

**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

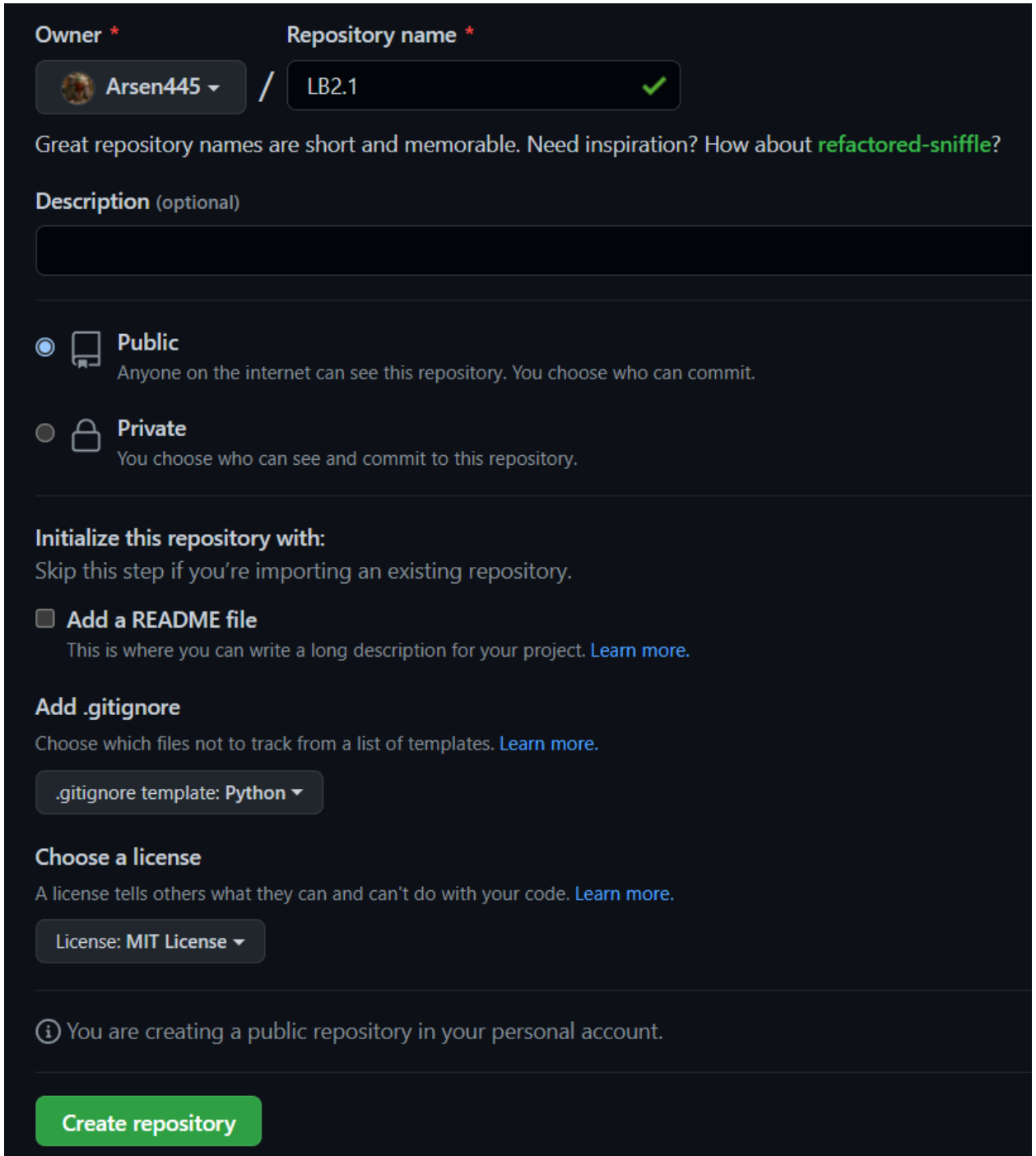
Эсеналиев Арсен Мурадинович

Ставрополь 2022

Ссылка на репозиторий: <https://github.com/Arsen445/LB2.1>

Выполнение работы:

1. Создал репозиторий GitHub с лицензией MIT, добавил .gitignore с ЯП python, клонировал репозиторий на ПК и организовал репозиторий согласно модели ветвления git-flow:



The screenshot shows the GitHub repository creation page. At the top, the 'Owner' is set to 'Arsen445' and the 'Repository name' is 'LB2.1', which is marked as valid with a green checkmark. Below this, a message suggests great repository names are short and memorable, with a link to 'refactored-sniffle?'. The 'Description' field is empty. Under the 'Visibility' section, 'Public' is selected, indicating that anyone on the internet can see the repository. The 'Private' option is also visible. The 'Initialize this repository with:' section includes a checkbox for 'Add a README file'. The 'Add .gitignore' section shows a dropdown menu with 'Python' selected. The 'Choose a license' section has a dropdown menu with 'MIT License' selected. At the bottom, a green button labeled 'Create repository' is visible.

