

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:
Юрьев Илья Евгеньевич
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 _____

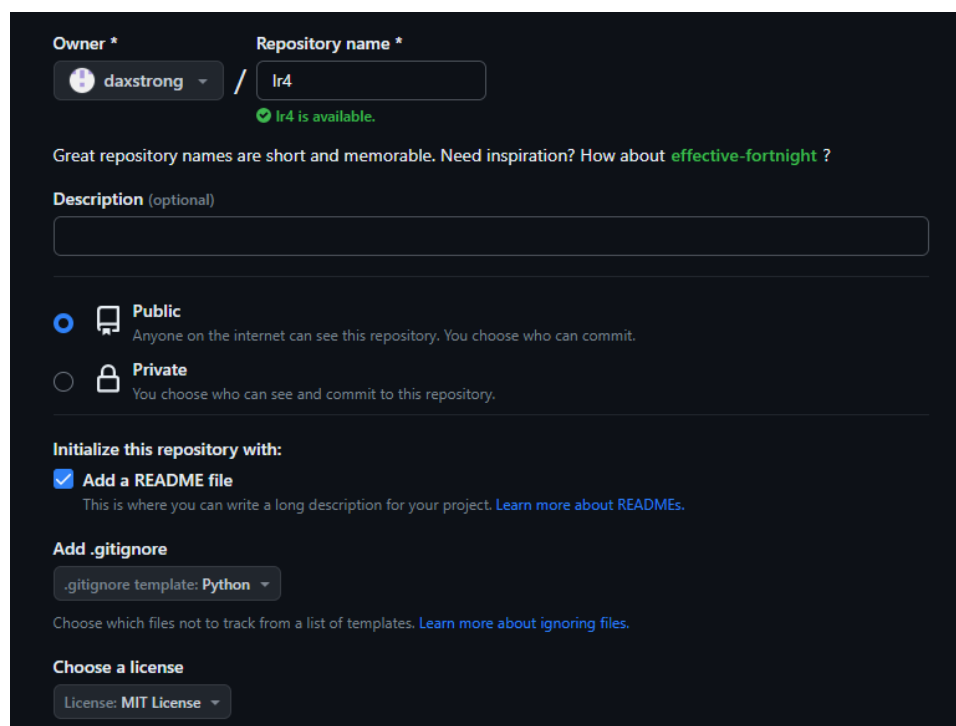
Ставрополь, 2023 г.

Тема: Основы языка Python.

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

Ход выполнения работы:

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



The screenshot shows the GitHub repository creation interface. The 'Owner' is 'daxstrong' and the 'Repository name' is 'lr4', with a green checkmark indicating 'lr4 is available'. The 'Description' field is empty. The 'Public' option is selected under 'Initialize this repository with:'. The 'Add a README file' checkbox is checked. The '.gitignore' template is set to 'Python'. The 'License' is set to 'MIT License'.

Рисунок 1 – Создание репозитория с заданными настройками

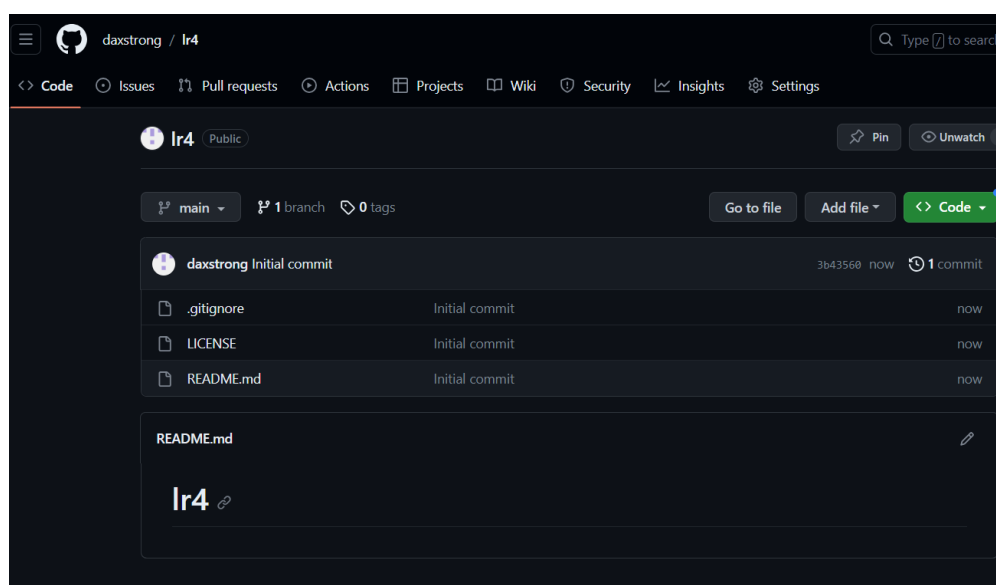


Рисунок 2 – Созданный репозиторий

```
MINGW64/c/Users/ilyay/OneDrive/Рабочий стол/Основы программной инженерии
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии
$ git clone https://github.com/daxstrong/lr4.git
Cloning into 'lr4'...
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.
```

