

**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 ПРАКТИЧЕСКОЙ РАБОТЕ №2.1
дисциплины
«Основы кроссплатформенного программирования»
Исследование
основных возможностей Git и GitHub**

Выполнила студентка группы
ИВТ-б-о-21-1
Яковлева Е.А. « » _____ 20__ г.
Подпись студента _____
Работа защищена
« » _____ 20__ г.

Проверил доцент
Кафедры инфокоммуникаций,
старший преподаватель
Воронкин Р.А.

(подпись)

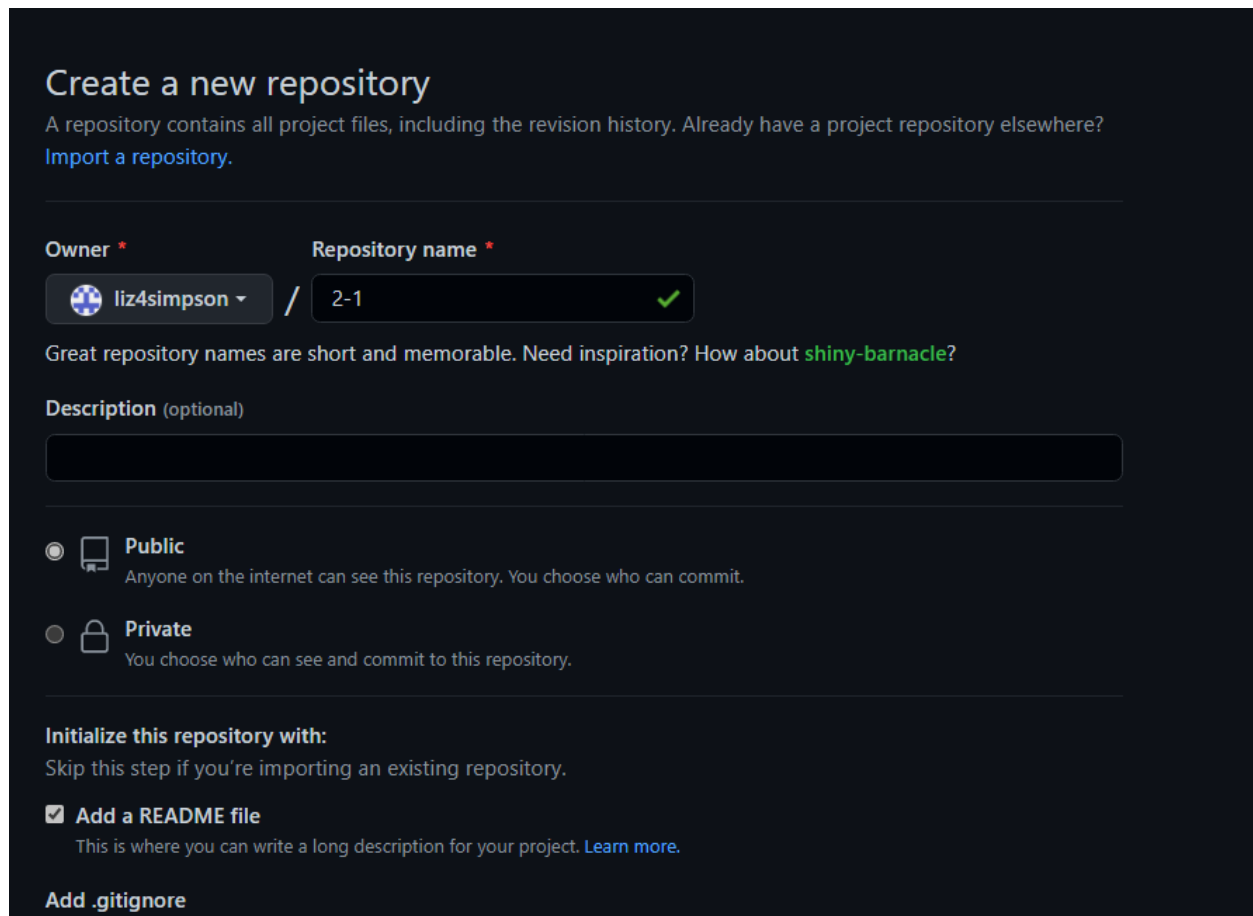
Ставрополь, 2022 г.