Owner * Repository name *

Arsen445 / LB2.1 ✓

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

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
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

i You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория

```
C:\>git clone https://github.com/Arsen445/LB2.1.git
Cloning into 'LB2.1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

```
C:\LB2.1>git flow init

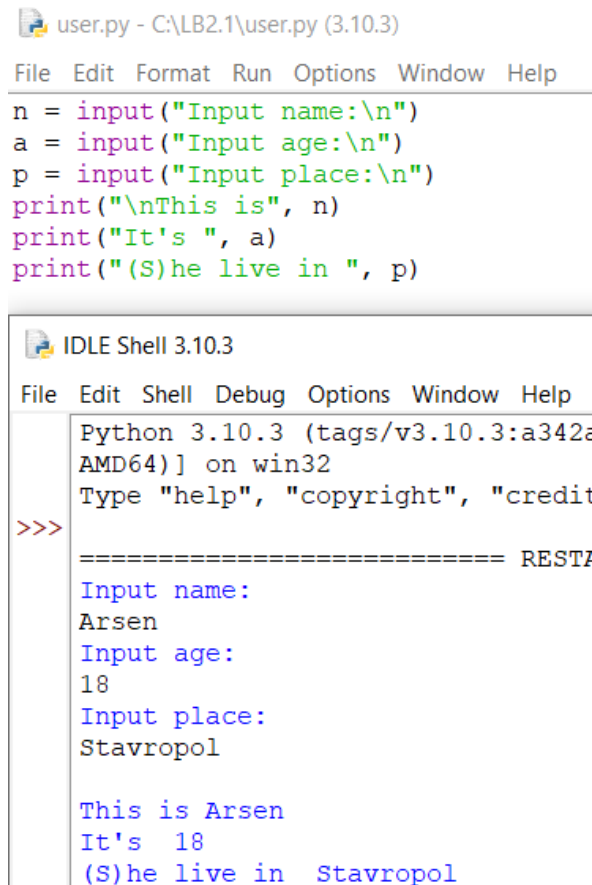
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

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:/LB2.1/.git/hooks]
```

Рисунок 1.2 Клонирование и организация репозитория согласно модели ветвления git-flow

2. Написал программу user.py, которая запрашивает у пользователя имя, возраст и место жительства, после этого выводит бы 3 строки

```
"This is `имя`"
"It is `возраст`"
"(S)he live in `место_жительства`"
```



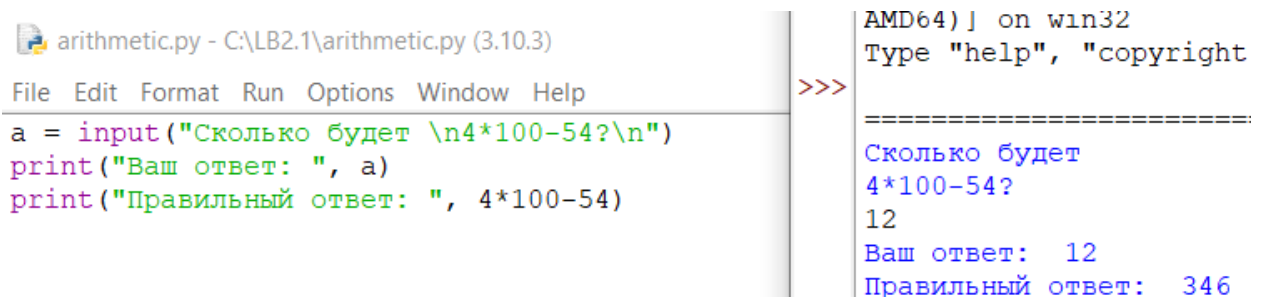
```
user.py - C:\LB2.1\user.py (3.10.3)
File Edit Format Run Options Window Help
n = input("Input name:\n")
a = input("Input age:\n")
p = input("Input place:\n")
print("\nThis is", n)
print("It's ", a)
print("(S)he live in ", p)

IDLE Shell 3.10.3
File Edit Shell Debug Options Window Help
Python 3.10.3 (tags/v3.10.3:a342a) on win32
Type "help", "copyright", "credit" for more
>>>
===== RESTART: >>>
Input name:
Arsen
Input age:
18
Input place:
Stavropol