Рисунок 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr4 (main)
$ git branch develop

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr4 (main)
$ git checkout develop
Switched to branch 'develop'
```

Рисунок 4 – Организация репозитория в соответствии с моделью ветвления git-flow

```
.gitignore
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[od]
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/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into
32 *.manifest
33 *.spec
```

Рисунок 5 – Файл .gitignore, созданный в GitHub

2. Создадим проект PyCharm в репозитории:

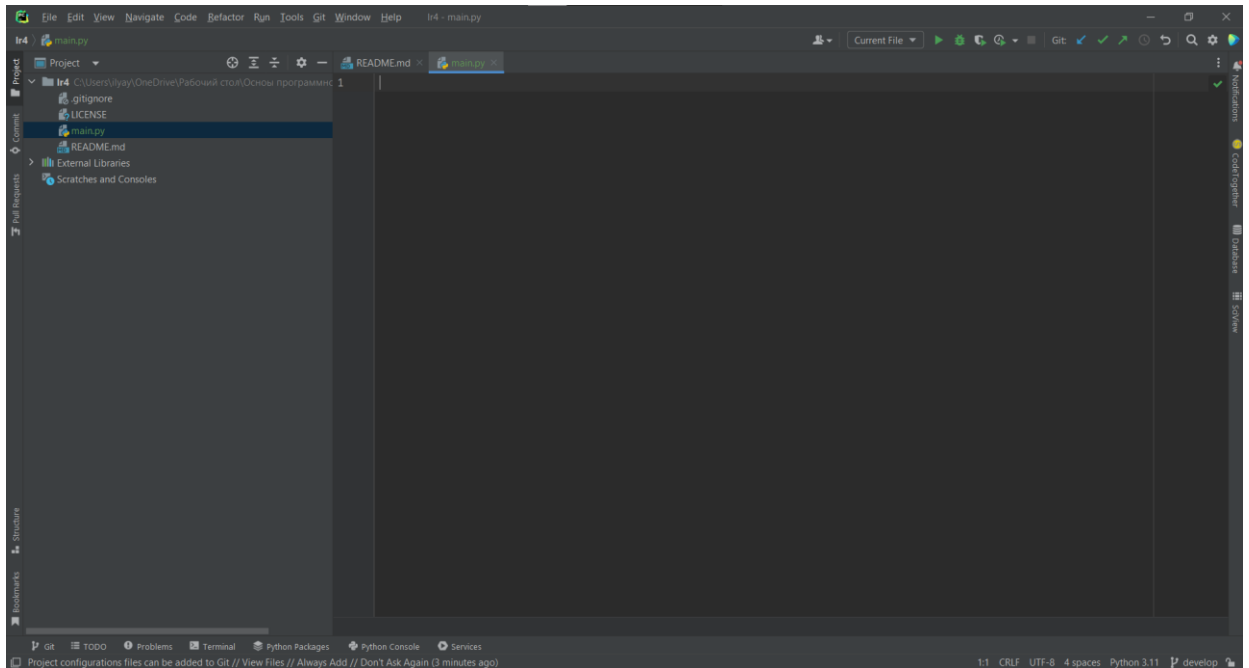


Рисунок 6 – Проект PyCharm

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

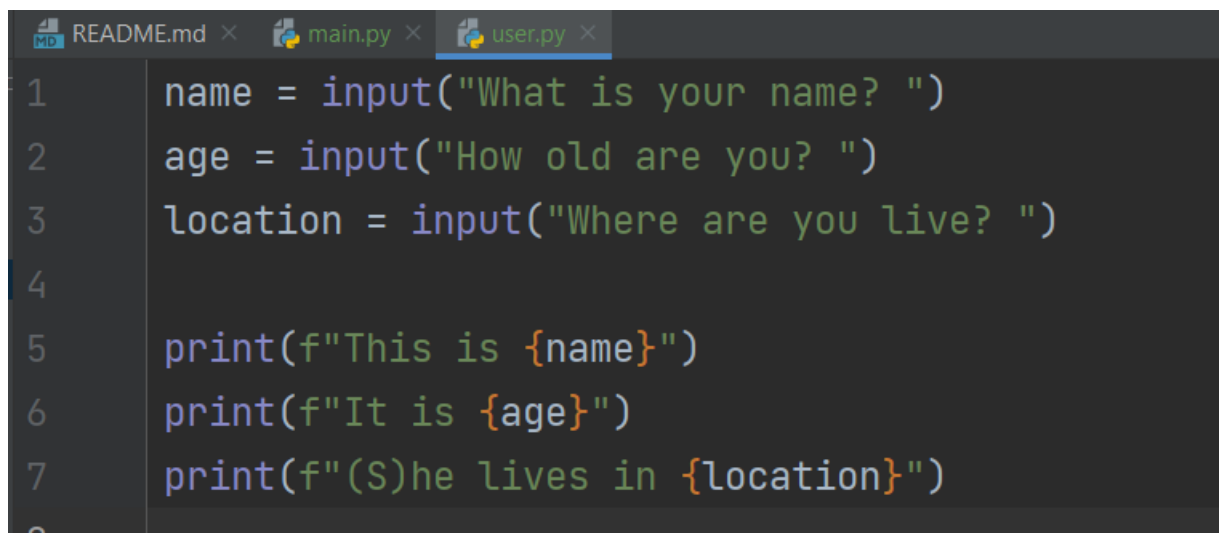
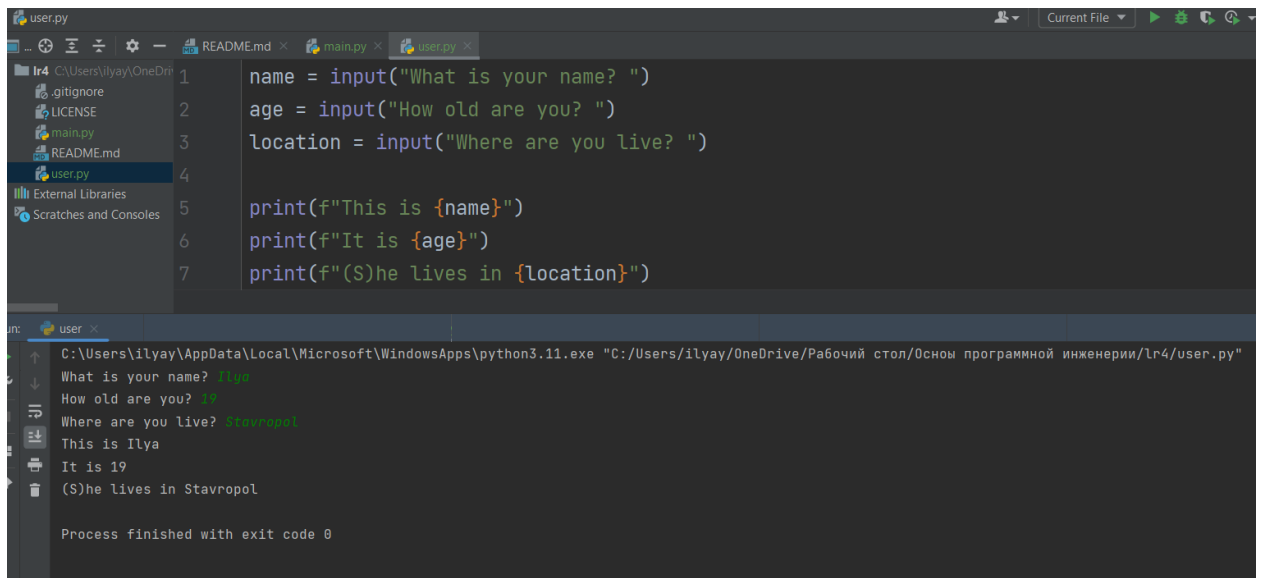


Рисунок 7 – Программа вывода данных о пользователе



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'lr4' with files like '.gitignore', 'LICENSE', 'main.py', 'README.md', and 'user.py'. The code editor displays a Python script in 'user.py' with the following code:

