

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 №4**  
**дисциплины «Основы программной инженерии»**

Выполнила:  
Панюкова Ксения Юрьевна  
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:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

# Ход работы

## 1. Я изучила теоретический материал работы

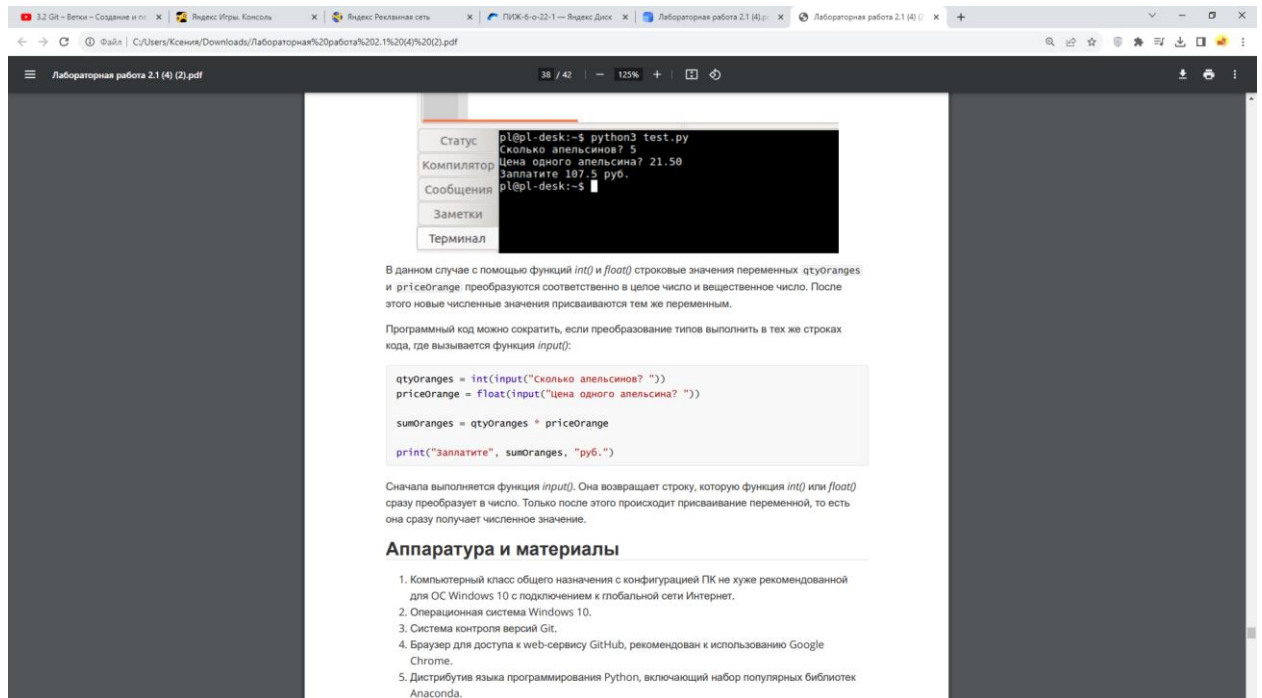


Рисунок 1.1 – Изучение материала для лабораторной работы


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

## 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 \*

 MrFinichек

Repository name \*

AgainPython

✓ AgainPython is available.

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

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

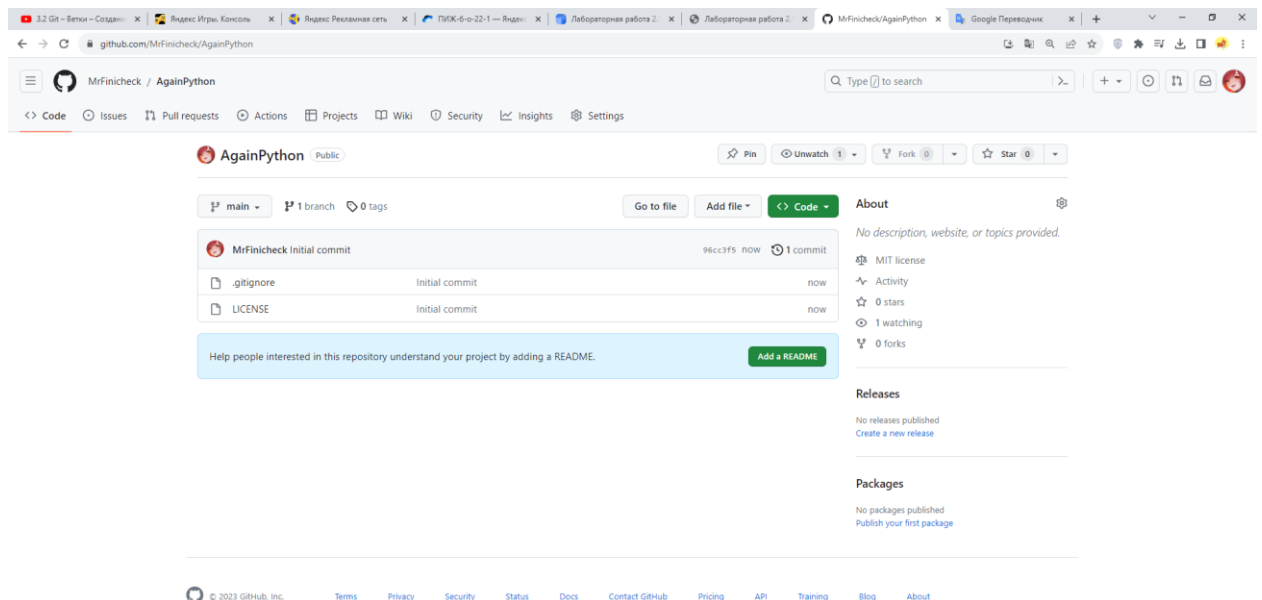
License: **MIT License**

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 2.1 – Настройка репозитория



