

**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  
«Основы языка Python»  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » октября 2022 г.

Подпись студента \_\_\_\_\_

Работа защищена

« » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь, 2022

## Цель работы:

Исследование процесса установки и базовых возможностей языка Python версии 3.x.

## Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

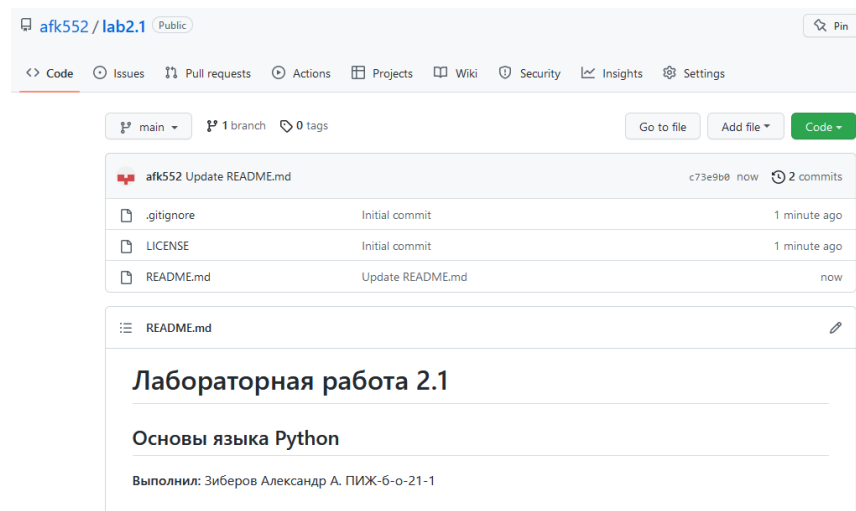


Рисунок 1 – Репозиторий GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

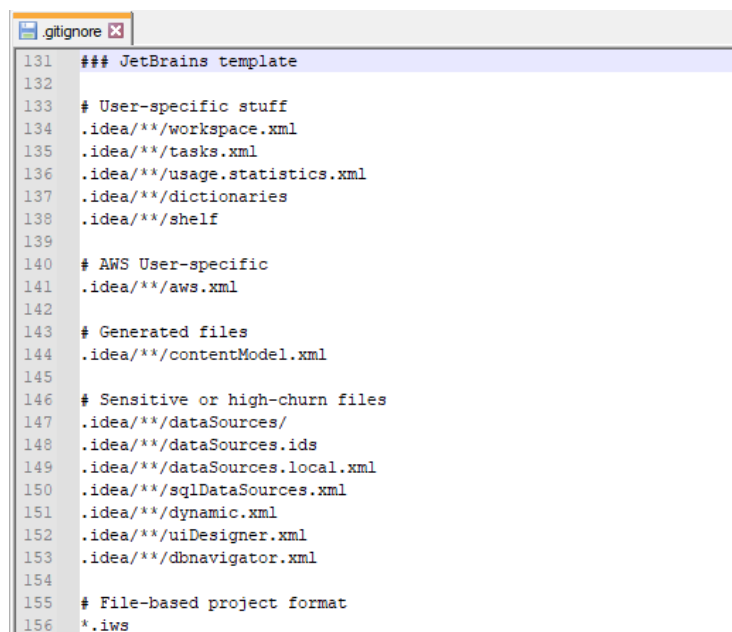
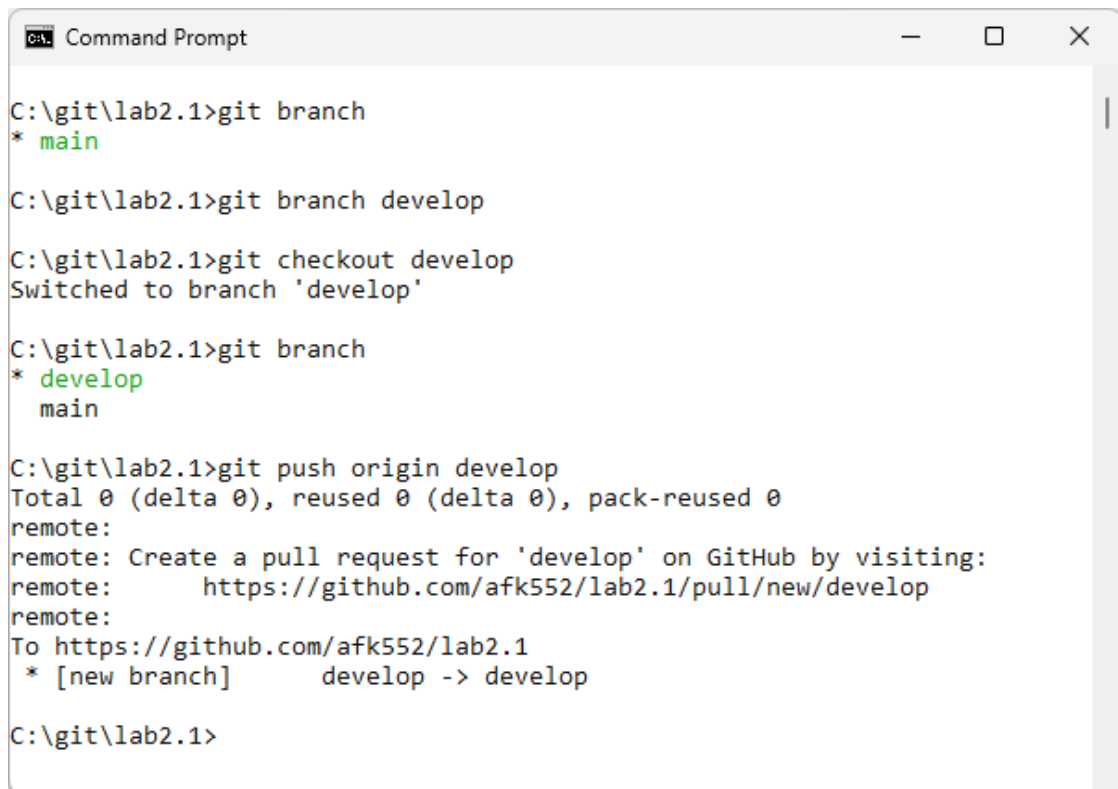