```
1 name = input("What is your name? ")
2 age = input("How old are you? ")
3 location = input("Where are you live? ")
4
5 print(f"This is {name}")
6 print(f"It is {age}")
7 print(f"(S)he lives in {location}")
```

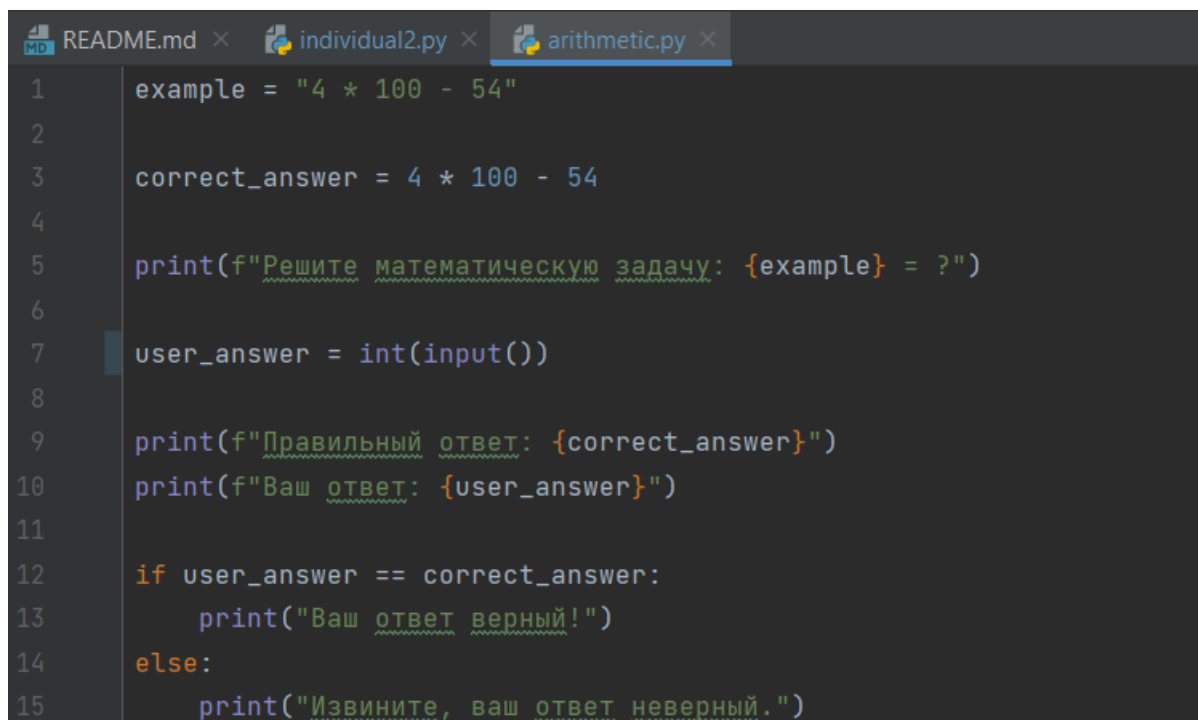
The terminal at the bottom shows the execution of the script using Python 3.11. The user inputs 'Ilya' for the name, '19' for the age, and 'Stavropol' for the location. The output of the script is:

