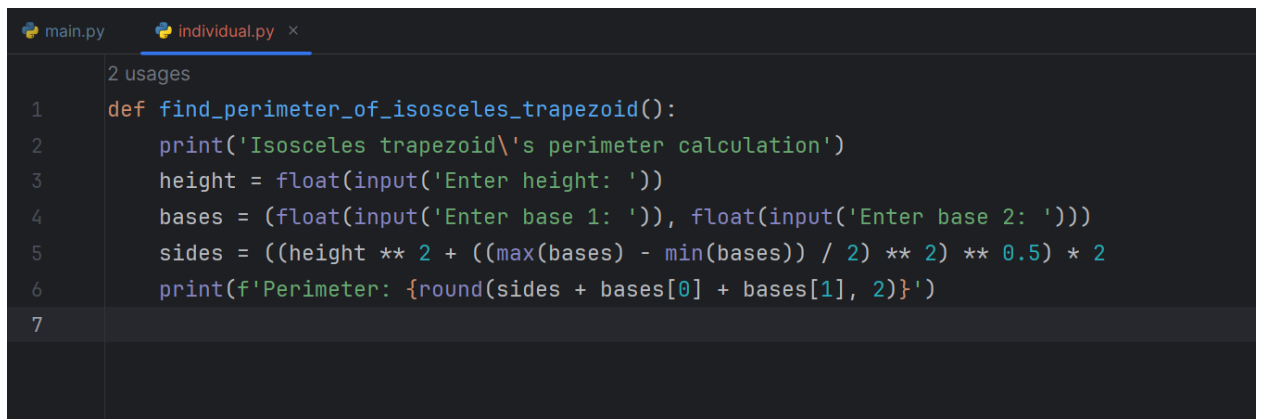
Run main

```
"C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2
.1\fundamentals_of_software_engineering_lab2_1\venv\Scripts\python.exe" "C:\Users\MrPot\OneDrive\Рабочий
стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\main.py"
Enter 4 non-null numbers splitted by space:
1 2 3 4
Sum of first elements: 3
Sum of last elements: 7
Divided sums: 0.43

Process finished with exit code 0
```

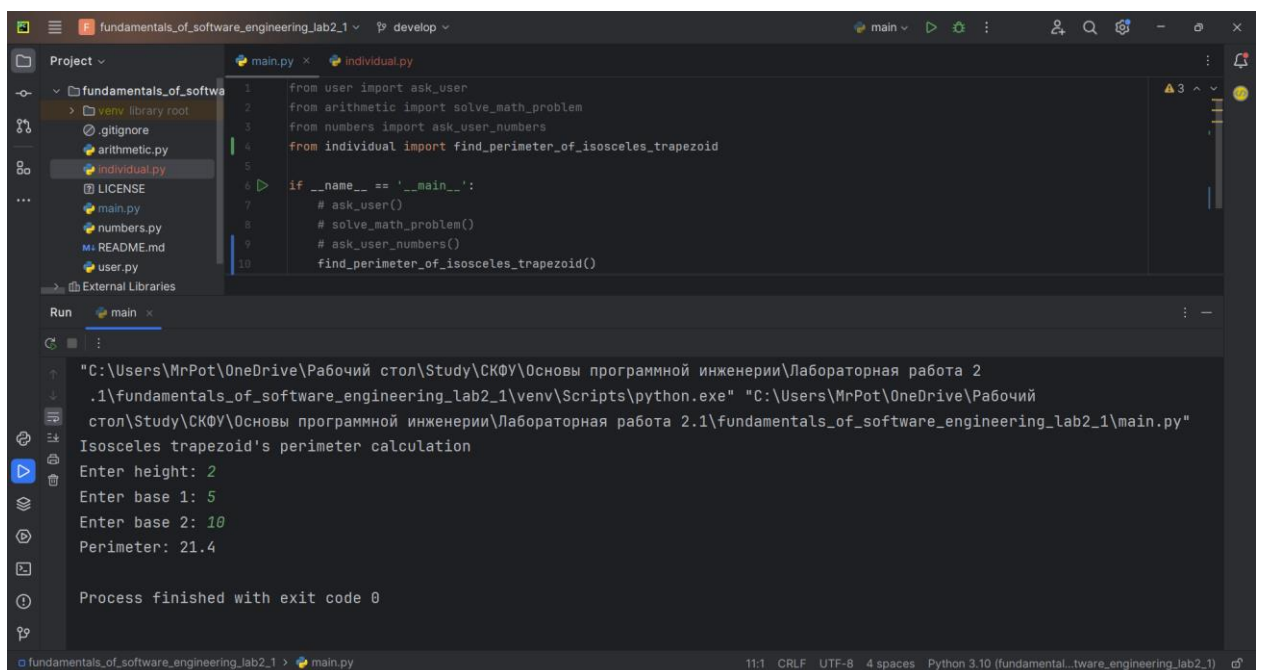
Рисунок 28 – Запуск программы

9. Напишите программу (файл individual.py) для решения индивидуального задания. Даны основания и высота равнобедренной трапеции. Найти периметр трапеции:



```
main.py individual.py x
2 usages
1 def find_perimeter_of_isosceles_trapezoid():
2     print('Isosceles trapezoid\'s perimeter calculation')
3     height = float(input('Enter height: '))
4     bases = (float(input('Enter base 1: ')), float(input('Enter base 2: ')))
5     sides = ((height ** 2 + ((max(bases) - min(bases)) / 2) ** 2) ** 0.5) * 2
6     print(f'Perimeter: {round(sides + bases[0] + bases[1], 2)}')
7
```