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\git\lab2.1>git branch
* main

C:\git\lab2.1>git branch develop

C:\git\lab2.1>git checkout develop
Switched to branch 'develop'

C:\git\lab2.1>git branch
* develop
  main

C:\git\lab2.1>git push 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/afk552/lab2.1/pull/new/develop
remote:
To https://github.com/afk552/lab2.1
 * [new branch]      develop -> develop

C:\git\lab2.1>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

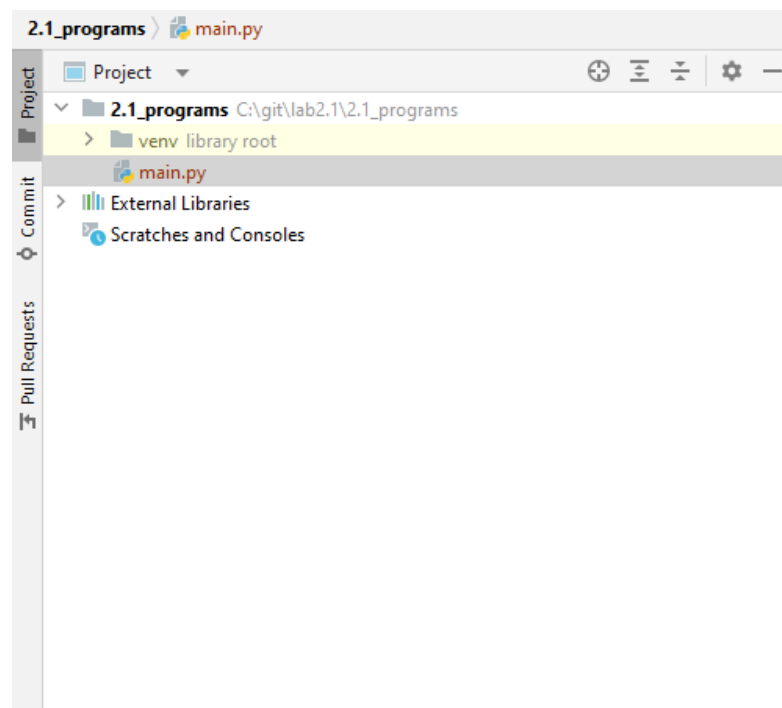
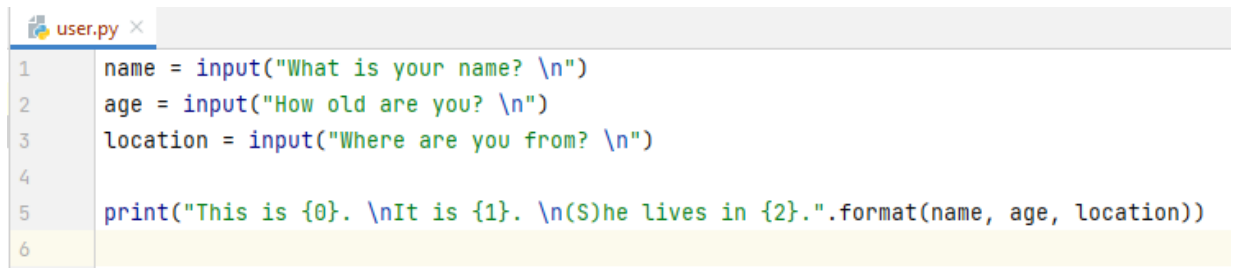