```
What is your name? Ilya
How old are you? 19
Where are you live? Stavropol
This is Ilya
It is 19
(S)he lives in Stavropol

Process finished with exit code 0
```

Рисунок 8 – Запуск программы

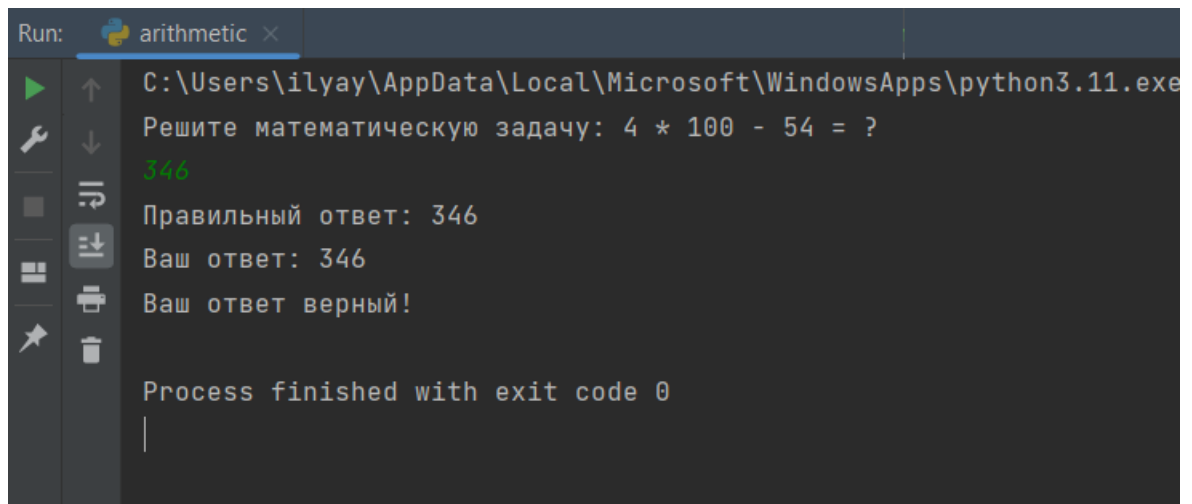
4. Напишите программу (файл arithmetic.py), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран правильный ответ и ответ пользователя:



The screenshot shows a code editor with three tabs: 'README.md', 'individual2.py', and 'arithmetic.py'. The 'arithmetic.py' tab is active, showing the following code:

```
1 example = "4 * 100 - 54"
2
3 correct_answer = 4 * 100 - 54
4
5 print(f"Решите математическую задачу: {example} = ?")
6
7 user_answer = int(input())
8
9 print(f"Правильный ответ: {correct_answer}")
10 print(f"Ваш ответ: {user_answer}")
11
12 if user_answer == correct_answer:
13     print("Ваш ответ верный!")
14 else:
15     print("Извините, ваш ответ неверный.")
```

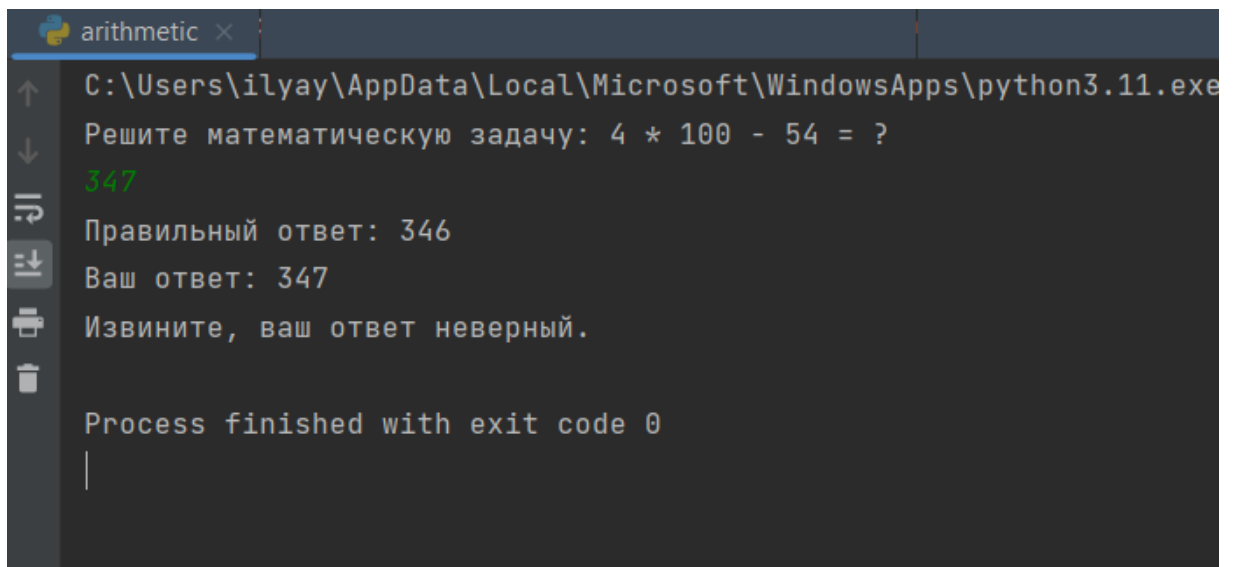
Рисунок 9 – Программа проверки решения математического примера



```
Run: arithmetic x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Решите математическую задачу: 4 * 100 - 54 = ?
346
Правильный ответ: 346
Ваш ответ: 346
Ваш ответ верный!

Process finished with exit code 0
```

Рисунок 10 – Запуск программы (1)

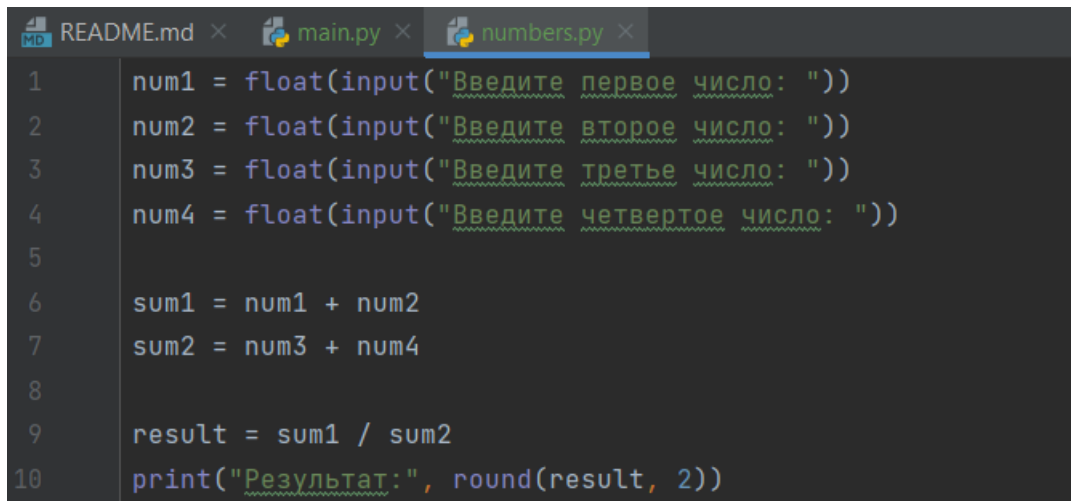


```
arithmetic x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Решите математическую задачу: 4 * 100 - 54 = ?
347
Правильный ответ: 346
Ваш ответ: 347
Извините, ваш ответ неверный.

Process finished with exit code 0
```

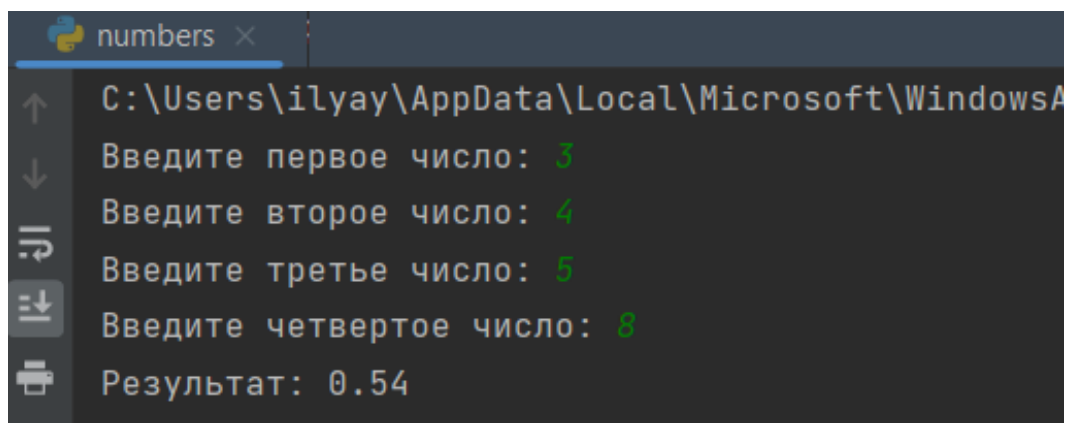
Рисунок 11 – Запуск программы (2)

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:



```
1 num1 = float(input("Введите первое число: "))
2 num2 = float(input("Введите второе число: "))
3 num3 = float(input("Введите третье число: "))
4 num4 = float(input("Введите четвертое число: "))
5
6 sum1 = num1 + num2
7 sum2 = num3 + num4
8
9 result = sum1 / sum2
10 print("Результат:", round(result, 2))
```

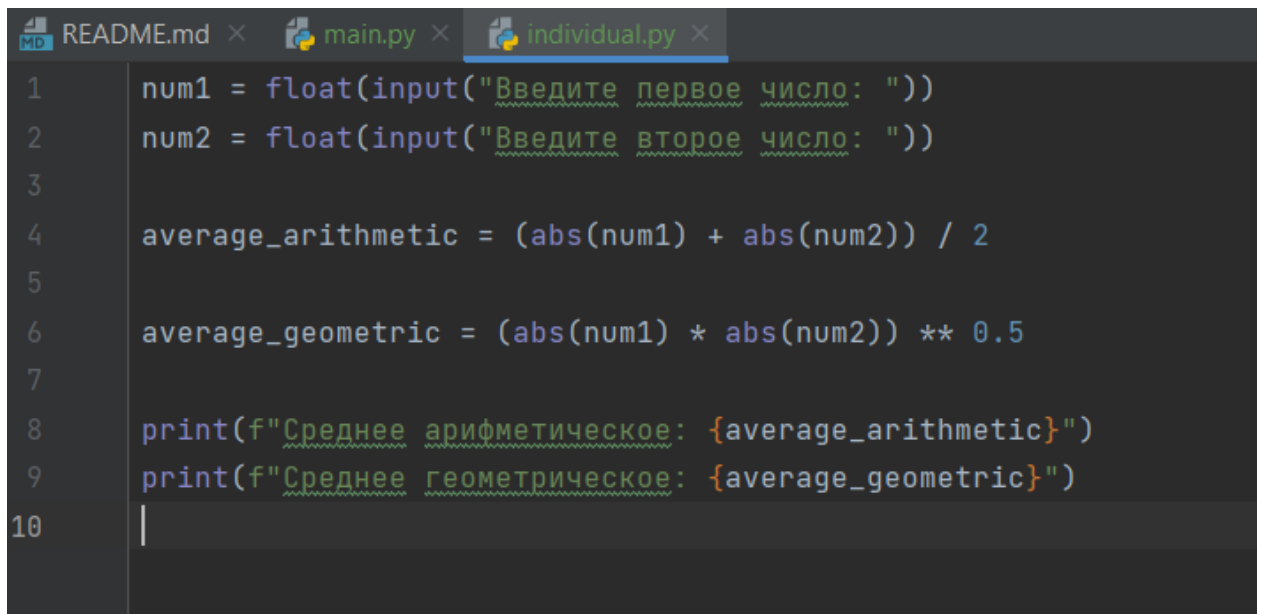
Рисунок 12 – Программа, запрашивающая у пользователя числа и выводящая результат деления первой и последней сумм