Тема: условные операторы и циклы в языке Python

Цель: исследование процесса установки и базовых возможностей языка Python

Порядок выполнения работы:

Создала репозиторий GitHub с лицензией MIT, добавил .gitignore с ЯП python, клонировала репозиторий на ПК и организовала репозиторий согласно модели ветвления git-flow:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * liz4simpson / Repository name * 2-1 ✓

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

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:

Skip this step if you're importing an existing repository.

☒ Add a README file This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore

Рисунок 1 – Создание нового репозитория

```
c:\Users\Elizaveta\Desktop\git>git clone https://github.com/liz4simpson/2.1.git
Cloning into '2.1'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 4.48 KiB | 76.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

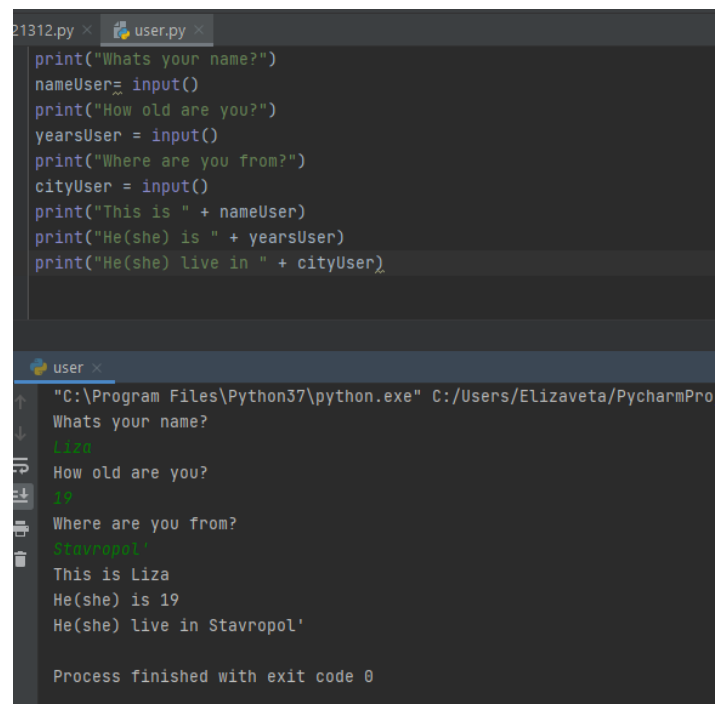
Рисунок 2 – Клонировала репозиторий на свой компьютер

```

c:\Users\Elizaveta\Desktop\git\2-1> git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]
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? [C:/Users/Elizaveta/Desktop/git/2-1/.git/hooks]
c:\Users\Elizaveta\Desktop\git\2-1>

```

Рисунок 3 – Организация репозитория согласно модели git-flow