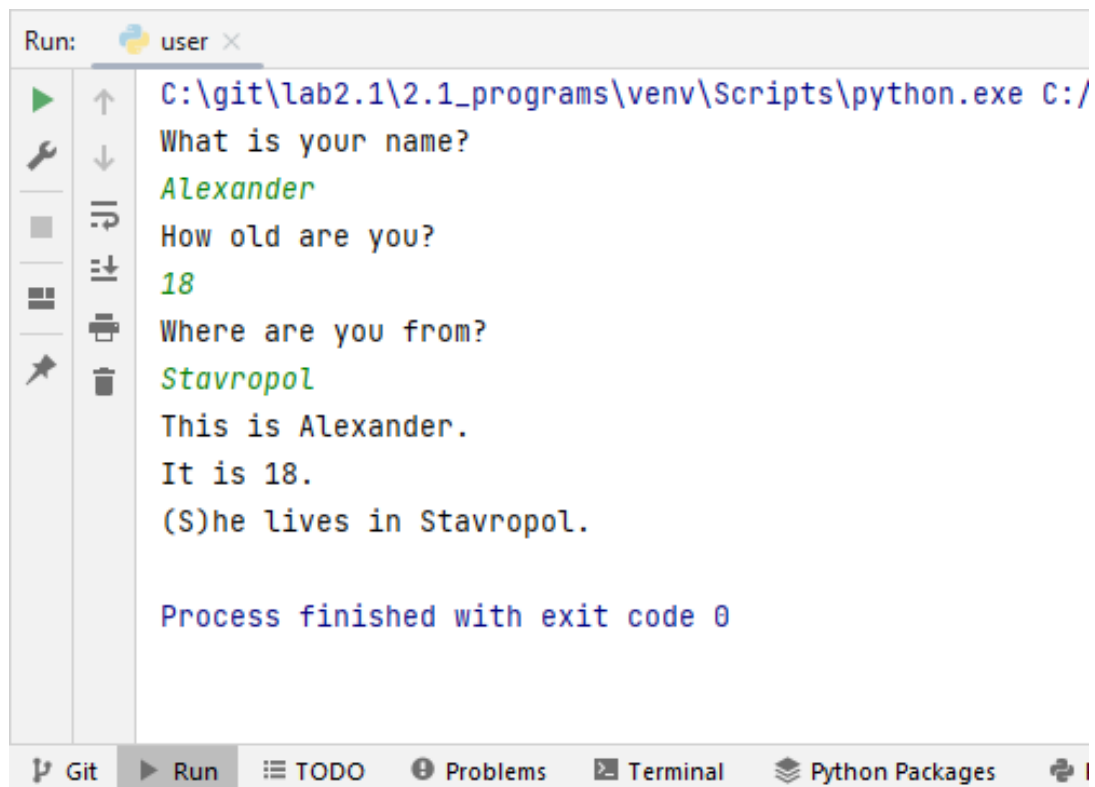
Рисунок 4 – Окно проекта в PyCharm

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 с введенными пользователем данными. Рисунки 5, 6.



```
1 name = input("What is your name? \n")
2 age = input("How old are you? \n")
3 location = input("Where are you from? \n")
4
5 print("This is {0}. \nIt is {1}. \n(S)he lives in {2}.".format(name, age, location))
6
```

Рисунок 5 – Окно редактора кода PyCharm

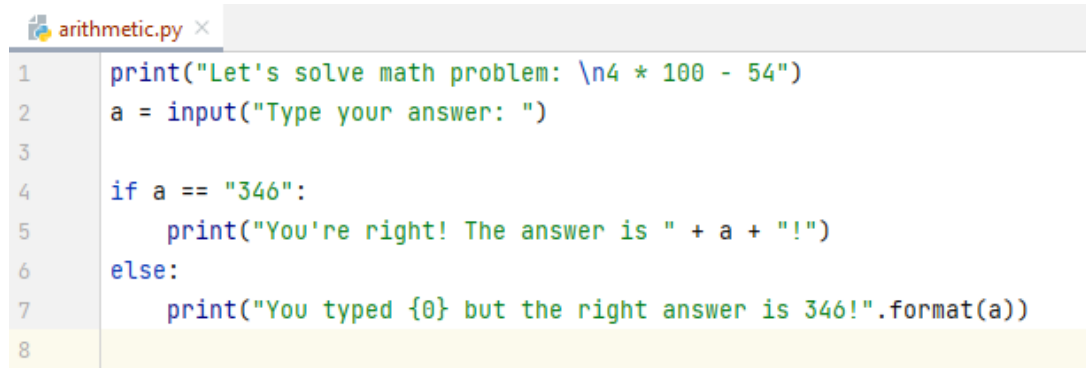


```
Run: user x
C:\git\lab2.1\2.1_programs\venv\Scripts\python.exe C:/
What is your name?
Alexander
How old are you?
18
Where are you from?
Stavropol
This is Alexander.
It is 18.
(S)he lives in Stavropol.

Process finished with exit code 0
```

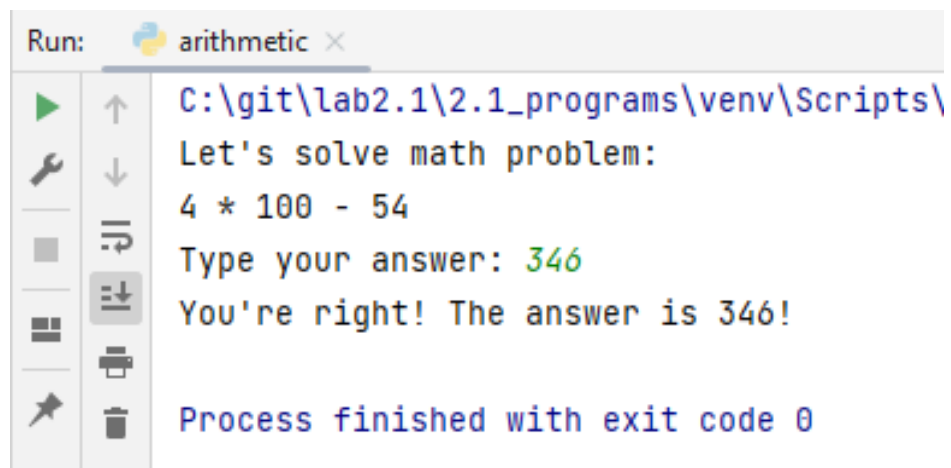
Рисунок 6 – Результат выполнения программы user.py

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. Рисунки 7-9.



```
1 print("Let's solve math problem: \n4 * 100 - 54")
2 a = input("Type your answer: ")
3
4 if a == "346":
5     print("You're right! The answer is " + a + "!")
6 else:
7     print("You typed {0} but the right answer is 346!".format(a))
8
```

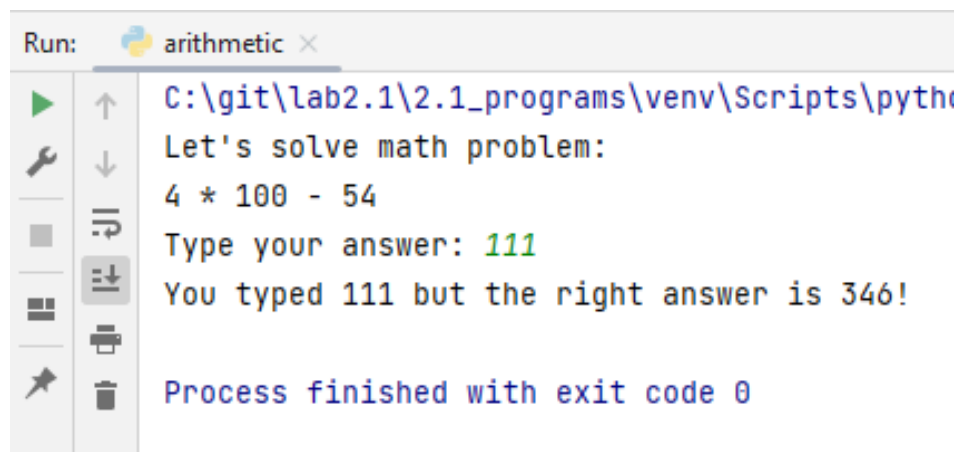
Рисунок 7 – Окно редактора кода PyCharm



```
Run: arithmetic x
C:\git\lab2.1\2.1_programs\venv\Scripts\
Let's solve math problem:
4 * 100 - 54
Type your answer: 346
You're right! The answer is 346!

Process finished with exit code 0
```