```
numbers x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsA
Введите первое число: 3
Введите второе число: 4
Введите третье число: 5
Введите четвертое число: 8
Результат: 0.54
```

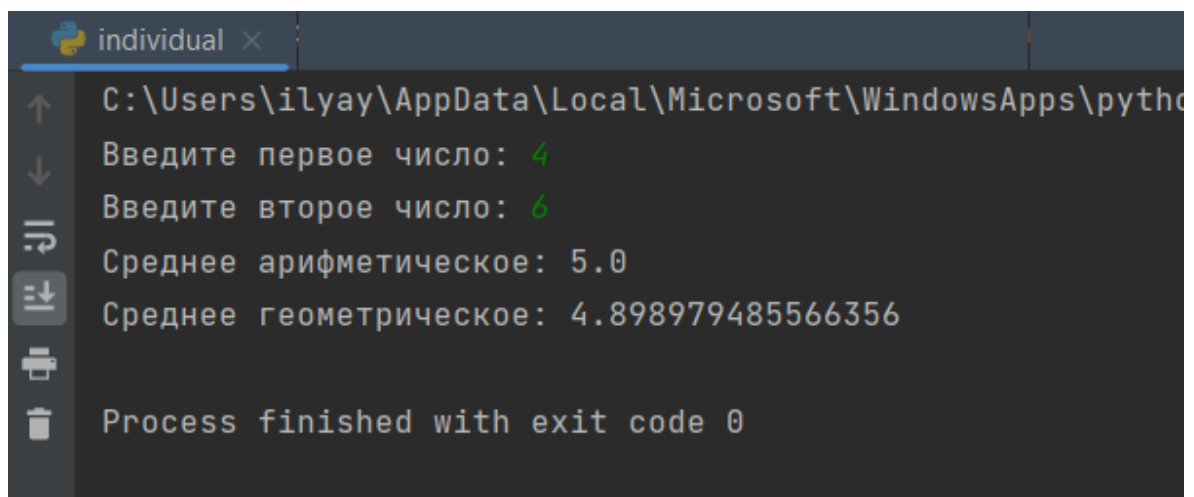
Рисунок 13 – Запуск программы

6. Напишите программу (файл individual.py) для решения индивидуального задания. Даны два числа. Найти среднее арифметическое и среднее геометрическое их модулей:



```
1 num1 = float(input("Введите первое число: "))
2 num2 = float(input("Введите второе число: "))
3
4 average_arithmetic = (abs(num1) + abs(num2)) / 2
5
6 average_geometric = (abs(num1) * abs(num2)) ** 0.5
7
8 print(f"Среднее арифметическое: {average_arithmetic}")
9 print(f"Среднее геометрическое: {average_geometric}")
10
```

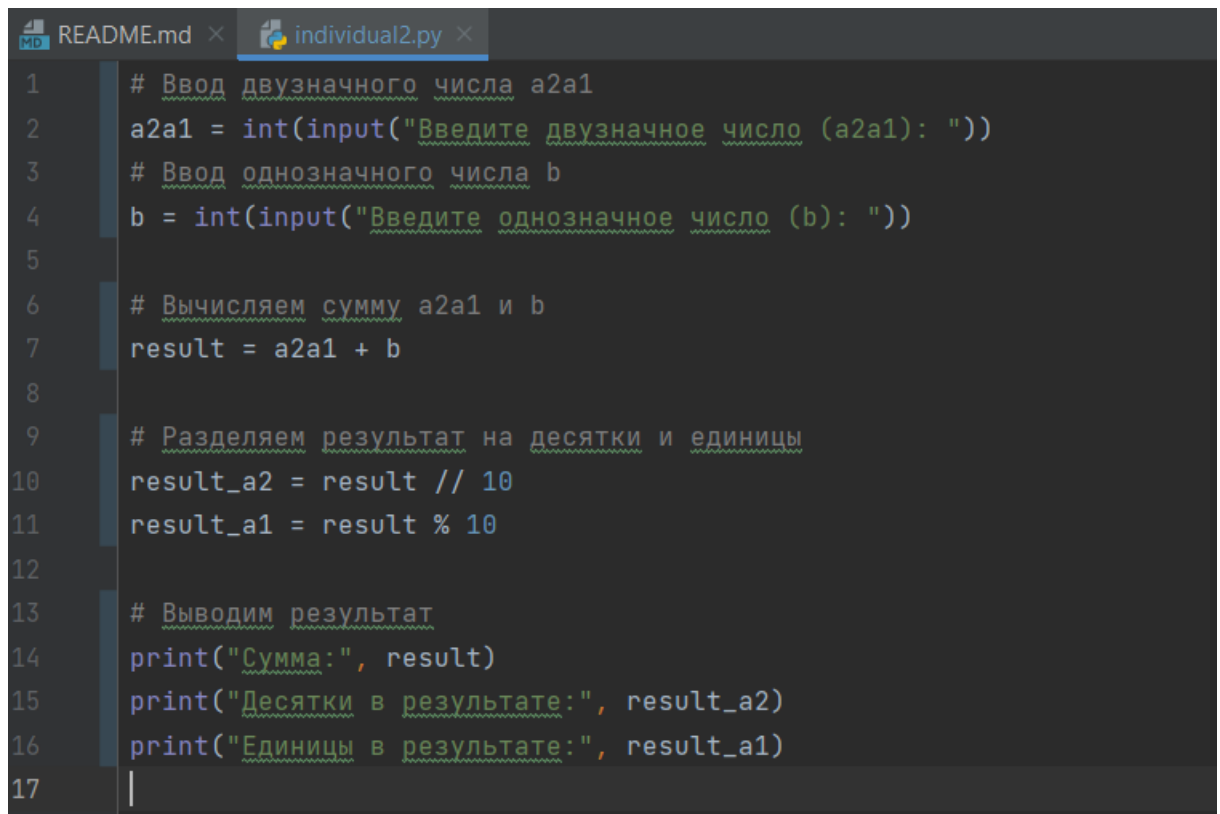
Рисунок 14 – Программа нахождения среднего арифметического и среднего геометрического их модулей двух чисел