## Рисунок 2.2 – Готовый репозиторий

### 3. Выполняю клонирование созданного репозитория

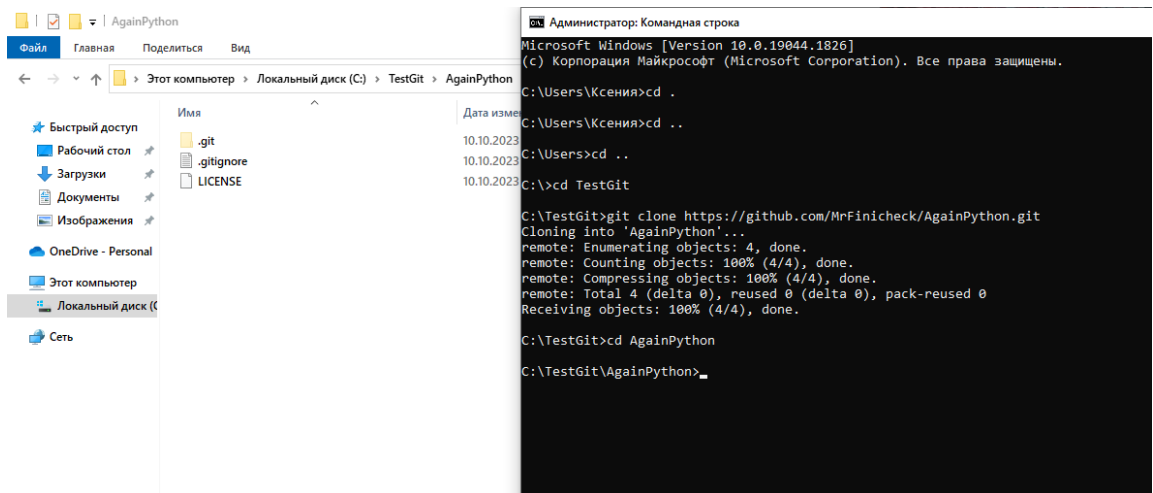


Рисунок 3.1 – Клонирование репозитория на локальный диск

### 4. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm

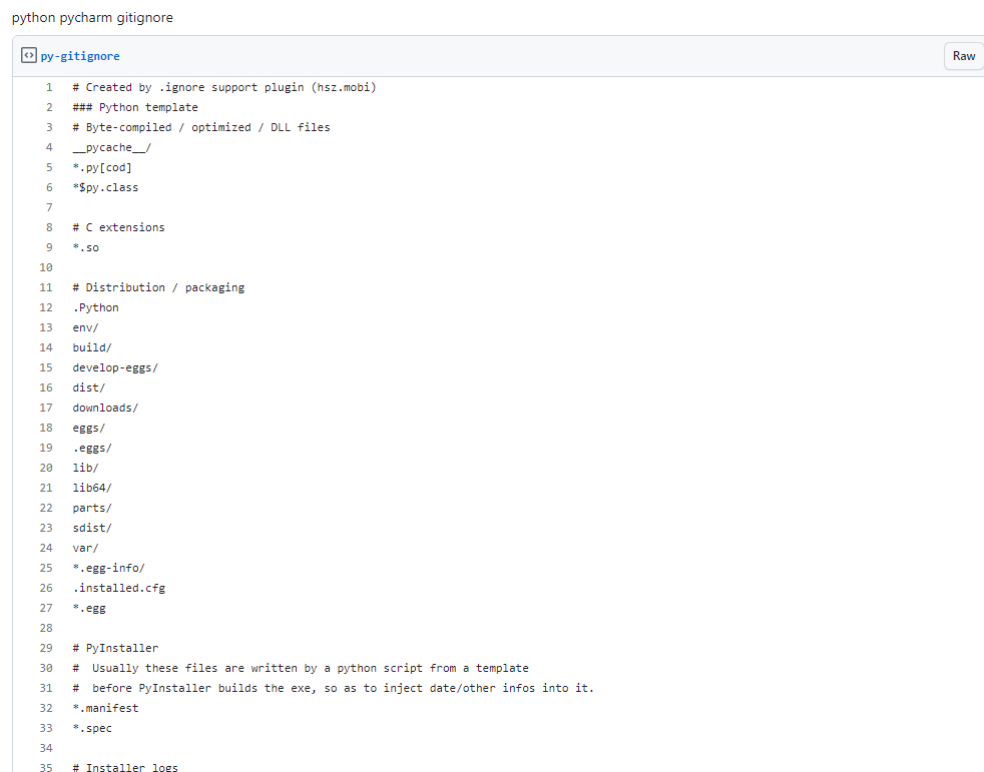


Рисунок 4.1 – .gitignore для IDE PyCharm

### 5. Организовала свой репозиторий в соответствии с моделью ветвления git-flow

```
C:\TestGit\AgainPython>git branch develop

C:\TestGit\AgainPython>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinicheck/AgainPython/pull/new/develop
remote:
To https://github.com/MrFinicheck/AgainPython.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\TestGit\AgainPython>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
```