Рисунок 8 – Результат выполнения программы arithmetic.py

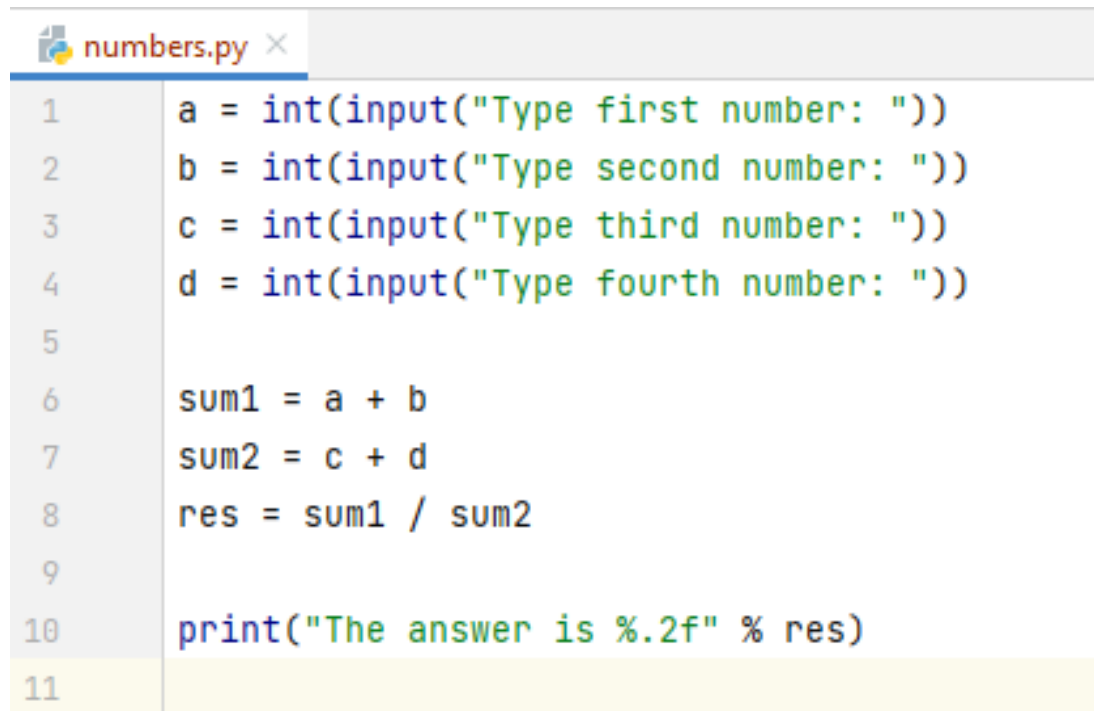


```
Run: arithmetic x
C:\git\lab2.1\2.1_programs\venv\Scripts\pyth
Let's solve math problem:
4 * 100 - 54
Type your answer: 111
You typed 111 but the right answer is 346!

Process finished with exit code 0
```

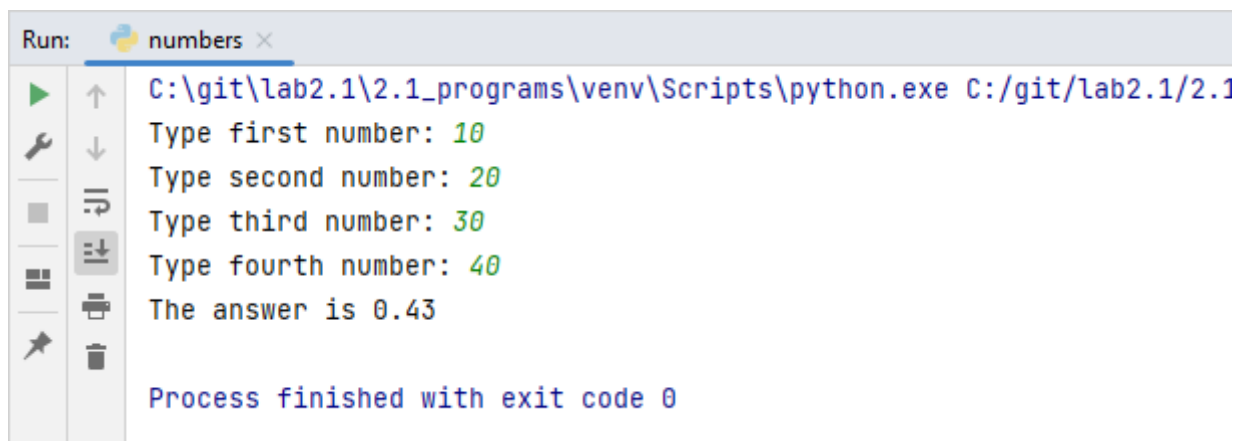
Рисунок 9 – Результат выполнения программы arithmetic.py

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. Рисунки 10, 11.



```
1 a = int(input("Type first number: "))
2 b = int(input("Type second number: "))
3 c = int(input("Type third number: "))
4 d = int(input("Type fourth number: "))
5
6 sum1 = a + b
7 sum2 = c + d
8 res = sum1 / sum2
9
10 print("The answer is %.2f" % res)
11
```