Рисунок 29 – Функция нахождения периметра равнобедренной трапеции



```
fundamentals_of_software_engineering_lab2_1 develop
Project
fundamentals_of_software_engineering_lab2_1
venv library root
.gitignore
arithmetic.py
individual.py
LICENSE
main.py
numbers.py
README.md
user.py
External Libraries
main.py individual.py
1 from user import ask_user
2 from arithmetic import solve_math_problem
3 from numbers import ask_user_numbers
4 from individual import find_perimeter_of_isosceles_trapezoid
5
6 if __name__ == '__main__':
7     # ask_user()
8     # solve_math_problem()
9     # ask_user_numbers()
10    find_perimeter_of_isosceles_trapezoid()
Run main
"C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\venv\Scripts\python.exe" "C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\main.py"
Isosceles trapezoid's perimeter calculation
Enter height: 2
Enter base 1: 5
Enter base 2: 10
Perimeter: 21.4
Process finished with exit code 0
fundamentals_of_software_engineering_lab2_1 main.py 11:1 CRLF UTF-8 4 spaces Python 3.10 (fundamental_tware_engineering_lab2_1)
```

Рисунок 30 – Запуск программы

10. Задача повышенной сложности. Часовая стрелка образует угол y с лучом, проходящим через центр и через точку, соответствующую 12 часам на циферблате $0 < y \leq 2\pi$. Определить значение угла для минутной стрелки, а также количество полных часов и полных минут:

```

2 usages
def get_clock_data_by_hour_hand_degree():
    hour_degree = float(input('Enter hour hand degree (from 0 to 360): '))
    hours, minutes = hour_degree // 30, round(hour_degree % 30 * 2, 2)
    print(f'Hours: {hours}, minutes: {minutes}')
    print(f'Minute hand degree: {round((minutes // 5 * 30) + (minutes % 5 * 6), 2)}')

```

Рисунок 31 – Функция нахождения полных часов, минут и градусов поворота минутной стрелки

```

1 from individual import get_clock_data_by_hour_hand_degree
2
3 if __name__ == '__main__':
4     get_clock_data_by_hour_hand_degree()
5
Run main
"C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2
.1\fundamentals_of_software_engineering_lab2_1\venv\Scripts\python.exe" "C:\Users\MrPot\OneDrive\Рабочий
стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\main.py"
Enter hour hand degree (from 0 to 360): 46
Hours: 1.0, minutes: 32.0
Minute hand degree: 192.0

Process finished with exit code 0

```

Рисунок 32 – Запуск программы (1)

```

1 from individual import get_clock_data_by_hour_hand_degree
2
3 if __name__ == '__main__':
4     get_clock_data_by_hour_hand_degree()
5
Run main
"C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2
.1\fundamentals_of_software_engineering_lab2_1\venv\Scripts\python.exe" "C:\Users\MrPot\OneDrive\Рабочий
стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1\main.py"
Enter hour hand degree (from 0 to 360): 178.4
Hours: 5.0, minutes: 56.8
Minute hand degree: 340.8

Process finished with exit code 0

```

Рисунок 33 – Запуск программы (2)

11. Коммит всех созданных файлов:

```
(venv) PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1> git log --oneline
b6bc7bd (HEAD -> develop) individual.py added
bf89ac6 numbers.py added
7dece19 arithmetic.py added
86cb822 user.py added
4a5d788 Setup Project
77a3536 (origin/main, origin/HEAD, main) Initial commit
(venv) PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1>
```

Рисунок 34 – Коммиты ветки develop во время выполнения лабораторной работы

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

```
software_engineering_lab2_1> git merge develop
Updating 77a3536..b6bc7bd
Fast-forward
 .idea/.gitignore | 3 +++
 .idea/fundamentals_of_software_engineering_lab2_1.iml | 10 ++++++++
 .idea/inspectionProfiles/Project_Default.xml | 12 ++++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/misc.xml | 7 +++++
 .idea/modules.xml | 8 +++++
 .idea/vcs.xml | 6 +++++
 arithmetic.py | 9 ++++++
 individual.py | 13 ++++++++
```

Рисунок 35 – Слияние ветки develop в ветку main

```
software_engineering_lab2_1> git push origin main
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 12 threads
Compressing objects: 100% (29/29), done.
Writing objects: 100% (31/31), 4.56 KiB | 4.56 MiB/s, done.
Total 31 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/PoTtaTo/fundamentals_of_software_engineering_lab2_1
 77a3536..aeeb49b main -> main
(venv) PS C:\Users\MrPot\OneDrive\Рабочий стол\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.1\fundamentals_of_software_engineering_lab2_1>
```

Рисунок 36 – 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 ручную, 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 во время 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.

```
>>> pupil = "Ben"
>>> old = 16
>>> grade = 9.2
>>> print("It's %s, %d. Level: %f" % (pupil, old, grade))
It's Ben, 16. Level: 9.200000
```

Рисунок 37 – Пример форматированием Си-стилем

```
>>> print("This is a {0}. It's {1}.".format("ball", "red"))
This is a ball. It's red.
>>> print("This is a {0}. It's {1}.".format("cat", "white"))
This is a cat. It's white.
>>> print("This is a {0}. It's {1} {2}.".format(1, "a", "number"))
This is a 1. It's a number.
```

Рисунок 38 – Пример форматирования функцией format()

```
# Python3 program introducing f-string
val = 'Geeks'
print(f"{val}for{val} is a portal for {val}.")

name = 'Tushar'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

Рисунок 39 – Пример форматирования с помощью f-strings

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

int(input()) или float(input()).