Рисунок 5.1 – Создание ветки develop от ветки main

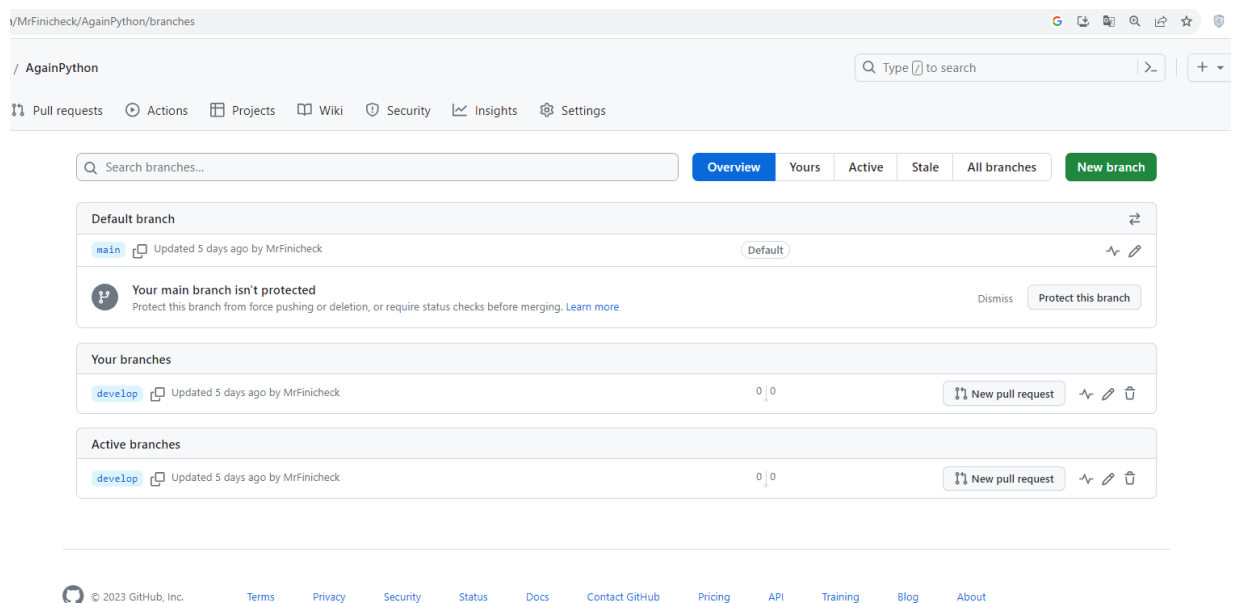


Рисунок 5.2 – Ветка develop на GitHub

## 6. Создала проект PyCharm в папке репозитория

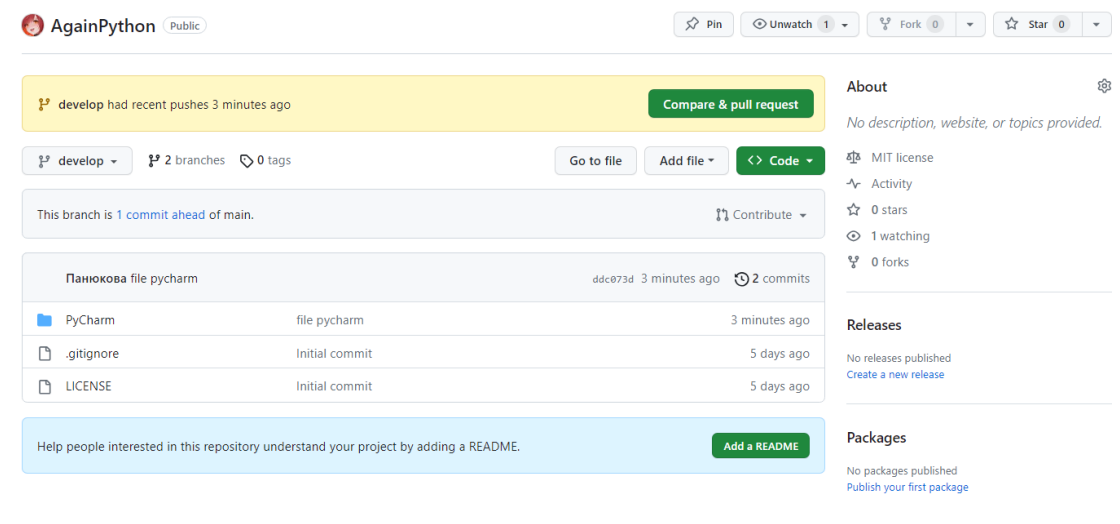


Рисунок 6.1 – Репозиторий с проектом PyCharm

## 7. Решила следующие задачи с помощью языка программирования Python3 и IDE PyCharm

8. 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:

```
"This is `имя`"
"It is `возраст`"
"(S)he live in `место_жительства`"
```

Вместо `имя`, `возраст`, `место_жительства` должны быть данные, введенные пользователем.