Рисунок 10 – Окно редактора кода PyCharm



```
Run: numbers x
C:\git\lab2.1\2.1_programs\venv\Scripts\python.exe C:/git/lab2.1/2.1
Type first number: 10
Type second number: 20
Type third number: 30
Type fourth number: 40
The answer is 0.43

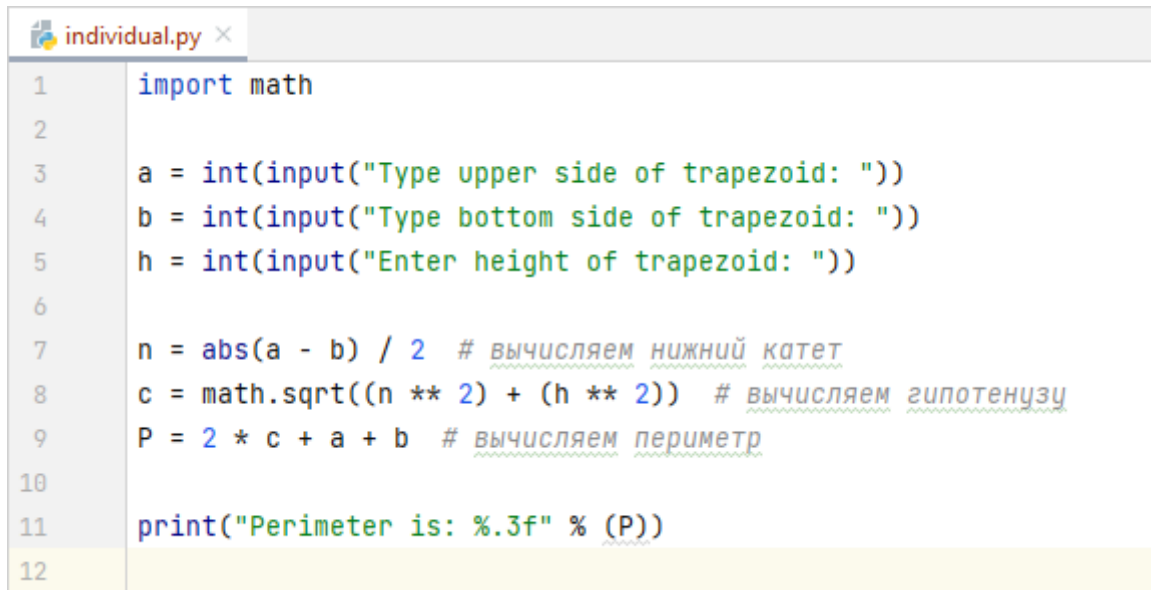
Process finished with exit code 0
```

Рисунок 11 – Результат выполнения программы numbers.py

Напишите программу (файл individual.py) для решения индивидуального задания. Вариант индивидуального задания уточните у преподавателя. Рисунки 12, 13.

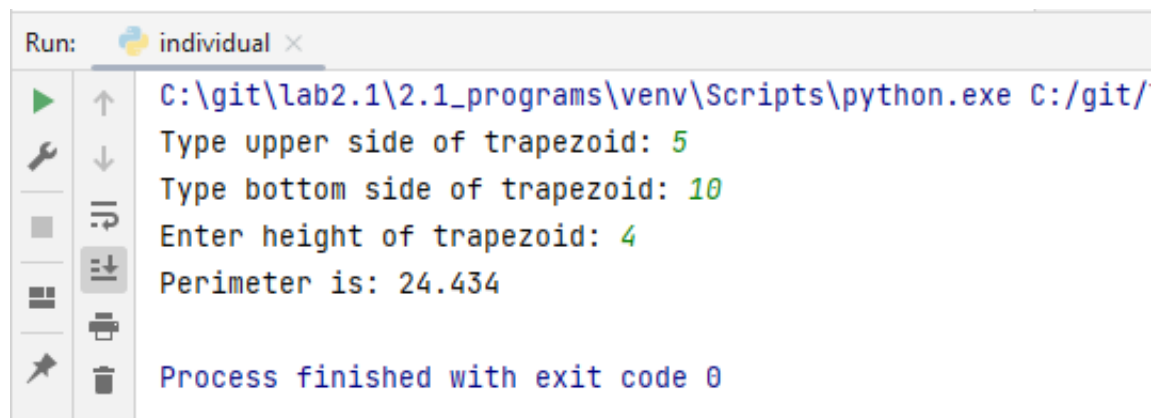
Вариант 7:

Даны основания и высота равнобедренной трапеции. Найти периметр трапеции.



```
1 import math
2
3 a = int(input("Type upper side of trapezoid: "))
4 b = int(input("Type bottom side of trapezoid: "))
5 h = int(input("Enter height of trapezoid: "))
6
7 n = abs(a - b) / 2 # вычисляем нижний катет
8 c = math.sqrt((n ** 2) + (h ** 2)) # вычисляем гипотенузу
9 P = 2 * c + a + b # вычисляем периметр
10
11 print("Perimeter is: %.3f" % (P))
12
```

Рисунок 12 – Окно редактора кода PyCharm



```
Run: individual ×
C:\git\lab2.1\2.1_programs\venv\Scripts\python.exe C:/git/
Type upper side of trapezoid: 5
Type bottom side of trapezoid: 10
Enter height of trapezoid: 4
Perimeter is: 24.434
Process finished with exit code 0
```

