

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение  
высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития**

**Кафедра информационных систем и технологий**

Отчет по лабораторной работе №4.

Дисциплина: «Основы программной инженерии»

**Выполнил:**

Студент группы ПИЖ-б-о-22-1,  
направление подготовки: 09.03.04

«Программная инженерия» ФИО:

Джараян Арег Александрович

**Проверил:**

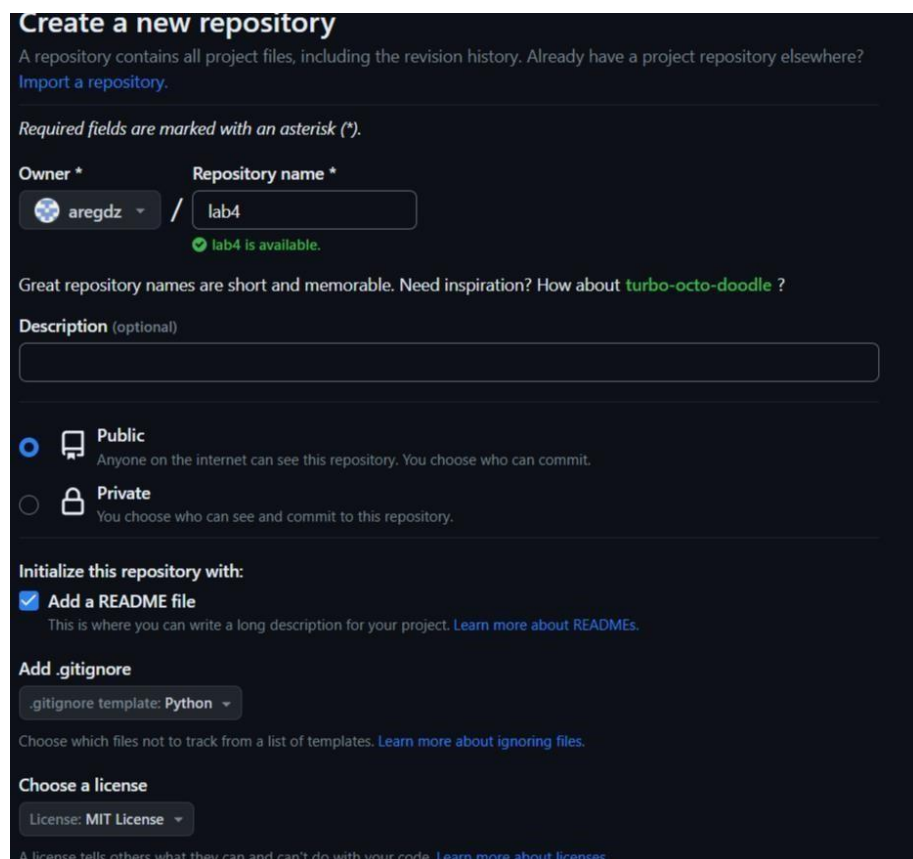
Воронкин Р. А.

Тема: Лабораторная работа 2.1 Основы языка Python

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

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.



The screenshot shows the 'Create a new repository' page on GitHub. The form includes the following fields and options:

- Owner \***: A dropdown menu showing 'aregdz'.
- Repository name \***: A text input field containing 'lab4'. Below it, a green checkmark indicates 'lab4 is available'.
- Description (optional)**: A large text area for entering a description.
- Visibility**: Two radio buttons. 'Public' is selected, with the description 'Anyone on the internet can see this repository. You choose who can commit.' 'Private' is unselected, with the description 'You choose who can see and commit to this repository.'
- Initialize this repository with:**
  - ☒ **Add a README file**: This is where you can write a long description for your project. [Learn more about READMEs.](#)
  - Add .gitignore**: A dropdown menu showing '.gitignore template: Python'.
  - Choose a license**: A dropdown menu showing 'License: MIT License'.