Примечание: можно писать фразы на русском языке, но если вы планируете стать профессиональным программистом, привыкайте к английскому.

9. 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.

10. 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.

11. Напишите программу (файл `individual.py`) для решения индивидуального задания. Вариант индивидуального задания уточните у преподавателя.

Рисунок 7.1 – Задачи для решения

8. 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:

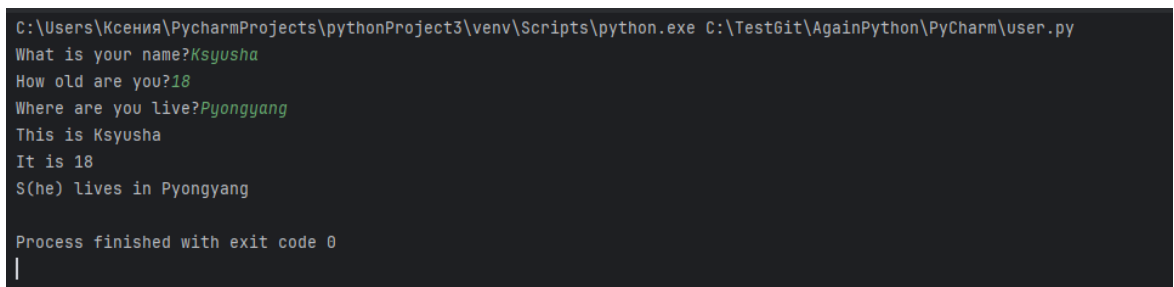
- "This is `имя`"
- "It is возраст"
- "(S)he lives in `место\_жительства`"

Вместо имя, возраст, место\_жительства должны быть данные, введенные пользователем.



```
user.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      name = input("What is your name?")
5      age = input("How old are you?")
6      location = input("Where are you live?")
7      print("This is %s\nIt is %s\nS(he) lives in %s" %
8          (name, age, location))
9
```

Рисунок 8.1 – Код программы user.py в IDE PyCharm

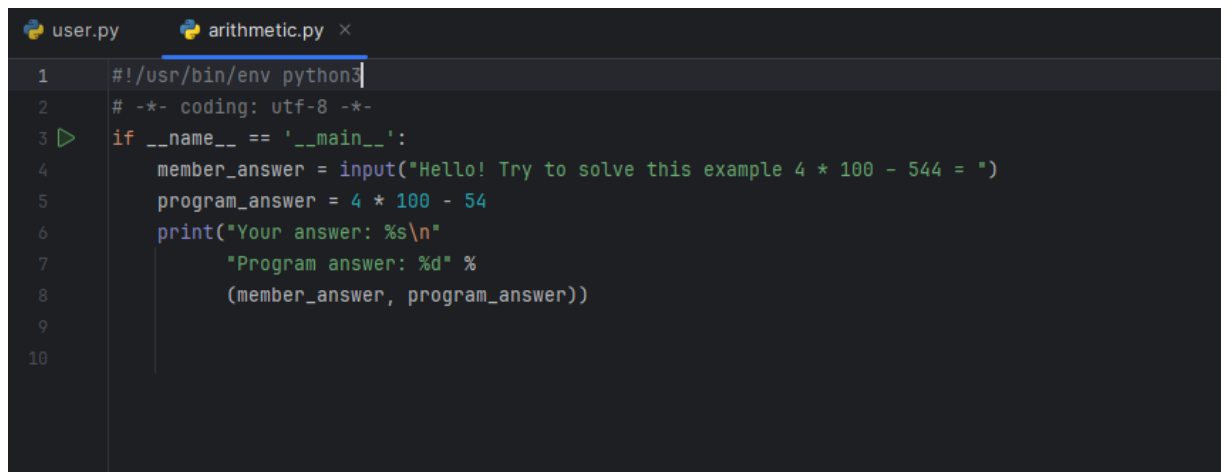


```
C:\Users\Ксения\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\TestGit\AgainPython\PyCharm\user.py
What is your name?Ksyusha
How old are you?18
Where are you live?Pyongyang
This is Ksyusha
It is 18
S(he) lives in Pyongyang

Process finished with exit code 0
|
```

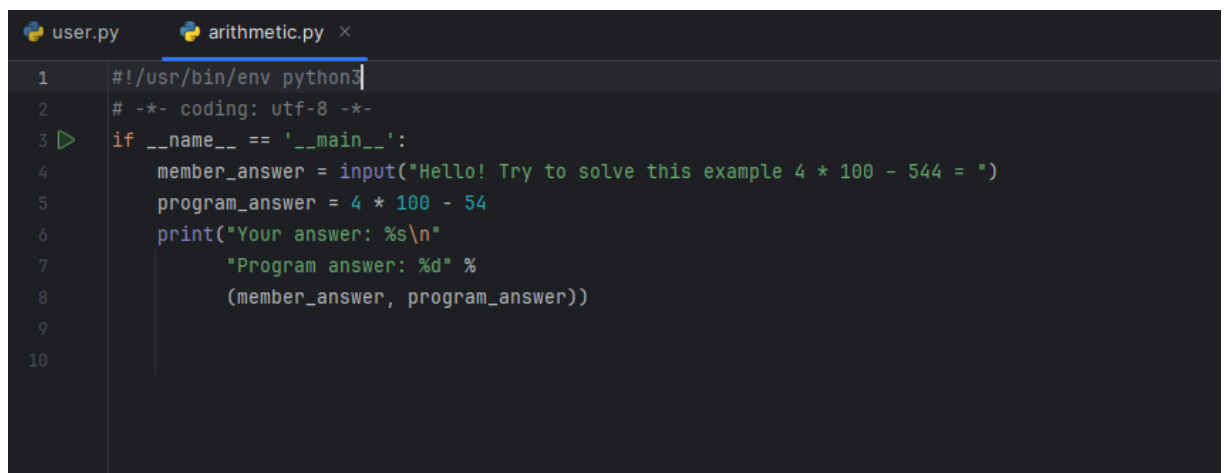
Рисунок 8.2 – Результат выполнения программы user.py