Рисунок 13 – Результат выполнения программы individual.py

Выполните коммит файлов `user.py`, `arithmetic.py`, `numbers.py` и `individual.py` в репозиторий `git` в ветку для разработки, рисунки 14, 15.

```
C:\git\lab2.1>git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    2.1_programs/

nothing added to commit but untracked files present (use "git add" to track)

C:\git\lab2.1>git add .

C:\git\lab2.1>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.1_programs/arithmetic.py
    new file:   2.1_programs/individual.py
    new file:   2.1_programs/numbers.py
    new file:   2.1_programs/user.py

C:\git\lab2.1>git branch
* develop
  main

C:\git\lab2.1>git commit -m "Add programs"
[develop 3e30e0f] Add programs
4 files changed, 33 insertions(+)
create mode 100644 2.1_programs/arithmetic.py
create mode 100644 2.1_programs/individual.py
create mode 100644 2.1_programs/numbers.py
create mode 100644 2.1_programs/user.py

C:\git\lab2.1>git push origin develop
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.15 KiB | 1.15 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.1
    21817aa..3e30e0f  develop -> develop

C:\git\lab2.1>
```

Рисунок 14 – Окно командной строки

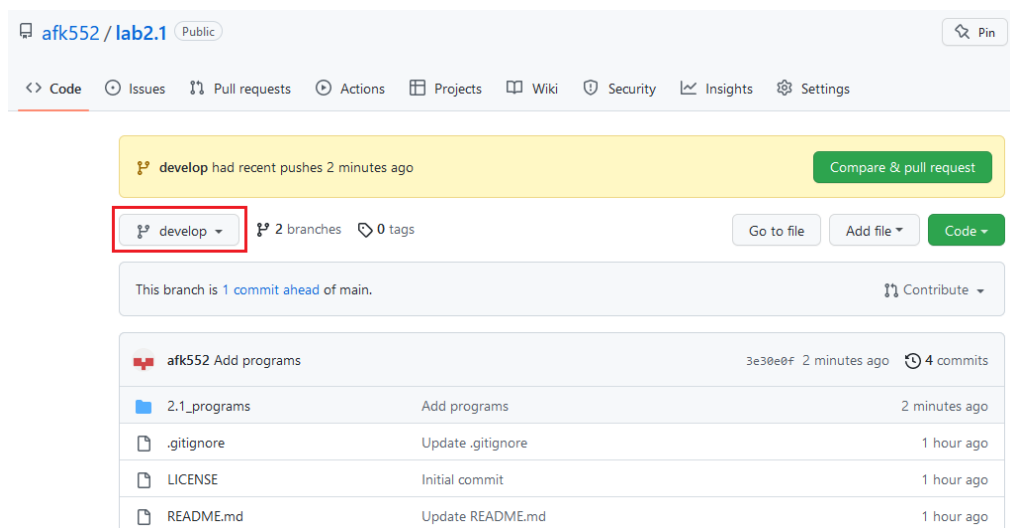
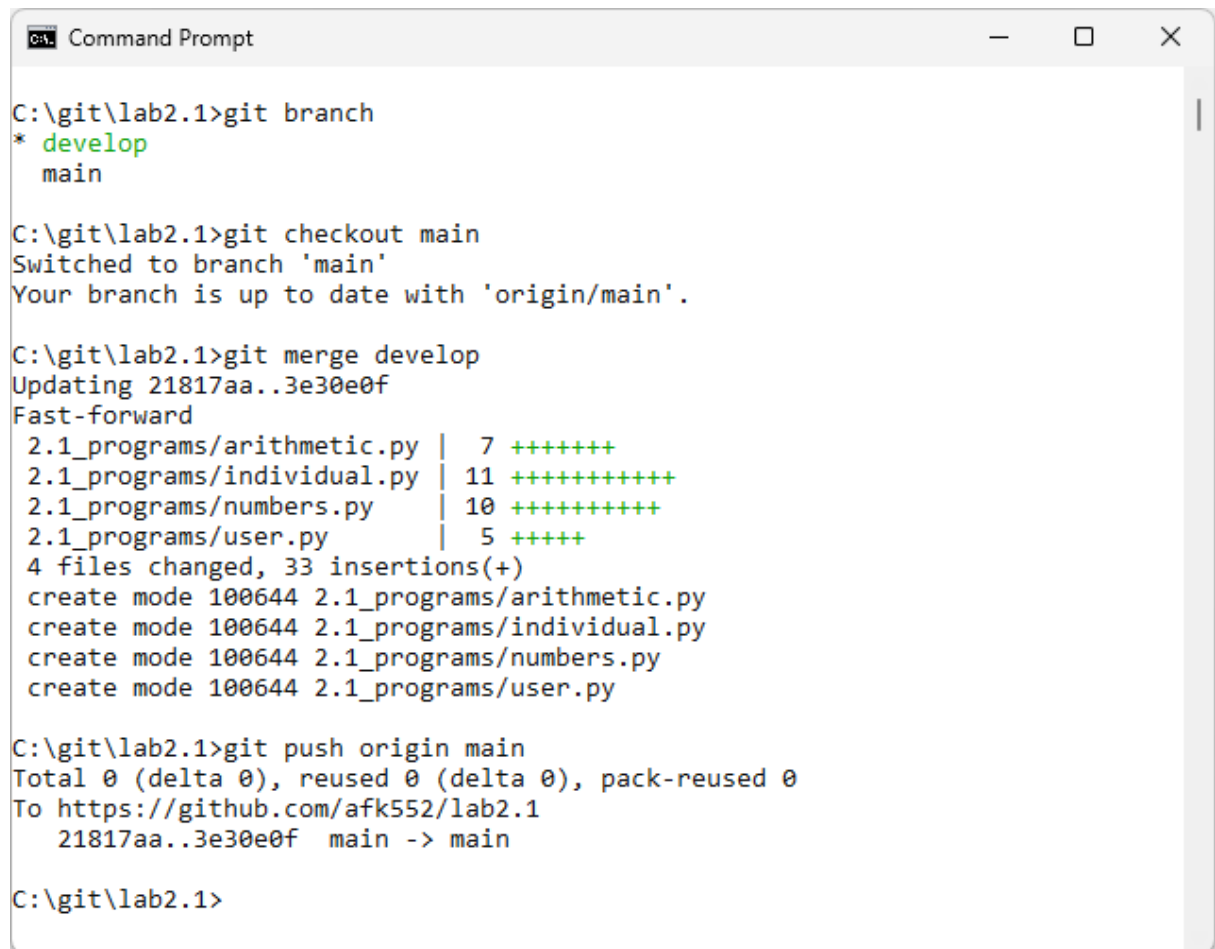


