

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 «Программирование на python»**

Выполнил:  
Кожуховский Виктор Андреевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных систем  
», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович

---

(подпись)

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

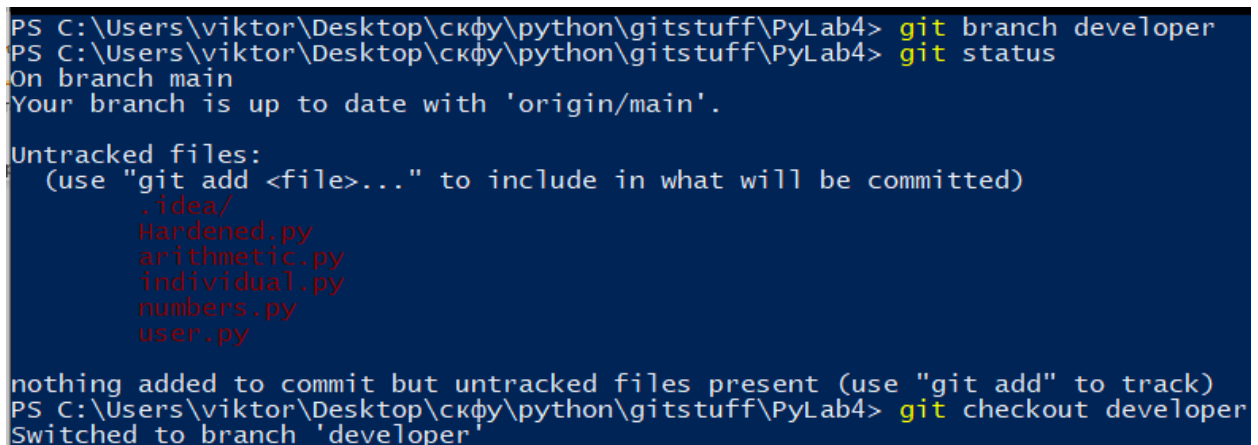
Ставрополь, 2023 г.

**Тема:** Основы языка Python.

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

### Методика и порядок выполнения работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.
2. Выполнил клонирование созданного репозитория.
3. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.
4. Организовал свой репозиторий в соответствии с моделью ветвления git-flow.



```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git branch developer
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
.idea/
Hardened.py
arithmetic.py
individual.py
numbers.py
user.py

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git checkout developer
Switched to branch 'developer'
```

Рисунок 1. Организация ветки в соответствии с моделью ветвления git-flow

5. Создал проект PyCharm в папке репозитория.
6. Решил следующие задачи с помощью языка программирования Python3 и IDE PyCharm:
7. Написал программу, которая запрашивает у пользователя: его имя, возраст, место жительства. После этого выводила бы эти три строки.

```
print("What's your name?")
a = input()
print("How old are you?")
b = input()
print("Where do you live?")
c = input()
print("This is", a, "\nThey are", b, "\nThey live in", c)
```

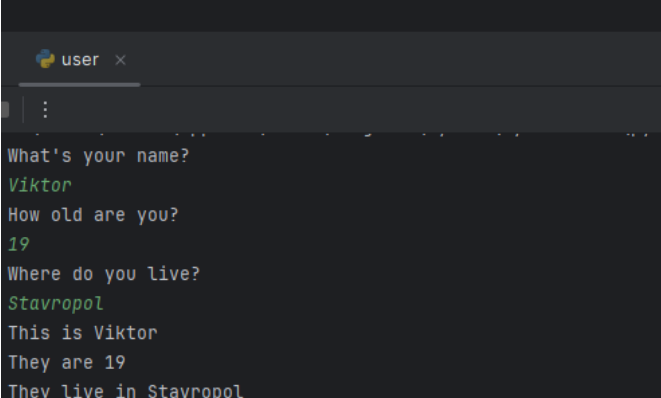


Рисунок 2. Программа, собирающая информацию о пользователе

8. Написал программу, которая предлагает пользователю решить пример  $4 * 100 - 54$ . Потом выводит на экран правильный ответ и ответ пользователя.

```
print("What's 4 * 100 - 54?")
a = input()
print("Your answer is", a, "\nCorrect answer is", 4*100-54)
if int(a) == 4*100-54:
    print("You're correct.")
else:
    print("You're wrong.")
```