9. 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



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      member_answer = input("Hello! Try to solve this example 4 * 100 - 544 = ")
5      program_answer = 4 * 100 - 54
6      print("Your answer: %s\n"
7            "Program answer: %d" %
8            (member_answer, program_answer))
9
10
```

Рисунок 9.1 – Код программы arithmetic.py в IDE PyCharm

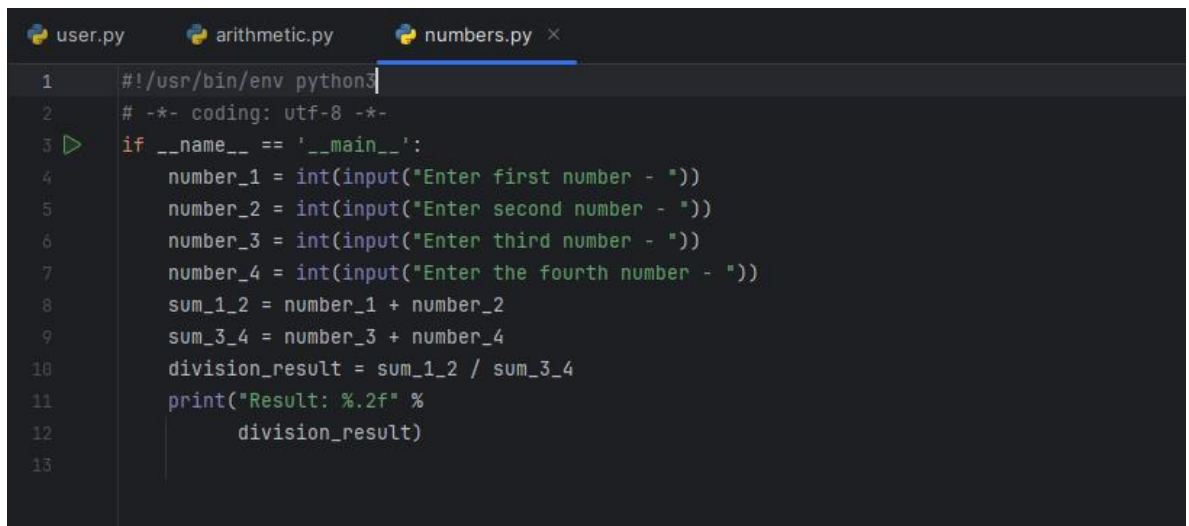


```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      member_answer = input("Hello! Try to solve this example 4 * 100 - 544 = ")
5      program_answer = 4 * 100 - 54
6      print("Your answer: %s\n"
7            "Program answer: %d" %
8            (member_answer, program_answer))
9
10
```

Рисунок 9.2 – Результат выполнения программы arithmetic.py

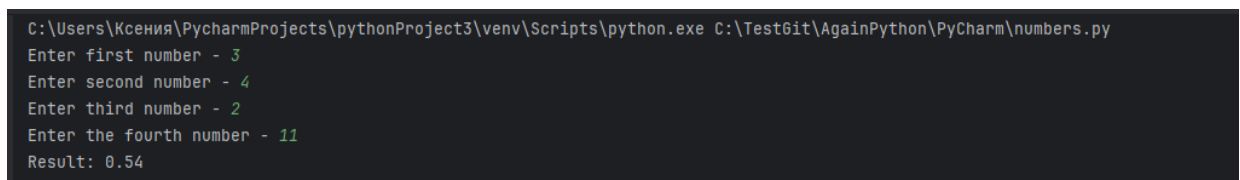
10. Запросите у пользователя четыре числа (файл numbers.py). Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.





