

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО  
ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего  
образования  
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ  
УНИВЕРСИТЕТ»**

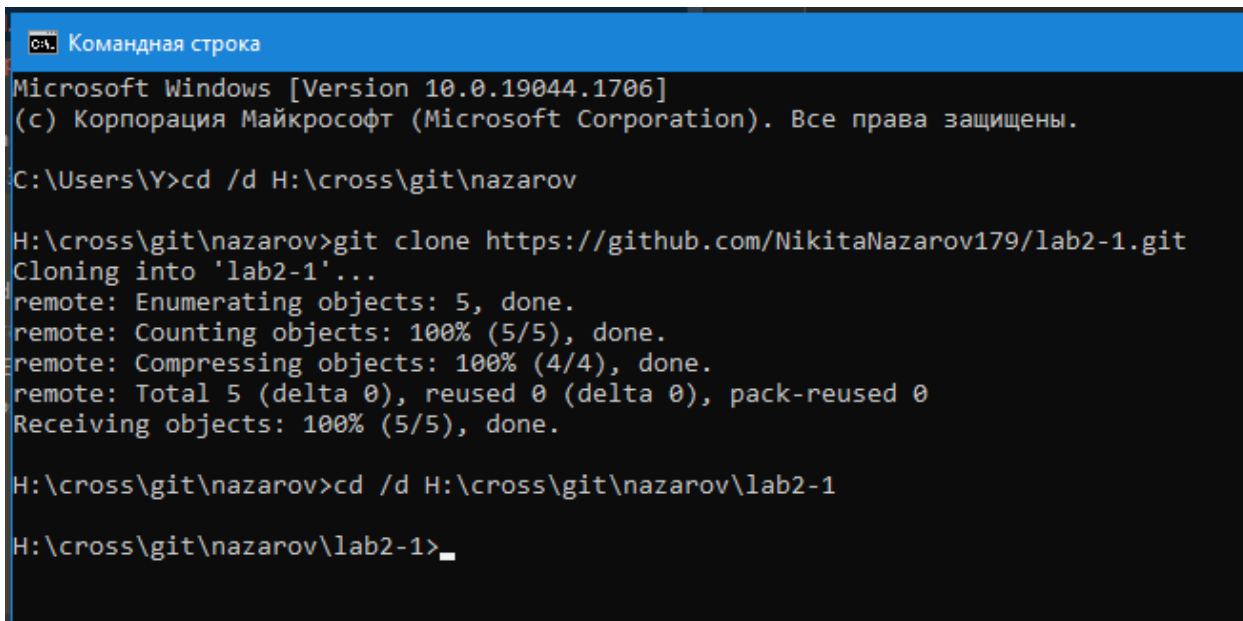
**Кафедра  
инфокоммуникаций  
Институт цифрового  
развития**

**ОТЧЁТ**  
**по лабораторной работе №2.1**  
Дисциплина: «Основы кроссплатформенного  
программирования» Тема: «Основы языка Python»

Выполнил: студент 1 курса  
группы ИВТ-б-о-21-1  
Назаров Никита Юрьевич

Ставрополь 2022

## Выполнение работы.



```
Командная строка
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

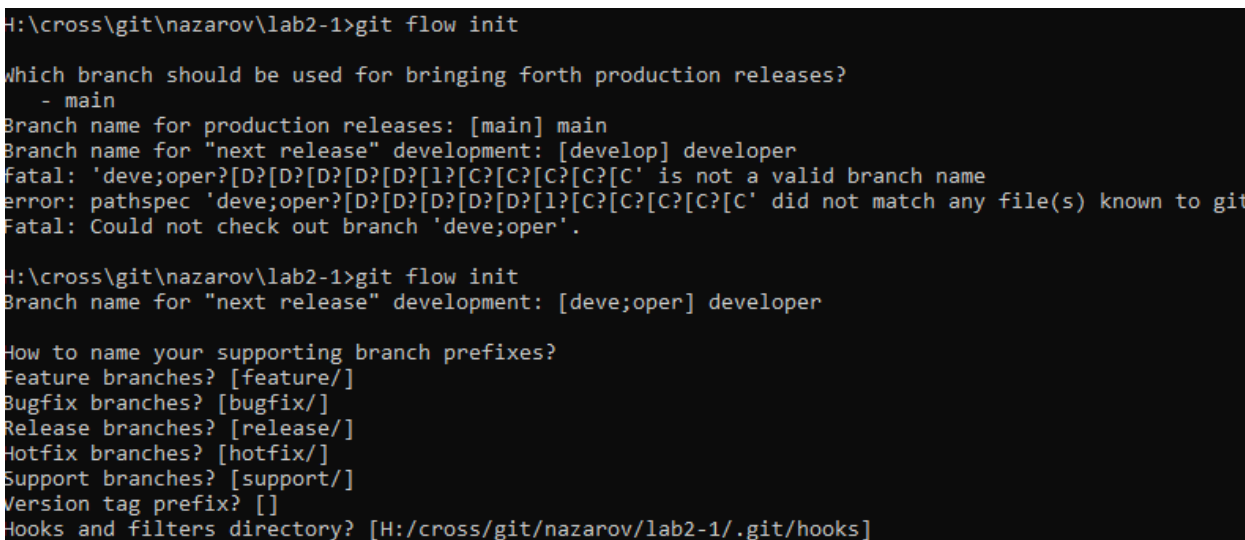
C:\Users\Y>cd /d H:\cross\git\nazarov

H:\cross\git\nazarov>git clone https://github.com/NikitaNazarov179/lab2-1.git
Cloning into '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.

H:\cross\git\nazarov>cd /d H:\cross\git\nazarov\lab2-1

H:\cross\git\nazarov\lab2-1>_
```