```

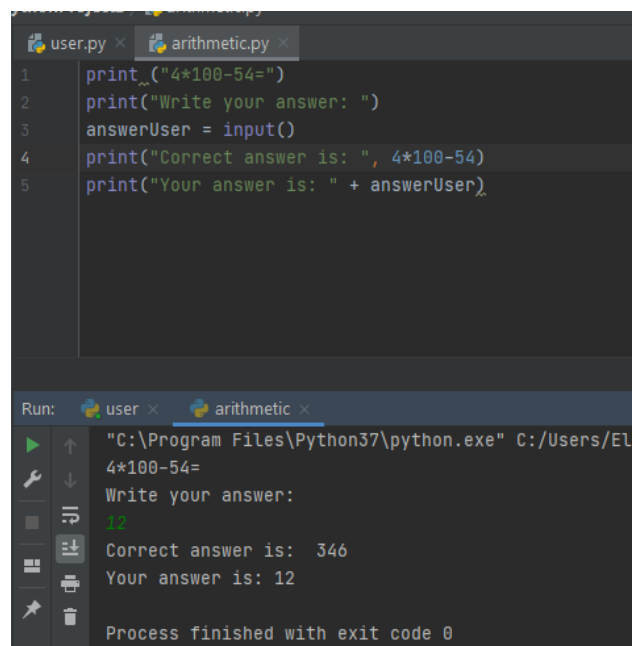
21312.py x user.py x
print("Whats your name?")
nameUser= input()
print("How old are you?")
yearsUser = input()
print("Where are you from?")
cityUser = input()
print("This is " + nameUser)
print("He(she) is " + yearsUser)
print("He(she) live in " + cityUser)

user x
"C:\Program Files\Python37\python.exe" C:/Users/Elizaveta/PycharmPro
Whats your name?
Liza
How old are you?
19
Where are you from?
Stavropol'
This is Liza
He(she) is 19
He(she) live in Stavropol'

Process finished with exit code 0

```

Рисунок 4 – Задача 1



```

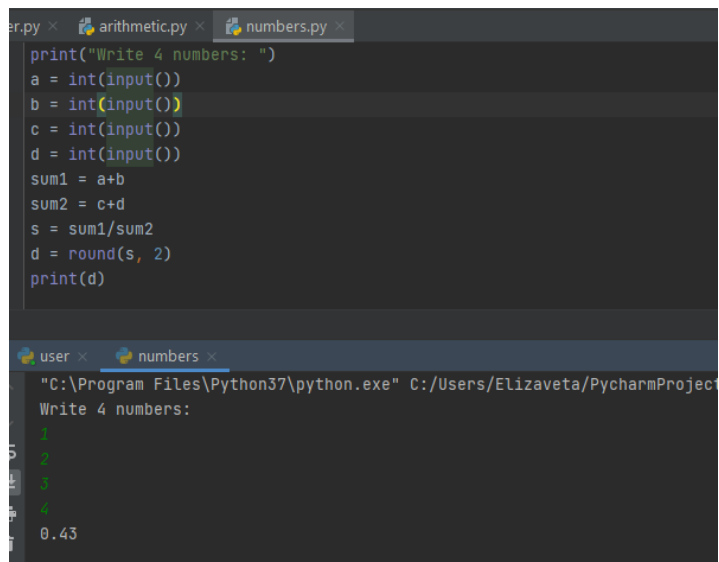
user.py x arithmetic.py x
1 print("4*100-54=")
2 print("Write your answer: ")
3 answerUser = input()
4 print("Correct answer is: ", 4*100-54)
5 print("Your answer is: " + answerUser)

Run: user x arithmetic x
"C:\Program Files\Python37\python.exe" C:/Users/EL
4*100-54=
Write your answer:
12
Correct answer is: 346
Your answer is: 12

Process finished with exit code 0

```

Рисунок 5 – Задача 2

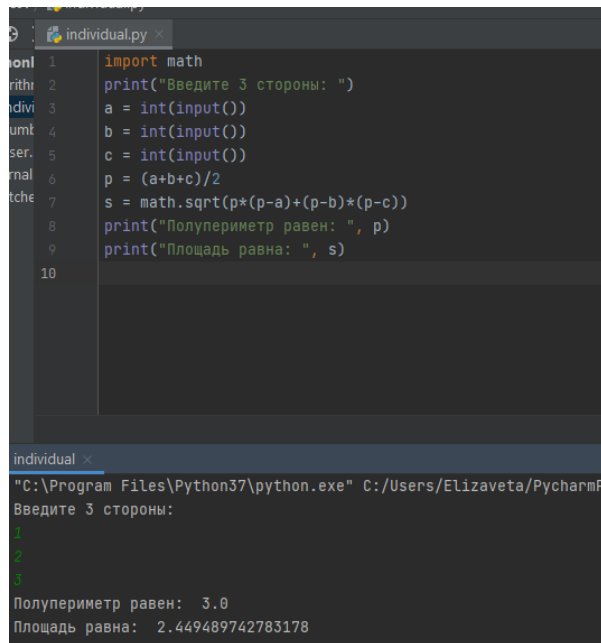


The screenshot shows a Python IDE with two tabs: 'arithmetic.py' and 'numbers.py'. The 'numbers.py' tab is active, displaying the following code:

```
print("Write 4 numbers: ")
a = int(input())
b = int(input())
c = int(input())
d = int(input())
sum1 = a+b
sum2 = c+d
s = sum1/sum2
d = round(s, 2)
print(d)
```

Below the code editor, the terminal window shows the execution of the script. It prompts the user to "Write 4 numbers:" and receives four inputs: 1, 2, 3, and 4. The output of the script is 0.43.

Рисунок 6 – Задача 3



The screenshot shows a Python IDE with a tab for 'individual.py'. The code in the editor is as follows:

```
1 import math
2 print("Введите 3 стороны: ")
3 a = int(input())
4 b = int(input())
5 c = int(input())
6 p = (a+b+c)/2
7 s = math.sqrt(p*(p-a)*(p-b)*(p-c))
8 print("Полупериметр равен: ", p)
9 print("Площадь равна: ", s)
10
```

The terminal window below shows the execution of the script. It prompts the user to "Введите 3 стороны:" and receives three inputs: 1, 2, and 3. The output of the script is:

```
Полупериметр равен: 3.0
Площадь равна: 2.449489742783178
```

Рисунок 7 – Индивидуальное задание (В-27)

```
individual.py x slojno.py x
a3 = int(input("Введите a3: "))
a2 = int(input("Введите a2: "))
a1 = int(input("Введите a1: "))
b3 = int(input("Введите b3: "))
b2 = int(input("Введите b2: "))
b1 = int(input("Введите b1: "))

c3 = a3 + b3 + ((a2 + b2 + (a1 + b1))//10)//10
c2 = (a2 + b2 + (a1 + b1) // 10) % 10
c1 = (a1 + b1) % 10

print(c3, c2, c1)
|

Введите a3: 1
Введите a2: 2
Введите a1: 3
Введите b3: 4
Введите b2: 5
Введите b1: 6
Process finished with exit code 0
```

Рисунок 8 – Задача повышенной сложности (В-27)

```
c:\Users\Elizaveta\Desktop\git\lr2.1>
c:\Users\Elizaveta\Desktop\git\lr2.1>git branch
* develop
  main

c:\Users\Elizaveta\Desktop\git\lr2.1>
c:\Users\Elizaveta\Desktop\git\lr2.1>git add .
warning: LF will be replaced by CRLF in .idea/inspectionProfiles/profiles_settings.xml.
The file will have its original line endings in your working directory

c:\Users\Elizaveta\Desktop\git\lr2.1>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .idea/.gitignore
    new file:   .idea/inspectionProfiles/profiles_settings.xml
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/pythonProject4.iml
    new file:   arithmetic.py
    new file:   individual.py
    new file:   numbers.py
    new file:   slojno.py
    new file:   user.py

c:\Users\Elizaveta\Desktop\git\lr2.1>git commit -m "Added programs"
[develop 6fa583e] Added programs
10 files changed, 74 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/pythonProject4.iml
create mode 100644 arithmetic.py
create mode 100644 individual.py
create mode 100644 numbers.py
create mode 100644 slojno.py
create mode 100644 user.py
```

Рисунок 9 – Коммит изменений в ветку develop