At the bottom, there is a link: [A license tells others what they can and can't do with your code. Learn more about licenses.](#)

Рисунок 1 – создание репозитория

3. Клонировал репозиторий.

```
MINGW64:/c/Users/aregd/OneDrive/Рабочий стол/git4

aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git4"

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4
$ git clone https://github.com/aregdz/lab4.git
Cloning into 'lab4'...
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.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4
$ |
```

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.

```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 - – .gitignore для IDE PyCharm

5. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4
$ cd lab4

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4/lab4 (main)
$ git branch develop

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4/lab4 (main)
$ ^[[200~@DESKTOP-5KV9QA9 MINGW64 ~/
bash: $'\E[200~@DESKTOP-5KV9QA9': command not found

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4/lab4 (main)
$ git push -u 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/aregdz/lab4/pull/new/develop
remote:
To https://github.com/aregdz/lab4.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

```

Рисунок 4 – создание ветки develop

6. Создание и переход на ветку feature\_branch.

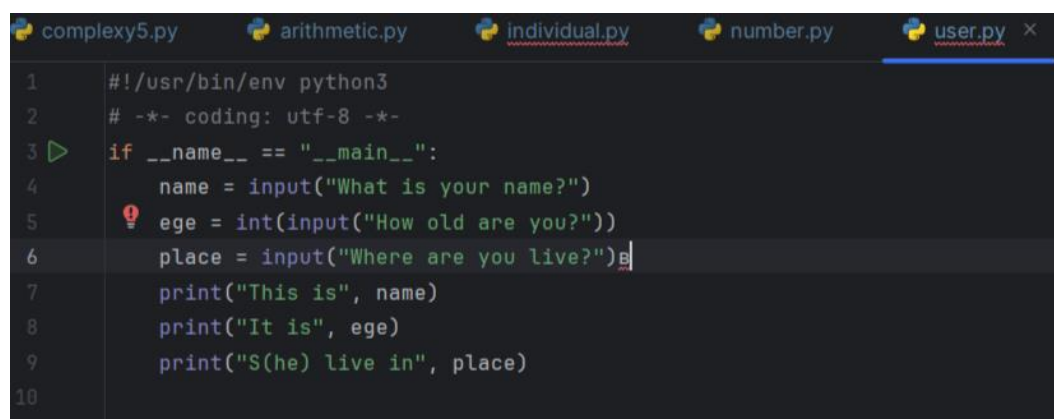
```

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git4/lab4 (main)
$ git checkout -b feature_branch
Switched to a new branch 'feature_branch'

```

Рисунок 5 - создание и переход на ветку feature\_branch.

7. Написать 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.



```

complexy5.py  arithmetic.py  individual.py  number.py  user.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      name = input("What is your name?")
5      ege = int(input("How old are you?"))
6      place = input("Where are you live?")
7      print("This is", name)
8      print("It is", ege)
9      print("S(he) live in", place)
10

```

Рисунок 6 — файл user.py

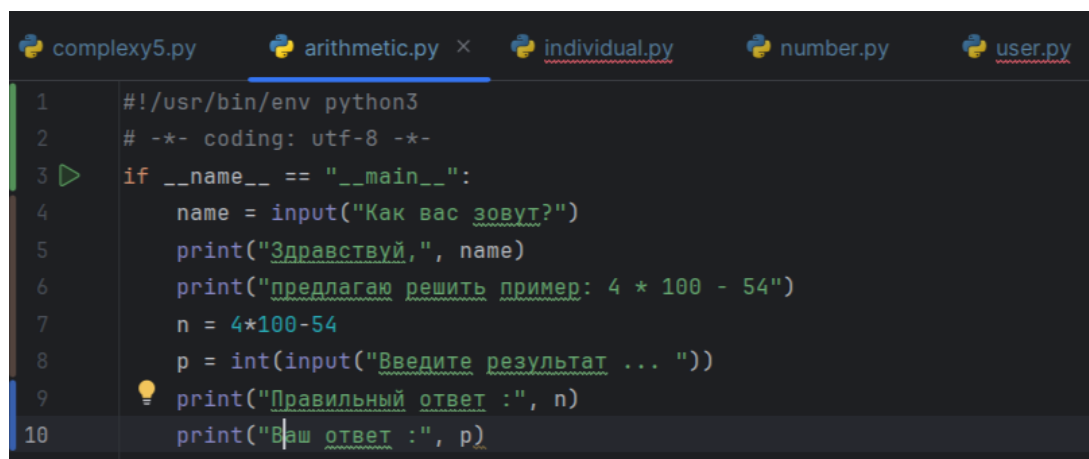
```

What is your name? areg
How old are you? 23
Where are you live?Greece
This is areg
It is 23
S(he) live in Greece