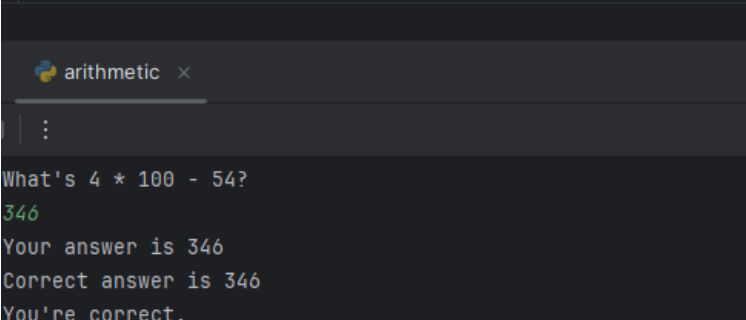


Рисунок 3. Программа арифметика

9. У пользователя запрашиваются четыре числа. Отдельно складываются первые два и отдельно вторые два. Первая сумма делится на вторую. На экран выводится результат так, чтобы ответ содержал две цифры после запятой.

```
print("Write four values:")
a = float(input())
b = float(input())
c = float(input())
d = float(input())

sum1 = a + b
sum2 = c + d
ans = sum1 / sum2

print("Answer is {:.2f}".format(ans))
```

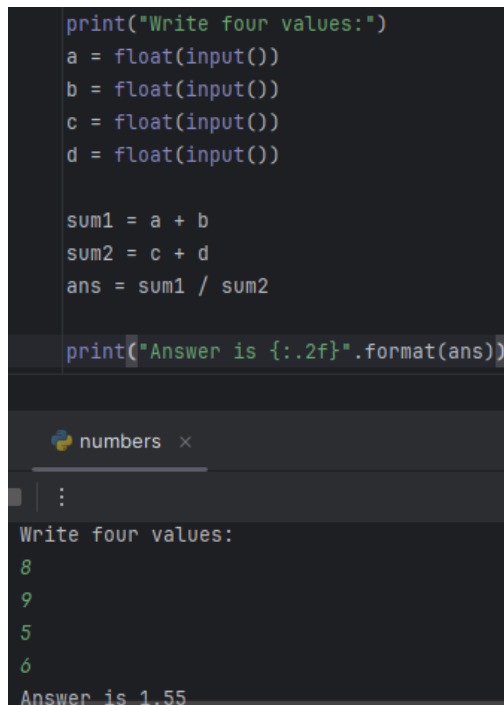


Рисунок 4. Программа, вычисляющая  $(a+b)/(c+d)$

10. Написал программу для решения индивидуального задания. Известно значение температуры по шкале Цельсия. Находятся соответствующее значение температуры по шкале Фаренгейта и Кельвина.

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

```
a = int(input())
print(a, "°C is", a*1.8+32, "degrees Fahrenheit and", a+273.15, "degrees Kelvin")
```

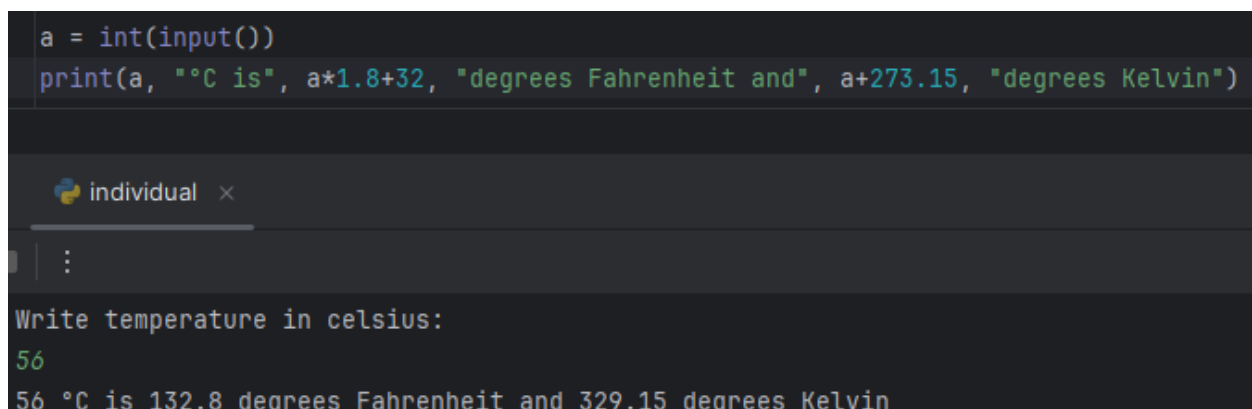
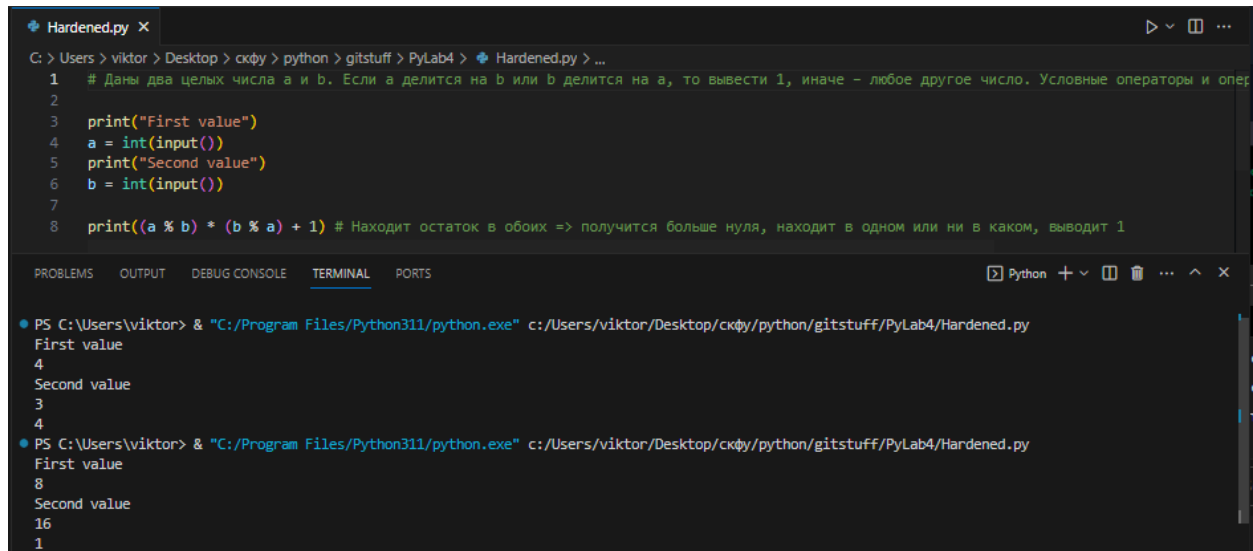