Рисунок 1 – клонирование репозитория



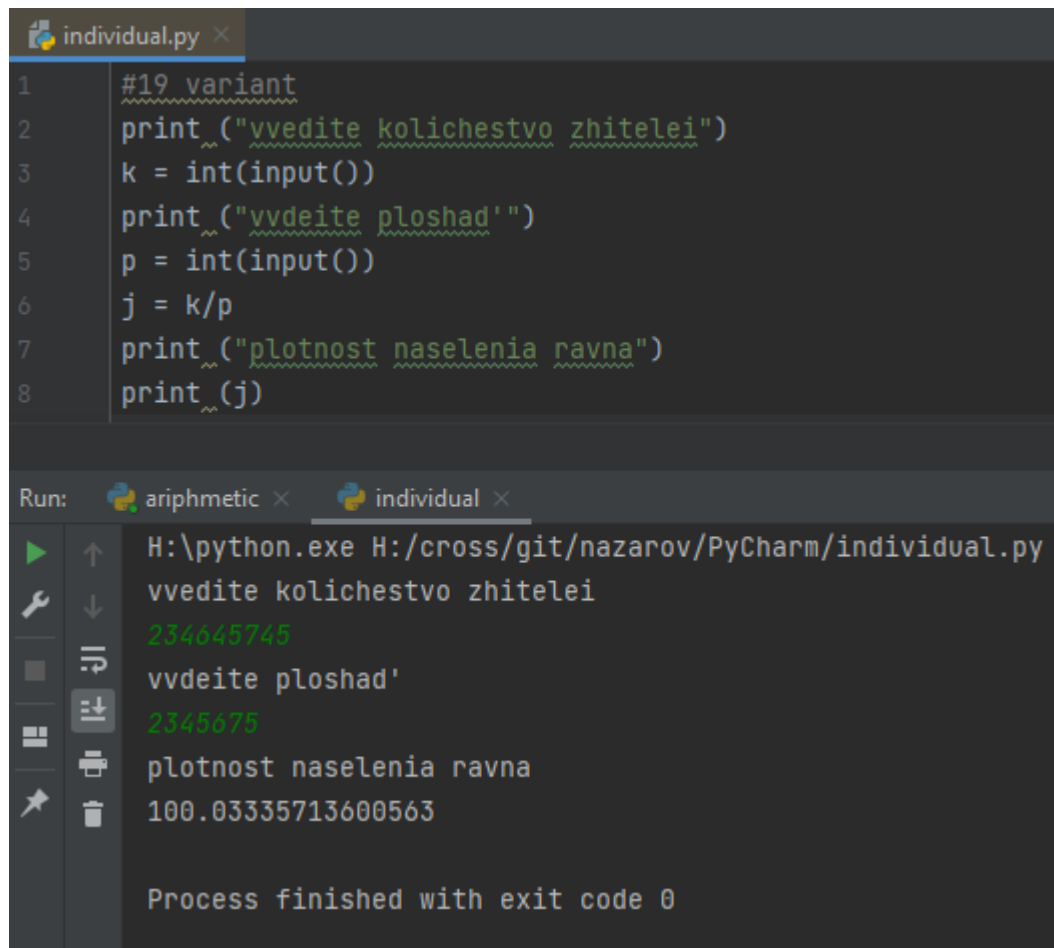
```
H:\cross\git\nazarov\lab2-1>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main] main
Branch name for "next release" development: [develop] developer
fatal: 'deve;oper?[D?[D?[D?[D?[1?[C?[C?[C?[C?[C' is not a valid branch name
error: pathspec 'deve;oper?[D?[D?[D?[D?[1?[C?[C?[C?[C?[C' did not match any file(s) known to git
Fatal: Could not check out branch 'deve;oper'.

H:\cross\git\nazarov\lab2-1>git flow init
Branch name for "next release" development: [deve;oper] developer

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [H:/cross/git/nazarov/lab2-1/.git/hooks]
```