```

Рисунок 7 – Результат выполнения файла user.py

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



```

complexy5.py  arithmetic.py ×  individual.py  number.py  user.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      name = input("Как вас зовут?")
5      print("Здравствуй,", name)
6      print("предлагаю решить пример: 4 * 100 - 54")
7      n = 4*100-54
8      p = int(input("Введите результат ... "))
9      print("Правильный ответ :", n)
10     print("Ваш ответ :", p)

```

Рисунок 8 — Файл arithmetic.py

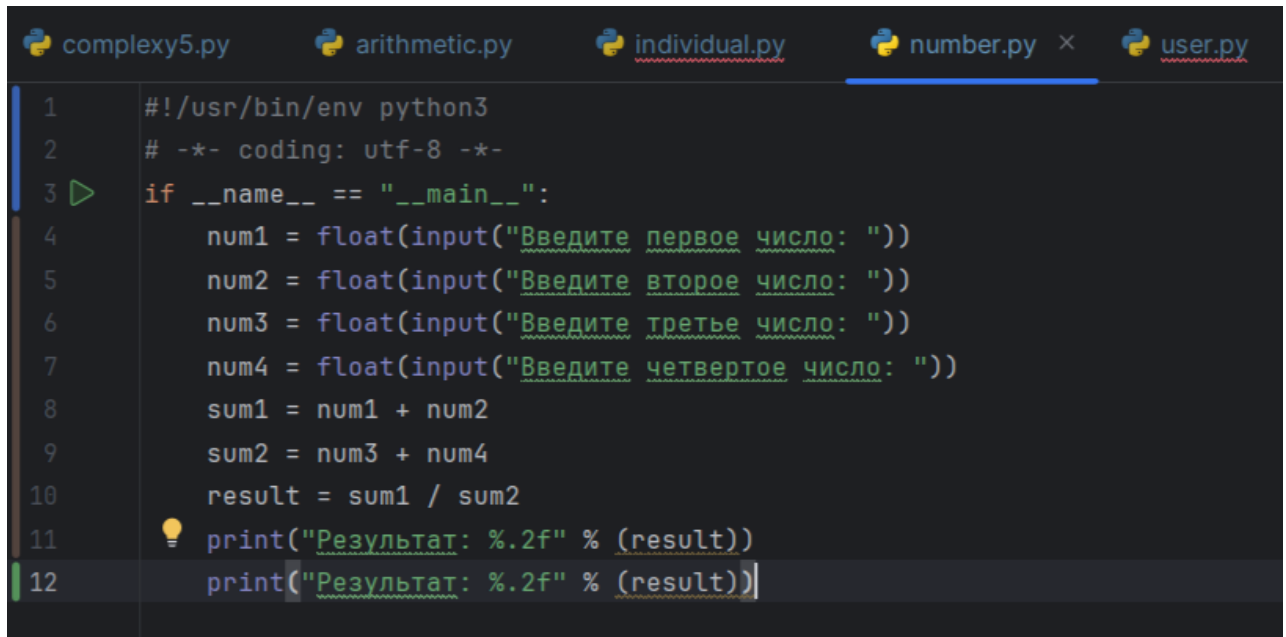
```

Как вас зовут?areg
Здравствуй, areg
предлагаю решить пример: 4 * 100 - 54
Введите результат ... 67
Не верный ответ, правильный ответ - 346

```

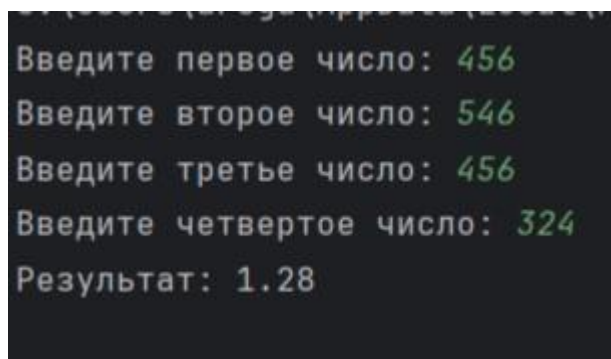
Рисунок 9 – результат выполнения файла arithmetic.py