Рисунок 15 – Удаленный репозиторий на GitHub



Выполните слияние ветки для разработки с веткой master (main), отправьте изменения на сервер. Рисунки 16, 17.



```
C:\git\lab2.1>git branch
* develop
  main

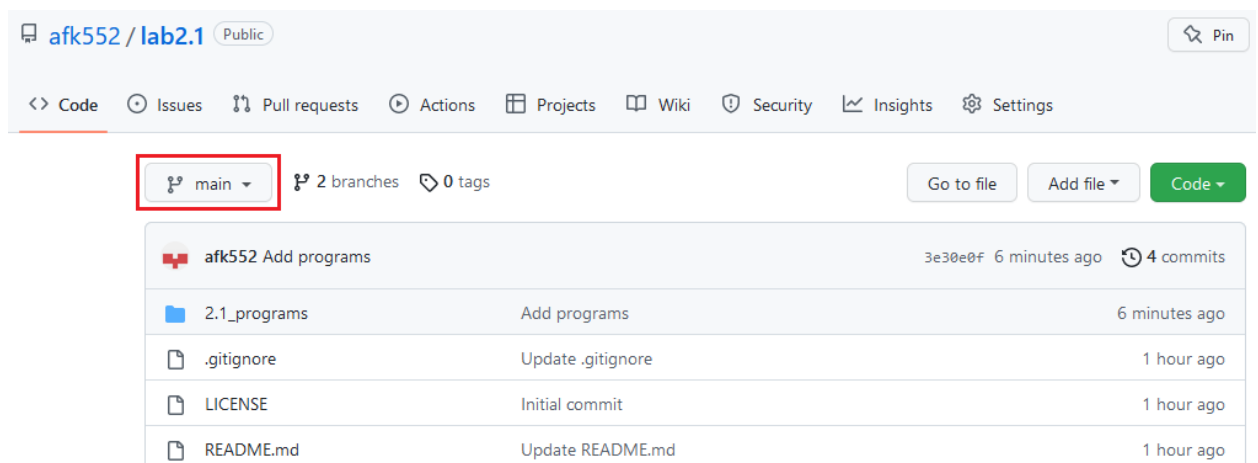
C:\git\lab2.1>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.1>git merge develop
Updating 21817aa..3e30e0f
Fast-forward
 2.1_programs/arithmetic.py | 7 ++++++
 2.1_programs/individual.py | 11 ++++++++
 2.1_programs/numbers.py    | 10 ++++++++
 2.1_programs/user.py       | 5 +++++
 4 files changed, 33 insertions(+)
 create mode 100644 2.1_programs/arithmetic.py
 create mode 100644 2.1_programs/individual.py
 create mode 100644 2.1_programs/numbers.py
 create mode 100644 2.1_programs/user.py

C:\git\lab2.1>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.1
 21817aa..3e30e0f  main -> main

C:\git\lab2.1>
```

Рисунок 16 – Окно командной строки



afk552 / lab2.1 Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags

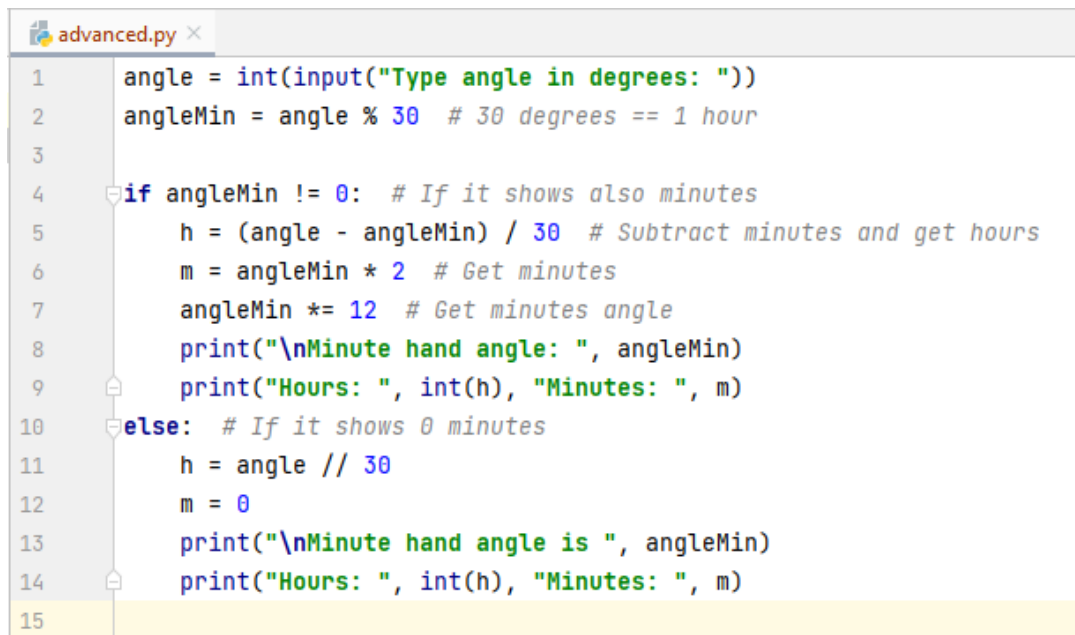
Go to file Add file Code

afk552	Add programs	3e30e0f 6 minutes ago	4 commits
2.1_programs	Add programs	6 minutes ago	
.gitignore	Update .gitignore	1 hour ago	
LICENSE	Initial commit	1 hour ago	
README.md	Update README.md	1 hour ago	