Рисунок 2 – моделирование репозитория согласно модели ветвления git flow



The image shows a PyCharm IDE window with a file named `individual.py`. The script contains the following code:

```
1 #19 variant
2 print("vvedite kolichество zhitelei")
3 k = int(input())
4 print("vvdeite ploshad'")
5 p = int(input())
6 j = k/p
7 print("plotnost naselenia ravna")
8 print(j)
```

Below the editor, the Run console shows the execution of the script. The output is as follows:

```
Run: ariphmetic × individual ×
H:\python.exe H:/cross/git/nazarov/PyCharm/individual.py
vvedite kolichество zhitelei
234645745
vvdeite ploshad'
2345675
plotnost naselenia ravna
100.03335713600563

Process finished with exit code 0
```

Рисунок 3 – решение индивидуального задания

```
print("Vvedite 4 chisla")
p = int(input())
k = int(input())
j = int(input())
i = int(input())
n = k+p
c = i+j
z = n/c
a = round(z, 2)
print(a)
```

numbers ×

H:\python.exe H:/cross/git/nazarov/PyCharm/numbers.py

Vvedite 4 chisla

67

4567

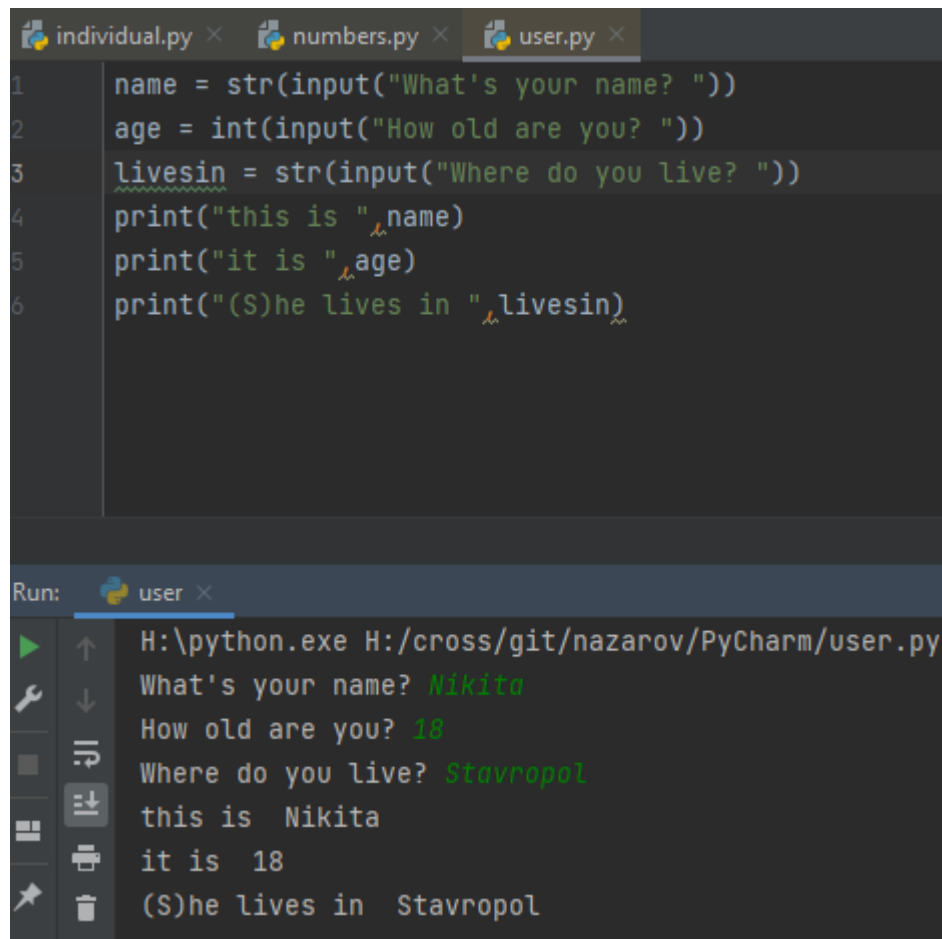
234

890

4.12

Process finished with exit code 0

Рисунок 4 – numbers.py



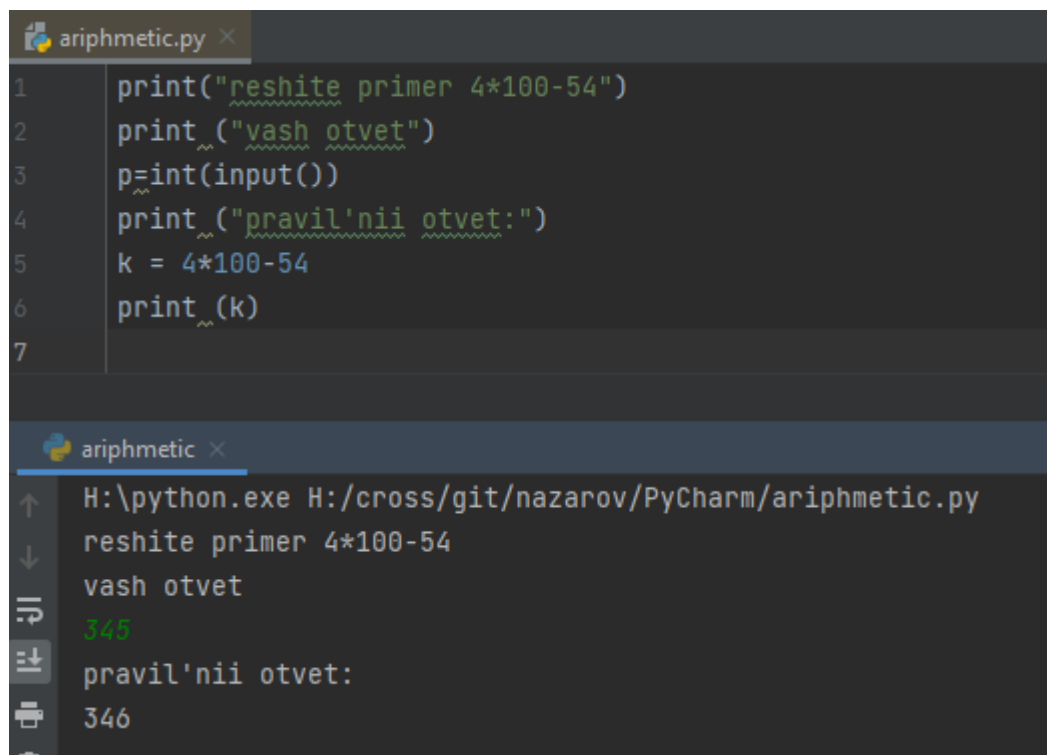