9. Запросить у пользователя четыре числа (файл numbers.py). Отдельно сложить первые два отдельно вторые два. Разделить первую сумму на вторую. Вывести результат на экран так, чтобы ответ содержал две цифры после запятой.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 if __name__ == '__main__':
4     num1 = float(input("Введите первое число: "))
5     num2 = float(input("Введите второе число: "))
6     num3 = float(input("Введите третье число: "))
7     num4 = float(input("Введите четвертое число: "))
8     sum1 = num1 + num2
9     sum2 = num3 + num4
10    result = sum1 / sum2
11    print("Результат: %.2f" % (result))
12    print("Результат: %.2f" % (result))
```

Рисунок 10 — файл numbers.py



```
Введите первое число: 456
Введите второе число: 546
Введите третье число: 456
Введите четвертое число: 324
Результат: 1.28
```

Рисунок 11 — Результат выполнения файла number.py

### Индивидуальное задание. Вариант 5.

5. Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      a = float(input("Введите длину параллелепипеда: "))
5      b = float(input("Введите ширину параллелепипеда: "))
6      c = float(input("Введите высоту параллелепипеда: "))
7      v = a * b * c
8      s = 2 * (a * b + b * c + a * c)
9      print("Объем параллелепипеда:", v)
10     print("Площадь боковой поверхности:", s)

```

Рисунок 7 – файл individual.py

```

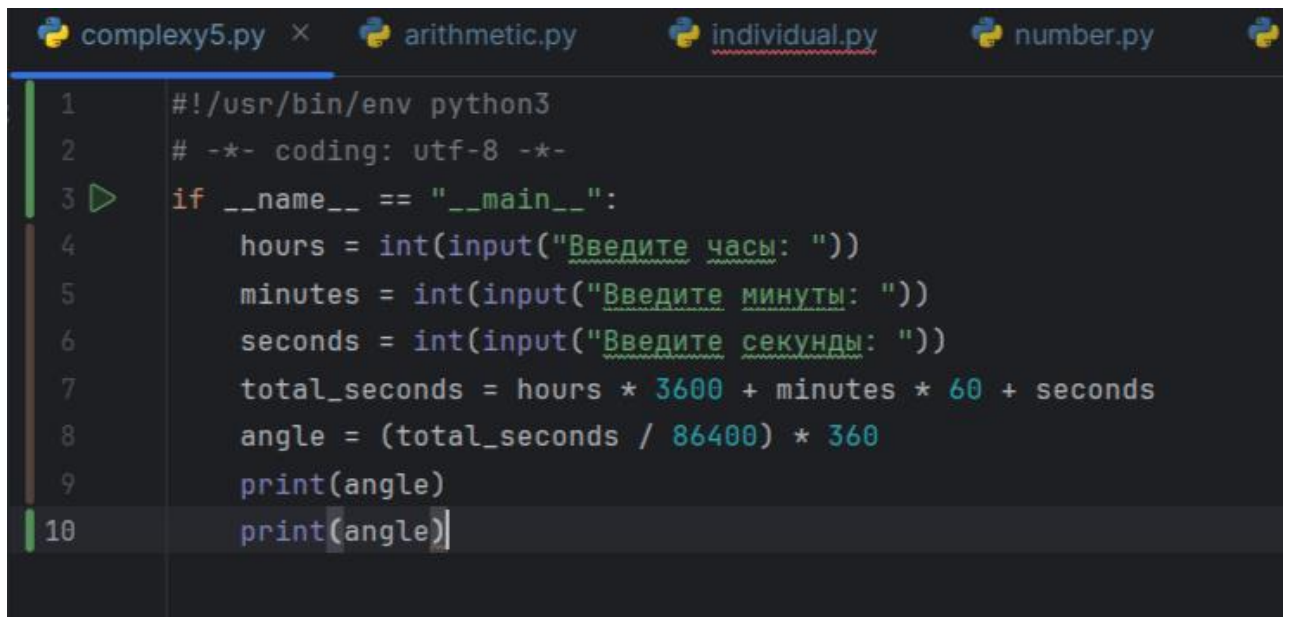
Введите длину параллелепипеда: 2
Введите ширину параллелепипеда: 3
Введите высоту параллелепипеда: 4
Объем параллелепипеда: 24.0
Площадь боковой поверхности: 52.0