```
individual x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python
Введите первое число: 4
Введите второе число: 6
Среднее арифметическое: 5.0
Среднее геометрическое: 4.898979485566356
Process finished with exit code 0
```

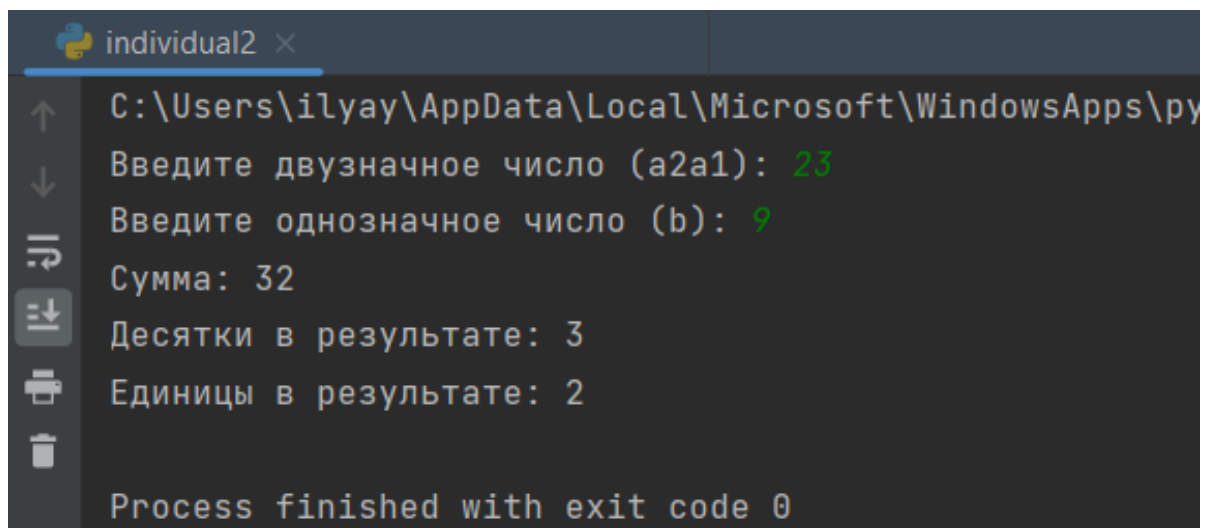
Рисунок 15 – Запуск программы

7. Задача повышенной сложности. Даны цифры двух целых чисел: двузначного a_2a_1 и однозначного b , где a_1 - число единиц, a_2 – число десятков. Получить цифры числа, равного сумме заданных чисел (известно, что это число двузначное). Условный оператор не использовать.



```
1 # Ввод двузначного числа a2a1
2 a2a1 = int(input("Введите двузначное число (a2a1): "))
3 # Ввод однозначного числа b
4 b = int(input("Введите однозначное число (b): "))
5
6 # Вычисляем сумму a2a1 и b
7 result = a2a1 + b
8
9 # Разделяем результат на десятки и единицы
10 result_a2 = result // 10
11 result_a1 = result % 10
12
13 # Выводим результат
14 print("Сумма:", result)
15 print("Десятки в результате:", result_a2)
16 print("Единицы в результате:", result_a1)
17 |
```

Рисунок 16 – Программа для получения цифр числа, равного сумме двух заданных чисел без использования условных операторов



```
individual2 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\py
Введите двузначное число (a2a1): 23
Введите однозначное число (b): 9
Сумма: 32
Десятки в результате: 3
Единицы в результате: 2
Process finished with exit code 0
```

Рисунок 17 – Запуск программы

8. Коммит всех созданных файлов:

```

$ git log
commit 2463eca7f6d1567643c4549374e1f04d7358d63d (HEAD -> main, origin/main, origin/HEAD, develop)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Tue Oct 10 02:27:43 2023 +0300

    Добавление программы повышенного уровня

commit abc5d2494afc25f455c9f8f728d851fe01101885
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Tue Oct 10 02:26:07 2023 +0300

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

commit 3b4356048cb85ef35e8d70e7f048af0c0b37fdb
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date: Tue Oct 10 01:14:24 2023 +0300

    Initial commit

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/1r4 (main)

```

Рисунок 18 – Коммиты ветки develop во время выполнения лабораторной работы

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

```

$ git merge develop
Updating 3b43560..2463eca
Fast-forward
 arithmetic.py | 17 ++++++++
 individual.py | 9 +++++
 individual2.py | 14 ++++++++
 main.py       | 0
 numbers.py    | 11 ++++++++
 user.py       | 7 +++++
 6 files changed, 58 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

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/1r4 (main)

```

Рисунок 19 – Слияние ветки develop в ветку main

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/1r4 (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), 1.80 KiB | 1.80 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/daxstrong/1r4.git
 3b43560..2463eca main -> main

```

Рисунок 20 – 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).

10. 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 и вещественной переменной в языке Python?

`int(input())` или `float(input())`.