The screenshot shows the PyCharm IDE with three tabs: individual.py, numbers.py, and user.py. The user.py tab is active, displaying the following Python code:

```
1 name = str(input("What's your name? "))
2 age = int(input("How old are you? "))
3 livesin = str(input("Where do you live? "))
4 print("this is " + name)
5 print("it is " + age)
6 print("(S)he lives in " + livesin)
```

Below the code editor, the Run window shows the execution of user.py. The command prompt displays the following input and output:

```
H:\python.exe H:/cross/git/nazarov/PyCharm/user.py
What's your name? Nikita
How old are you? 18
Where do you live? Stavropol
this is Nikita
it is 18
(S)he lives in Stavropol
```

Рисунок 5 – user.py



The screenshot shows the PyCharm IDE with one tab: ariphmetic.py. The ariphmetic.py tab is active, displaying the following Python code:

```
1 print("reshite primer 4*100-54")
2 print("vash otvet")
3 p=int(input())
4 print("pravil'nii otvet:")
5 k = 4*100-54
6 print(k)
7
```

Below the code editor, the Run window shows the execution of ariphmetic.py. The command prompt displays the following input and output:

```
H:\python.exe H:/cross/git/nazarov/PyCharm/arithmetic.py
reshite primer 4*100-54
vash otvet
345
pravil'nii otvet:
346
```