```

Рисунок 8 – результат выполнения файла individual.py

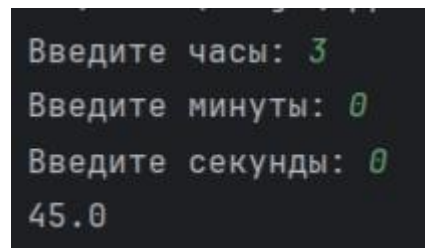
### Задачи повышенной сложности

5. Даны целые числа, указывающие момент времени: « часов, минут, секунд». Определить угол (в градусах) между положением часовой стрелки в начале суток и в указанный момент времени.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      hours = int(input("Введите часы: "))
5      minutes = int(input("Введите минуты: "))
6      seconds = int(input("Введите секунды: "))
7      total_seconds = hours * 3600 + minutes * 60 + seconds
8      angle = (total_seconds / 86400) * 360
9      print(angle)
10     print(angle)]
```

Рисунок 9 – файл complex5.py



```
Введите часы: 3
Введите минуты: 0
Введите секунды: 0
45.0
```

Рисунок 10 – результат выполнения файла complex5.py

Контрольные вопросы:

1. Опишите основные этапы установки Python в Windows и Linux

Установка Python в Windows

Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла (с расширением exe), либо в виде архивного файла (с расширением zip). Если вы используете Windows 7, не забудьте установить Service Pack 1.

Порядок установки.

1. Запустите скачанный установочный файл.
2. Выберите способ установки



3. Отметьте необходимые опции установки (доступно при выборе Customize installation)
4. Выберите место установки (доступно при выборе Customize installation)
5. После успешной установки вас ждет сообщение об успешной установке

### Установка Python в Linux

Чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить, набрав в терминале `python` или `python 3`. В первом случае вы запустите Python 2 во втором – Python 3. Если у вас, при попытке запустить Python, выдается сообщение о том, что он не установлен, или установлен, но не тот, что вы хотите, то у вас есть два пути: а) собрать Python из исходников; б) взять из репозитория. Для установки из репозитория в Ubuntu воспользуйтесь командой `sudo apt-get install python3`.

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

Anaconda и Python, скачиваемый с официального сайта, представляют собой два разных подхода к установке и управлению Python и его пакетами. Вот основные отличия между ними:

#### -Дистрибуция Python:

Python, скачиваемый с официального сайта (называемый CPython), представляет собой официальный интерпретатор Python, разрабатываемый Python Software Foundation.

Anaconda - это дистрибуция Python, предназначенная для научных вычислений и анализа данных. Она включает в себя не только Python, но и множество научных библиотек и инструментов.

-Библиотеки и пакеты:

Anaconda поставляется с предустановленным набором научных и аналитических библиотек, таких как NumPy, SciPy, pandas, Matplotlib и Jupyter Notebook. Это делает Anaconda идеальным инструментом для научных исследований и анализа данных.

CPython с официального сайта включает только базовый интерпретатор Python. Дополнительные библиотеки и инструменты должны быть установлены вручную с использованием инструментов управления пакетами, таких как pip.

-Управление пакетами:

Anaconda поставляется с собственным инструментом управления пакетами под названием conda. conda позволяет легко устанавливать, обновлять и удалять пакеты, а также создавать изолированные среды для проектов.

CPython с официального сайта использует pip для управления пакетами. pip также мощный инструмент, но conda обеспечивает большую надежность и совместимость, особенно в контексте научных библиотек. 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 Jupyter Notebook после чего запустится веб-сервер и среда разработки в браузере.

3. Как задать используемый интерпретатор языка Python в IDE PyCharm?

Запустите PyCharm и выберите Create New Project в появившемся окне. Укажите путь до проекта Python и интерпретатор, который будет использоваться для запуска и отладки..

4. Как осуществить запуск программы с помощью IDE PyCharm?

После того, как исходный код написан, чтобы первый раз запустить программу, проще всего нажать Ctrl+Shift+F10.

5. В чем суть интерактивного и пакетного режимов работы Python?

В интерактивный режим можно войти, набрав в командной строке python или python3. В результате Python запустится в интерактивном режиме и будет ожидать ввод команд пользователя.

Чтобы выполнить запуск в пакетном режиме, надо ввести в командной строке имя интерпретатора, плюс имя файла.

6. Почему язык программирования Python называется языком динамической типизации?

Если достаточно формально подходить к вопросу о типизации языка Python, то можно сказать, что он относится к языкам с неявной сильной динамической типизацией. Неявная типизация означает, что при объявлении переменной вам не нужно указывать её тип, при явной – это делать необходимо. Также языки бывают с динамической и статической типизацией. В первом случае тип переменной определяется непосредственно при выполнении программы, во втором – на этапе компиляции. Как уже было сказано Python – это динамически типизированный язык.