Рисунок 5. Программа перевода из цельсия в Фаренгейт и Кельвин

11. Написал программу для выполнения задания повышенной сложности:

Задание: Даны два целых числа  $a$  и  $b$ . Если  $a$  делится на  $b$  или  $b$  делится на  $a$ , то вывести 1, иначе – любое другое число. Условные операторы и операторы цикла не использовать.



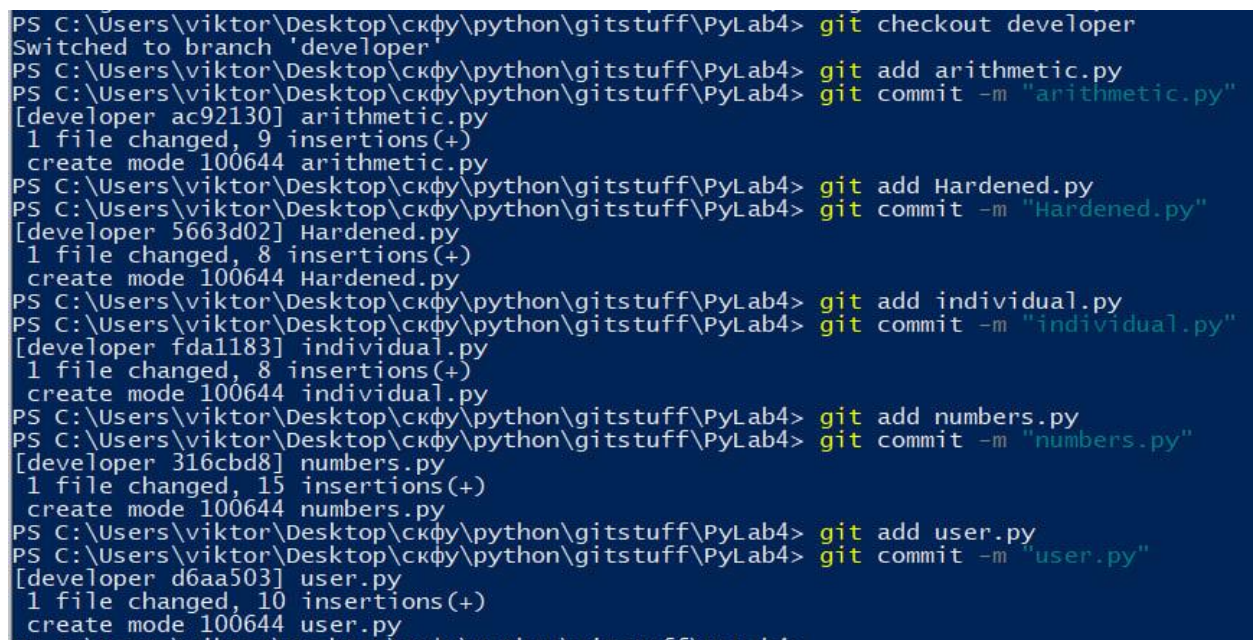
```
Hardened.py X
C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> Hardened.py ...
1 # Даны два целых числа a и b. Если a делится на b или b делится на a, то вывести 1, иначе - любое другое число. Условные операторы и опе
2
3 print("First value")
4 a = int(input())
5 print("Second value")
6 b = int(input())
7
8 print((a % b) * (b % a) + 1) # Находит остаток в обоих => получится больше нуля, находит в одном или ни в каком, выводит 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] ... ^ x

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/скфу/python/gitstuff/PyLab4/Hardened.py
First value
4
Second value
3
4
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/скфу/python/gitstuff/PyLab4/Hardened.py
First value
8
Second value
16
1
```