```
c:\Users\Elizaveta\Desktop\git\lr2.1>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

c:\Users\Elizaveta\Desktop\git\lr2.1>git merge develop
Updating a79583b..6fa583e
Fast-forward
 .idea/.gitignore | 3 +++
 .idea/inspectionProfiles/profiles_settings.xml | 6 ++++++
 .idea/misc.xml | 4 +++++
 .idea/modules.xml | 8 ++++++++
 .idea/pythonProject4.iml | 8 ++++++++
 arithmetic.py | 5 +++++
 individual.py | 9 ++++++++
 numbers.py | 10 ++++++++
 slojno.py | 12 ++++++++
 user.py | 9 ++++++++
10 files changed, 74 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/pythonProject4.iml
create mode 100644 arithmetic.py
create mode 100644 individual.py
create mode 100644 numbers.py
create mode 100644 slojno.py
create mode 100644 user.py
```

Рисунок 10 – Слияние ветки develop с веткой main

```
c:\Users\Elizaveta\Desktop\git\lr2.1>git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (14/14), 2.10 KiB | 358.00 KiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/liz4simpson/lr2.1.git
 a79583b..6fa583e main -> main
```

Рисунок 11 – Отправка изменений на удалённый репозиторий

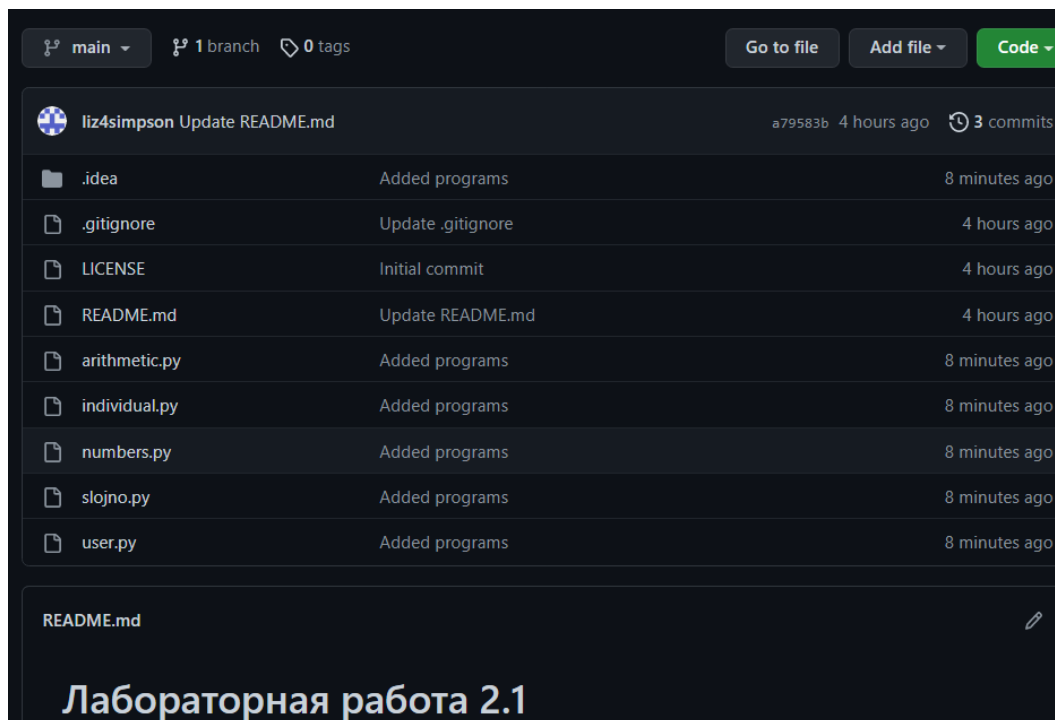


Рисунок 12 – Зафиксированные изменения на удалённом репозитории

Ответы на контрольные вопросы:

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

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

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 с выбором 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. Пакетный. 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 π .

`math.e` - число e .

16. Каково назначение именованных 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. 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())`.

Вывод: исследовала процесс установки и базовые возможности языка программирования Python версии 3.