```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      number_1 = int(input("Enter first number - "))
5      number_2 = int(input("Enter second number - "))
6      number_3 = int(input("Enter third number - "))
7      number_4 = int(input("Enter the fourth number - "))
8      sum_1_2 = number_1 + number_2
9      sum_3_4 = number_3 + number_4
10     division_result = sum_1_2 / sum_3_4
11     print("Result: %.2f" %
12           division_result)
13
```

Рисунок 10.1 – Код программы numbers.py в IDE PyCharm



```
C:\Users\Ксения\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\Test6it\AgainPython\PyCharm\numbers.py
Enter first number - 3
Enter second number - 4
Enter third number - 2
Enter the fourth number - 11
Result: 0.54
```

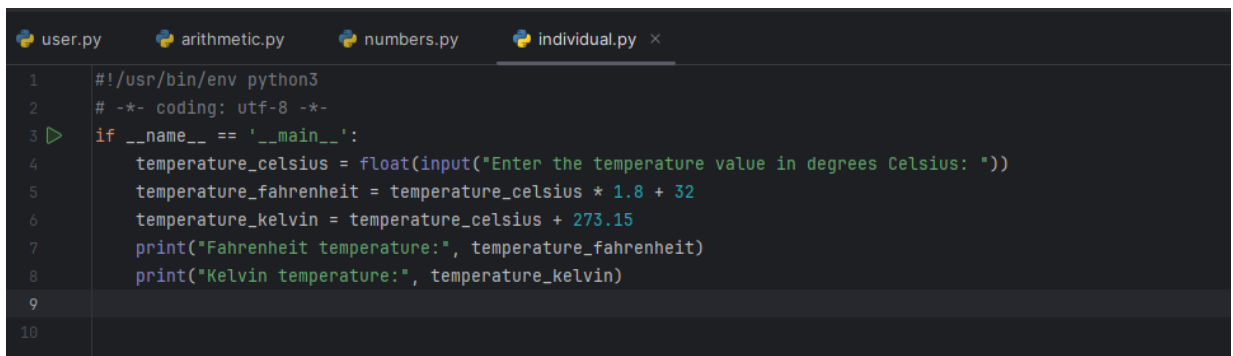
Рисунок 10.2 – Результат выполнения программы numbers.py

11. Напишите программу (файл individual.py) для решения индивидуального задания (вариант 16).

Известно значение температуры по шкале Цельсия. Найти соответствующее значение температуры по шкале:

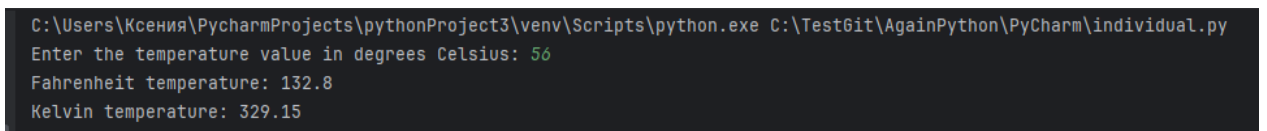
- Фаренгейта;
- Кельвина.

Для пересчета по шкале Фаренгейта необходимо исходное значение температуры умножить на 1,8 и к результату прибавить 32, а по шкале Кельвина абсолютное значение нуля соответствует  $-273,15$  градуса по шкале Цельсия.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == '__main__':
4      temperature_celsius = float(input("Enter the temperature value in degrees Celsius: "))
5      temperature_fahrenheit = temperature_celsius * 1.8 + 32
6      temperature_kelvin = temperature_celsius + 273.15
7      print("Fahrenheit temperature:", temperature_fahrenheit)
8      print("Kelvin temperature:", temperature_kelvin)
9
10
```

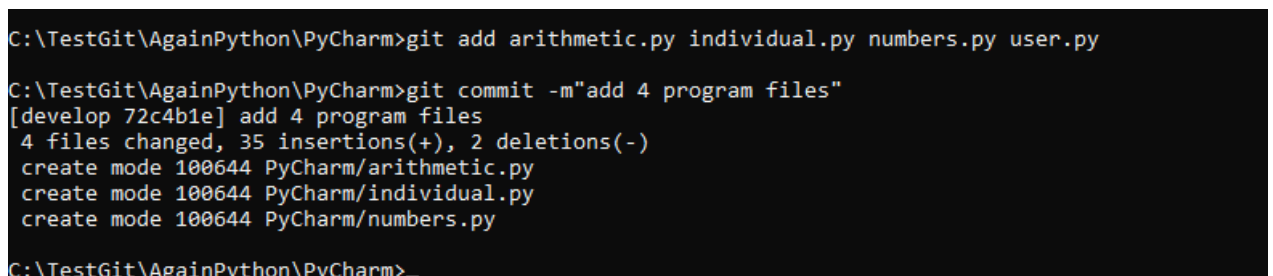
Рисунок 11.1 – Код программы individual.py в IDE PyCharm



```
C:\Users\Ксения\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\TestGit\AgainPython\PyCharm\individual.py
Enter the temperature value in degrees Celsius: 56
Fahrenheit temperature: 132.8
Kelvin temperature: 329.15
```