Рисунок 6 – ariphmetic.py

```
H:\cross\git\nazarov\lab2-1>git add .
warning: LF will be replaced by CRLF in PyCharm/.idea/inspectionProfiles/profiles_settings.xml.
The file will have its original line endings in your working directory

H:\cross\git\nazarov\lab2-1>git commit -m "решения"
[main 5e30649] решения
10 files changed, 64 insertions(+)
create mode 100644 PyCharm/.idea/.gitignore
create mode 100644 PyCharm/.idea/PyCharm.iml
create mode 100644 PyCharm/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 PyCharm/.idea/misc.xml
create mode 100644 PyCharm/.idea/modules.xml
create mode 100644 PyCharm/.idea/vcs.xml
create mode 100644 PyCharm/ariphmetic.py
create mode 100644 PyCharm/individual.py
create mode 100644 PyCharm/numbers.py
create mode 100644 PyCharm/user.py

H:\cross\git\nazarov\lab2-1>git push
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (15/15), 2.06 KiB | 1.03 MiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NikitaNazarov179/lab2-1.git
b5303f4..5e30649 main -> main
```

Рисунок 7 – коммит и пуш решенных заданий

The screenshot shows the PyCharm IDE interface. At the top, a tab for 'high\_difficulty.py' is open. The editor displays the following Python code:

```
1
2 y = float(input('add y: '))
3 print('hours: ', y // 30)
4 print('minutes: ', y % 30 * 2)
5
```

Below the editor, the 'Run' toolbar is visible, with a green play button icon. The 'Run' console shows the execution of the script:

```
Run: high_difficulty x
H:\python.exe H:/cross/git/nazarov/lab2-1/high_difficulty.py
add y: 270
hours: 9.0
minutes: 0.0
```

Рисунок 8 – решение 6-го задания повышенной сложности

Ответы на вопросы для защиты.

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

Linux: Чаще всего интерпретатор Python уже входит в состав дистрибутива.

Windows: Осн. этапы установки Python на Windows:

- 1) Скачать дистрибутив с официального сайта;
- 2) Запустить скачанный установочный файл;
- 3) Выбрать способ установки;
- 4) Отметить необходимые опции установки;

- 5) Выбрать место установки;
- 6) Готово.

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

Пакет Anaconda содержит версии языка Python 2 и 3, набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере, а также на Anaconda удобнее запускать примеры.

**3. Как осуществить проверку работоспособности пакета Anaconda?**  
Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. 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. Создать ноутбук для разработки, для этого нажать на кнопку New и в появившемся списке выбрать Python. В результате будет создана новая страница в браузере с ноутбуком. Ввести в первой ячейке команду `print("Hello, World!")` и нажать Alt+Enter на компьютере. Ниже ячейки должна появиться соответствующая надпись.

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

Указать путь до интерпретатора в настройках IDE, для этого:

- 1) Нажмите на шестеренку в верхнем правом углу, выберите "Add..".
- 2) Далее выберите "System Interpreter";
- 3) Нажмите на 3 точки "..." справа от поля в выборе интерпретатора;
- 4) Укажите путь до интерпретатора.

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

Сочетанием клавиш Shift+F10.

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

Интерактивный.

Python можно использовать как калькулятор для различных вычислений,

а если дополнительно подключить необходимые математические библиотеки, то по своим возможностям он становится практически равным таким пакетам как Matlab, Octave и т.п.

Проектный.

В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью.

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

Т. к. в ЯП Python проверка типа происходит во время выполнения, а не компиляции.

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

Типы в ЯП Python:

1. None
2. Логические переменные
3. Числа
4. Списки
5. Строки
6. Бинарные списки
7. Множества
8. Словари

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

Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана.

При инициализации переменной, на уровне интерпретатора, создается целочисленный объект, который имеет некоторый идентификатор, значение и тип. Посредством оператора “=” создается ссылка между переменной и



объектом.

## **10. Как получить список ключевых слов в Python?**

Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

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

Функция id() предназначена для получения значения идентичности объекта.

С помощью функции type() можно получить тип конкретного объекта.

## **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 (frozen set).

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

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

При целочисленном делении отбрасывается дробная часть от деления чисел, при операции деления дробная часть не 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 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`.**

Для выполнения математических операций необходим модуль `math`.  
Осн. операции библиотеки `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 пи.

`math.e` - число `e`.

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

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода `format`.

Символы `%s` , `%d` , `%f` подставляются значения переменных. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число.

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

Указать перед `input` тип данных: `int(input())`.