Рисунок 6. Программа усложненного задания

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



```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git checkout developer
Switched to branch 'developer'
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git add arithmetic.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git commit -m "arithmetic.py"
[developer ac92130] arithmetic.py
1 file changed, 9 insertions(+)
create mode 100644 arithmetic.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git add Hardened.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git commit -m "Hardened.py"
[developer 5663d02] Hardened.py
1 file changed, 8 insertions(+)
create mode 100644 Hardened.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git add individual.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git commit -m "individual.py"
[developer fda1183] individual.py
1 file changed, 8 insertions(+)
create mode 100644 individual.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git add numbers.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git commit -m "numbers.py"
[developer 316cbd8] numbers.py
1 file changed, 15 insertions(+)
create mode 100644 numbers.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git add user.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git commit -m "user.py"
[developer d6aa503] user.py
1 file changed, 10 insertions(+)
create mode 100644 user.py
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4>
```

Рисунок 7. Коммит в ветку *developer*

13. Выполнил слияние ветки для разработки с веткой *master*.

```

PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git rebase developer
Successfully rebased and updated refs/heads/main.
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\PyLab4> git log --graph --pretty=oneline
* d6aa503a258536a20f30cf26f69cf14400469383 (HEAD -> main, developer) user.py
* 316cbd80618dd76d33a02a70835d0cdd3edff276 numbers.py
* fda11836e6868df73fcd11f0cf2b3b3fecdd02674 individual.py
* 5663d025a3e9c9c498855edd8e591365f8ed9999 Hardened.py
* ac92130dc1077281ed8ccac97e41aa3e5763595f arithmetic.py
* 4919f148052fa3baa2da1a3d9fb2ac1a46ff64a (origin/main, origin/HEAD) Create README.md
* a2f8e31fce46b4ff902c55eb1c1f3ed2fa3a086a Initial commit

```

Рисунок 8. Слияние main и developer

14. Отправил сделанные изменения на сервер GitHub.

#### Вопросы для защиты работы

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

Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла (с расширением exe), либо в виде архивного файла (с расширением zip).

Порядок установки.

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

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?**

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, после чего запустится веб-сервер и среда разработки в браузере.

Создайте ноутбук для разработки, для этого нажмите на кнопку New (в правом углу окна) и в появившемся списке выберите Python.

В результате будет создана новая страница в браузере с ноутбуком. Введите в первой ячейке команду `print("Hello, World!")` и нажмите Alt+Enter на клавиатуре. Ниже ячейки должна появиться соответствующая надпись.

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

Необходимо добавить путь до интерпретатора в PATH.

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

Достаточно либо открыть файл с кодом .py, либо самой программе нажать открыть файл и найти необходимый файл с кодом .py.

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

В интерактивном режиме выполняет команду сразу после ввода в командной строке.

В режиме интерпретации файлов с исходным кодом необходимо сначала написать код в файле.

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

Также языки бывают с динамической и статической типизацией. В первом случае тип переменной определяется непосредственно при выполнении программы, во втором – на этапе компиляции.

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

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?**

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

Целочисленное значение 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.

**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()*?**

*id()* выводит идентификатор переменной, а *type()* – тип переменной.

**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).

Неизменяемость типа данных означает, что созданный объект больше не изменяется.

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

Целочисленное деление возвращает только целую часть от деления, отбрасывая остаток.

### **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, можно извлечь действительную и мнимую части.

### **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.

Функции:

`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  $e$ .

#### **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. По умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку

#### **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 строке в фигурных скобках указаны номера данных, которые будут сюда 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.

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода `format`. Старый стиль также называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке C.

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`. Сами значения записываются в скобках после знака процента(`%`)

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

При помощи оборачивания `input()` в `int()` (`int(input())`) для целочисленной и в `float()` (`float(input())`) для вещественной.