Рисунок 11.2 – Результат выполнения программы individual.py

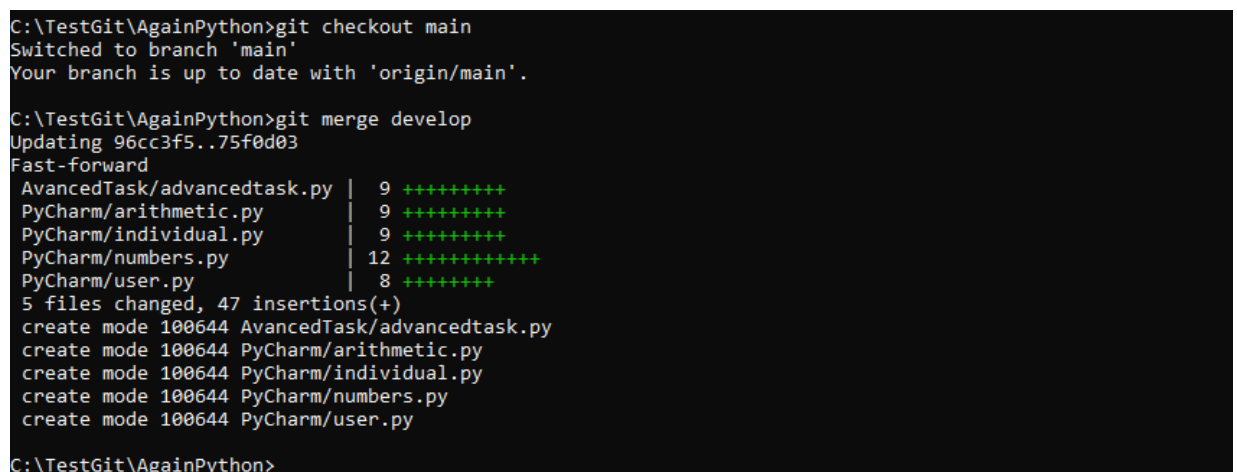
12. Выполнила коммит файлов user.py, arithmetic.py, numbers.py и individual.py в репозиторий git в ветку для разработки.



```
C:\TestGit\AgainPython\PyCharm>git add arithmetic.py individual.py numbers.py user.py
C:\TestGit\AgainPython\PyCharm>git commit -m"add 4 program files"
[develop 72c4b1e] add 4 program files
4 files changed, 35 insertions(+), 2 deletions(-)
create mode 100644 PyCharm/arithmetic.py
create mode 100644 PyCharm/individual.py
create mode 100644 PyCharm/numbers.py
C:\TestGit\AgainPython\PyCharm>
```

Рисунок 12.1 – Коммит файлов в репозиторий git

13. Выполнила слияние ветки для разработки с веткой main



```
C:\TestGit\AgainPython>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\TestGit\AgainPython>git merge develop
Updating 96cc3f5..75f0d03
Fast-forward
 AvancedTask/advancedtask.py | 9 ++++++++
 PyCharm/arithmetic.py       | 9 ++++++++
 PyCharm/individual.py       | 9 ++++++++
 PyCharm/numbers.py         | 12 ++++++++
 PyCharm/user.py            | 8 ++++++
 5 files changed, 47 insertions(+)
 create mode 100644 AvancedTask/advancedtask.py
 create mode 100644 PyCharm/arithmetic.py
 create mode 100644 PyCharm/individual.py
 create mode 100644 PyCharm/numbers.py
 create mode 100644 PyCharm/user.py
C:\TestGit\AgainPython>
```

## Рисунок 13.1 – Слияние ветки main с веткой develop

### 14. Отправила сделанные изменения на сервер GitHub

```
C:\TestGit\AgainPython>git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 4 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.76 KiB | 1.76 MiB/s, done.
Total 11 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MrFinichcheck/AgainPython.git
96cc3f5..75f0d03  main -> main
C:\TestGit\AgainPython>
```

### Рисунок 14.1 – Отправка изменений на сервер GitHub

## Контрольные вопросы

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

#### Установка Python в Windows

Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла (с расширением exe), либо в виде архивного файла (с расширением zip). Если вы используете Windows 7, не забудьте установить Service Pack 1.

Порядок установки.

1. Запустите скачанный установочный файл.
2. Выберите способ установки
3. Отметьте необходимые опции установки (доступно при выборе Customize installation)
4. Выберите место установки (доступно при выборе Customize installation)
5. После успешной установки вас ждет сообщение об успешной установке

#### Установка Python в Linux

Чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить, набрав в терминале python или python 3. В первом случае вы запустите Python 2 во втором – Python 3. Если у вас, при попытке запустить

Python, выдается сообщение о том, что он не установлен, или установлен, но не тот, что вы хотите, то у вас есть два пути: а) собрать Python из исходников; б) взять из репозитория. Для установки из репозитория в Ubuntu воспользуйтесь командой `sudo apt-get install python3`.

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

Пакет 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?

Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. В Windows это можно сделать, выбрав следующий пункт главного меню системы Пуск → Anaconda3 (64-bit) → Anaconda Prompt. В появившейся командной строке необходимо ввести `jupyter notebook` в результате чего отобразится процесс загрузки веб-среды Jupyter Notebook после чего запустится веб-сервер и среда разработки в браузере.

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

Запустите PyCharm и выберите Create New Project в появившемся окне. Укажите путь до проекта Python и интерпретатор, который будет использоваться для запуска и отладки.

5. Как осуществить запуск программы с помощью IDE PyCharm?

После того, как исходный код написан, чтобы первый раз запустить программу, проще всего нажать `Ctrl+Shift+F10`.

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

В интерактивный режим можно войти, набрав в командной строке `python` или `python3`. В результате Python запустится в интерактивном режиме и будет ожидать ввод команд пользователя.

Чтобы выполнить запуск в пакетном режиме, надо ввести в командной строке имя интерпретатора, плюс имя файла.

7. Почему язык программирования Python называется языком динамической типизации?