7. Какие существуют основные типы в языке программирования Python?

К основным встроенным типам относятся:

- None (неопределенное значение переменной);
- Логические переменные (Boolean Type);
- Числа (Numeric Type);
- Списки (Sequence Type);
- Строки (Text Sequence Type);
- Бинарные списки (Binary Sequence Types);
- Множества (Set Types);
- Словари (Mapping Types).

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

Объект – 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. 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 (например, 5), 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`).

9. Как получить список ключевых слов в Python?

Проверить является или нет идентификатор ключевым словом можно с помощью команды `keyword.iskeyword("название_переменной")`.

10. 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()`.

11. Что такое изменяемые и неизменяемые типы в Python  
В Python существуют изменяемые и неизменяемые типы.

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. Если тип данных изменяемый, то можно менять значение объекта.

12. Чем отличаются операции деления и целочисленного деления?

Целочисленное деление (`//`) отличается от обычной операции деления тем, что возвращает целую часть частного, а дробная часть отбрасывается.

13. 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 комплексно сопряженного числа необходимо использовать метод `conjugate()`.

14. 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. Основные функции `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  $\pi$
- `math.e` число  $e$

Модуль `math` предоставляет доступ к математическим функциям для комплексных чисел. Функции в этом модуле принимают в качестве аргументов целые числа, числа с плавающей запятой или комплексные числа. Они также будут принимать любой объект Python, у которого есть метод `__complex__()` или `__float__()`: данные методы используются для преобразования объекта в комплексное число или число с плавающей запятой соответственно, а затем функция применяется к результату преобразования.

Множество функций подобные, но не идентичные тому, что в модуле `math`. Причина наличия двух модулей в том, что некоторые пользователи не интересуются комплексными числами и, возможно, даже не знают, что они собой представляют. Функции, определённые в `cmath`, всегда возвращают комплексное число, даже если ответ может быть выражен в виде действительного числа (в этом случае комплексное число имеет нулевую мнимую часть). Основные функции `cmath`:

- `cmath.polar(x)` возвращает представление  $x$  в полярных координатах.
- `cmath.rect(r, phi)` возвращает комплексное число  $x$  с полярными координатами  $r$  и  $phi$
- `cmath.phase(x)` возвращает фазу  $x$  (также известную как `argument`)

$x$ ) в виде числа с плавающей запятой

- `cmath.isfinite(x)` возвращает `True`, если действительная, и мнимая части  $x$  конечны, и `False` в противном случае
- `cmath.isinf(x)` возвращает `True`, если действительная или мнимая часть  $x$  равна бесконечности, и `False` в противном случае
- `cmath.isnan(x)` возвращает `True`, если действительная или мнимая часть  $x$  является NaN, и `False` в противном случае
- `cmath.infj` комплексное число с нулевой действительной частью и положительной бесконечностью мнимой части
- `cmath.nanj` комплексное число с нулевой действительной частью и NaN мнимой частью.
- `math.sqrt(x)` корень квадратный от  $x$
- `math.nan` Значение с плавающей запятой «не число» (NaN)

15. 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.

16. 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.



В строке в фигурных скобках указаны номера данных, которые будут 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. Существует также Старый стиль его называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке С. Вместо трех комбинаций символов `%s` , `%d` , `%f` подставляются значения переменных. 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`.

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

За ввод в программу данных с клавиатуры в Python отвечает функция `input()`. Когда вызывается эта функция, программа останавливает свое выполнение и ждет, когда пользователь введет текст. Функция `input()` передает введенные данные в программу. Их можно присвоить переменной. Чтобы получить целочисленное и вещественное значение нужно воспользоваться функцией преобразования типов. В данном случае с помощью функций `int()` и `float()`.