Рисунок 17 – Удаленный репозиторий на GitHub

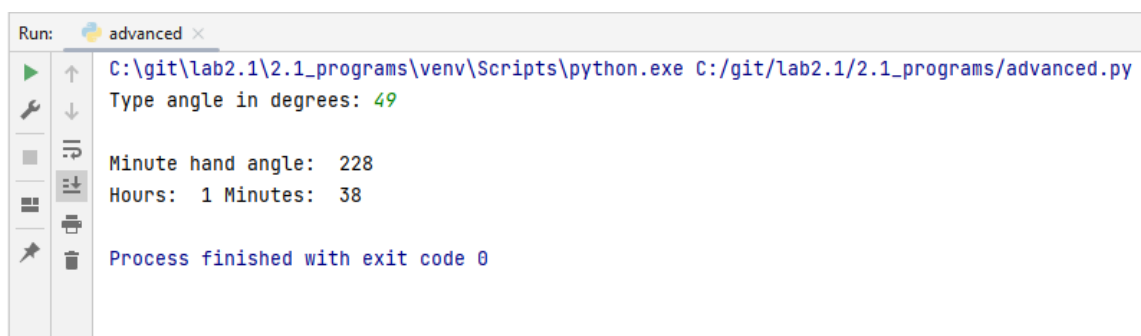
## Задание повышенной сложности

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



```
1  angle = int(input("Type angle in degrees: "))
2  angleMin = angle % 30 # 30 degrees == 1 hour
3
4  if angleMin != 0: # If it shows also minutes
5      h = (angle - angleMin) / 30 # Subtract minutes and get hours
6      m = angleMin * 2 # Get minutes
7      angleMin *= 12 # Get minutes angle
8      print("\nMinute hand angle: ", angleMin)
9      print("Hours: ", int(h), "Minutes: ", m)
10 else: # If it shows 0 minutes
11     h = angle // 30
12     m = 0
13     print("\nMinute hand angle is ", angleMin)
14     print("Hours: ", int(h), "Minutes: ", m)
15
```

Рисунок 18 – Окно редактора кода PyCharm



```
Run: advanced x
C:\git\lab2.1\2.1_programs\venv\Scripts\python.exe C:/git/lab2.1/2.1_programs/advanced.py
Type angle in degrees: 49
Minute hand angle: 228
Hours: 1 Minutes: 38
Process finished with exit code 0
```

Рисунок 19 – Результат выполнения программы advanced.py

**Вывод:** Были изучены основы языка Python версии 3, а именно: типы данных, арифметические и битовые операции, библиотека math, ввод/вывод данных. В процессе выполнения работы были установлены IDE для Python (IDLE, PyCharm, Anaconda). Загрузка файлов программ на GitHub

производилась через git по модели git-flow с разделением на главную ветку и ветку разработки (develop).

### **Контрольные вопросы:**

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

На Windows:

1. Запустить установочный файл Python.
2. Выбрать способ установки.
3. Отметить необходимые опции установки (доступно при выборе Customize installation).
4. Выбрать место установки (доступно при выборе Customize installation).

На дистрибутивах Linux:

```
sudo apt-get install python3
```

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

Этот пакет 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?**

Открыть меню Пуск, Anaconda3 (64-bit), Anaconda Prompt. В командной строке ввести: `jupyter notebook`.

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

PyCharm по умолчанию устанавливает свой интерпретатор Python и использует его, но при желании, его можно изменить на другой. Это можно

сделать как при создании проекта, так и в существующем проекте в его настройках (Надпись с версией Python в нижнем правом углу).

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

Нажать на кнопку «Run» в окне PyCharm выше редактора кода.

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

Интерактивный режим работы позволяет использовать Python как калькулятор для различных вычислений, при этом не запоминая код программы в файл.

Пакетный режим работы исполняет код непосредственно из файла.

## **7. Почему язык программирования Python называется языком динамической типизации?**

Потому что при объявлении переменных не указывается их тип данных.

## **8. Какие существуют основные типы в языке программирования Python?**

К основным встроенным типам относятся:

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
  1. int – целое число
  2. float – число с плавающей точкой
  3. complex – комплексное число
4. Списки (Sequence Type)
  1. list – список
  2. tuple – кортеж
  3. range – диапазон
5. Строки (Text Sequence Type )

1. str
6. Бинарные списки (Binary Sequence Types)
  1. bytes – байты
  2. bytearray – массивы байт
  3. memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (Set Types)
  1. set – множество
  2. frozenset – неизменяемое множество
8. Словари (Mapping Types)
  1. dict – словарь

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

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?**

```
import keyword  
print(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) неизменяемые множества (frozen set).

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 типа данных int, деление возвращает float.

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

Создание комплексного числа.

```
z = 1 + 2j
```

```
x = complex(3, 2)
```

Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень.

`x + z`

`x ** z`

`x ** 3`

У комплексного числа можно извлечь действительную и мнимую части.

`x.real`, `x.imag`

Для получения комплексносопряженного 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 библиотеке `math`, в которой содержится большое количество часто используемых математических функций.

`math.ceil(x)` `math.fabs(x)` `math.factorial(x)`

`math.floor(x)` `math.exp(x)` `math.log2(x)`

`math.log10(x)` `math.log(x[, base])` `math.pow(x, y)`

`math.sqrt(x)` `math.cos(x)` `math.sin(x)`

`math.pi` `math.e`

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

`sep` – разделитель между данными

`end` – символ в конце 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.**

Форматирование по стандартам C – `print(“%f” % (2.3))`

Метод `.format` – `print(“{0}”.format(2))`

f строки – `print(f”{2}”)`

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

Целое число – `int(input())`

Вещественное – `float(input())`