Если достаточно формально подходить к вопросу о типизации языка Python, то можно сказать, что он относится к языкам с неявной сильной динамической типизацией. Неявная типизация означает, что при объявлении переменной вам не нужно указывать её тип, при явной – это делать необходимо. Также языки бывают с динамической и статической типизацией. В первом случае тип переменной определяется непосредственно при выполнении программы, во втором – на этапе компиляции. Как уже было сказано Python – это динамически типизированный язык.

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);
- Списки (Sequence Type);
- Строки (Text Sequence Type);
- 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?

Объект – 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. 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 (например, 5), 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?

Проверить является ли идентификатор ключевым словом можно с помощью команды `keyword.iskeyword("название_переменной")`.

#### 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

В Python существуют изменяемые и неизменяемые типы.

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 (frozen set).

К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

Неизменяемость типа данных означает, что созданный объект больше не изменяется. Если тип данных изменяемый, то можно менять значение объекта.

#### 13. Чем отличаются операции деления и целочисленного деления?

Целочисленное деление (`//`) отличается от обычной операции деления тем, что возвращает целую часть частного, а дробная часть отбрасывается.

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. У комплексного числа можно извлечь действительную (`x.real`) и мнимую части (`x.imag`). Для получения комплексно сопряженного числа необходимо использовать метод `conjugate()`.

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. Основные функции `math`:

- `math.ceil(x)` возвращает ближайшее целое число большее, чем  $x$
- `math.fabs(x)` возвращает абсолютное значение числа
- `math.factorial(x)` вычисляет факториал  $x$
- `math.floor(x)` возвращает ближайшее целое число меньшее, чем  $x$
- `math.exp(x)` вычисляет  $e^{**}x$
- `math.log2(x)` логарифм по основанию 2
- `math.log10(x)` логарифм по основанию 10
- `math.log(x[, base])` по умолчанию вычисляет логарифм по основанию  $e$ , дополнительно можно указать основание логарифма
- `math.pow(x, y)` вычисляет значение  $x$  в степени  $y$
- `math.sqrt(x)` корень квадратный от  $x$
- `math.cos(x)` косинус от  $x$
- `math.sin(x)` синус от  $x$

- `math.tan(x)` тангенс от  $x$
- `math.acos(x)` арккосинус от  $x$
- `math.asin(x)` арксинус от  $x$
- `math.atan(x)` арктангенс от  $x$
- `math.pi` число  $\pi$
- `math.e` число  $e$

Модуль `math` предоставляет доступ к математическим функциям для комплексных чисел. Функции в этом модуле принимают в качестве аргументов целые числа, числа с плавающей запятой или комплексные числа. Они также будут принимать любой объект Python, у которого есть метод `__complex__()` или `__float__()`: данные методы используются для преобразования объекта в комплексное число или число с плавающей запятой соответственно, а затем функция применяется к результату преобразования.

Множество функций подобные, но не идентичные тому, что в модуле `math`. Причина наличия двух модулей в том, что некоторые пользователи не интересуются комплексными числами и, возможно, даже не знают, что они собой представляют. Функции, определённые в `cmath`, всегда возвращают комплексное число, даже если ответ может быть выражен в виде действительного числа (в этом случае комплексное число имеет нулевую мнимую часть). Основные функции `cmath`:

- `cmath.polar(x)` возвращает представление  $x$  в полярных координатах.
- `cmath.rect(r, phi)` возвращает комплексное число  $x$  с полярными координатами  $r$  и  $\phi$
- `cmath.phase(x)` возвращает фазу  $x$  (также известную как `argument`  $x$ ) в виде числа с плавающей запятой
- `cmath.isfinite(x)` возвращает `True`, если и действительная, и мнимая части  $x$  конечны, и `False` в противном случае
- `cmath.isinf(x)` возвращает `True`, если действительная или мнимая часть  $x$  равна бесконечности, и `False` в противном случае



- `cmath.isnan(x)` возвращает True, если действительная или мнимая часть `x` является NaN, и False в противном случае
- `cmath.infj` комплексное число с нулевой действительной частью и положительной бесконечностью мнимой части
- `cmath.nanj` комплексное число с нулевой действительной частью и NaN мнимой частью.`math.sqrt(x)` корень квадратный от `x`
- `math.nan` Значение с плавающей запятой «не число» (NaN)

16. Каково назначение именных параметров `sep` и `end` в функции `print()`?

Через параметр `sep` можно указать отличный от пробела разделитель строк. Параметр `end` позволяет указывать, что делать, после вывода строки.

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.

В строке в фигурных скобках указаны номера данных, которые будут подставлены. Далее к строке применяется метод `format()`. В его скобках указываются сами данные (можно использовать переменные). На нулевое место подставится первый аргумент метода `format()`, на место с номером 1 – второй и т. д. Существует также Старый стиль его называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке C. Вместо трех комбинаций символов `%s` , `%d` , `%f` подставляются значения переменных. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число. Если бы требовалось подставить три строки, то во всех случаях использовалось бы сочетание `%s`.

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

За ввод в программу данных с клавиатуры в Python отвечает функция `input()`. Когда вызывается эта функция, программа останавливает свое выполнение и ждет, когда пользователь введет текст. Функция `input()` передает

введенные данные в программу. Их можно присвоить переменной. Чтобы получить целочисленное и вещественное значение нужно воспользоваться функцией преобразования типов. В данном случае с помощью функций `int()` и `float()`.