This is Arsen
It's 18
(S)he live in Stavropol
```

Рисунок 1.1 Программа user

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

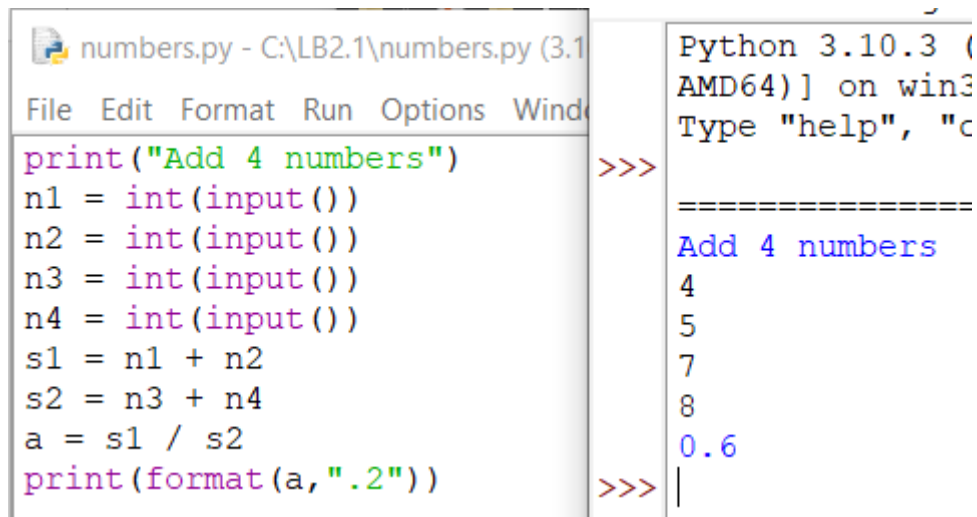


```
arithmetic.py - C:\LB2.1\arithmetic.py (3.10.3)
File Edit Format Run Options Window Help
a = input("Сколько будет 4*100-54?\n")
print("Ваш ответ: ", a)
print("Правильный ответ: ", 4*100-54)

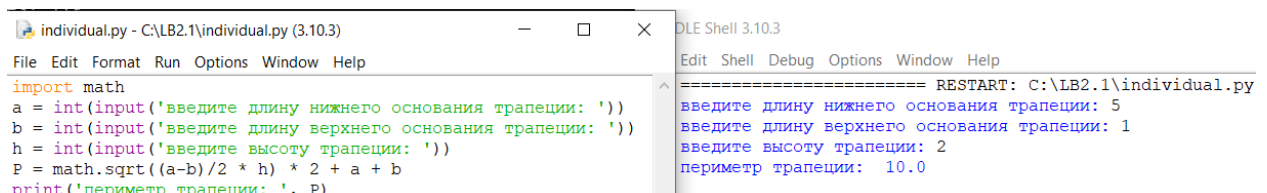
AMD64)] on win32
Type "help", "copyright", "credit" for more
>>>
=====
Сколько будет
4*100-54?
12
Ваш ответ: 12
Правильный ответ: 346
```

Рисунок 4.1 Программа arithmetic.py

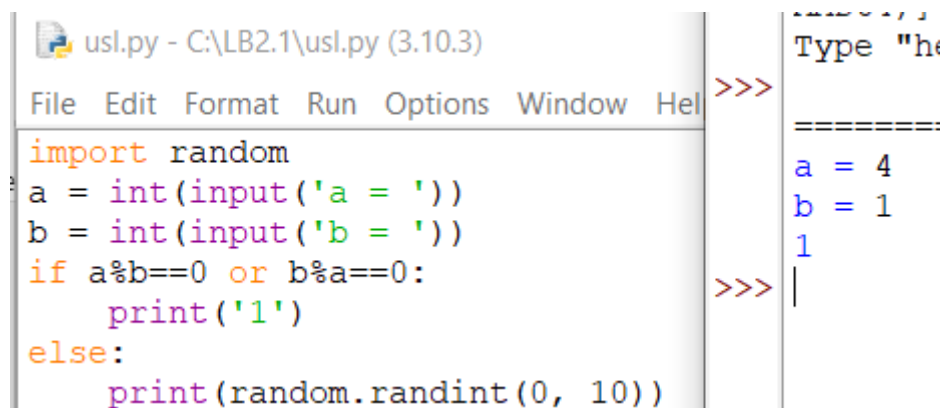
5. Написал программу numbers.py, которая запрашивает у пользователя 4 числа, отдельно складывает первые два и вторые два, затем делит первую сумму на вторую, после выводит рез-т на экран с точностью до сотен.



5. Написал программу для индивидуального задания (24 вариант):



6. Написал программу для усложненного задания:



7. Сделал коммит изменений в ветку разработки, выполнил ее слияние с веткой main и отправил сделанные изменения на уд. репозиторий.

```

C:\LB2.1>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   arithmetic.py
        new file:   individual.py
        new file:   numbers.py
        new file:   user.py

C:\LB2.1>git branch
* develop
  main

C:\LB2.1>git commit -m 'added_progs'
[develop df05335] 'added_progs'
 4 files changed, 24 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 numbers.py
 create mode 100644 user.py

```

Рисунок 7.1 Коммит изменений в ветку develop

```

Updating 424224d..4b6ef55
Fast-forward
 arithmetic.py | 3 +++
 individual.py | 6 ++++++
 numbers.py    | 11 ++++++++
 user.py       | 7 ++++++
 4 files changed, 27 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 numbers.py
 create mode 100644 user.py

```

Рисунок 7.2 Слияние ветки develop с веткой main

```

C:\LB2.1>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.17 KiB | 601.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Arsen445/LB2.1.git
   94a2cca..f042663  main -> main
branch 'main' set up to track 'origin/main'.

```

Рисунок 7.3 Push коммитов на уд. репозиторий

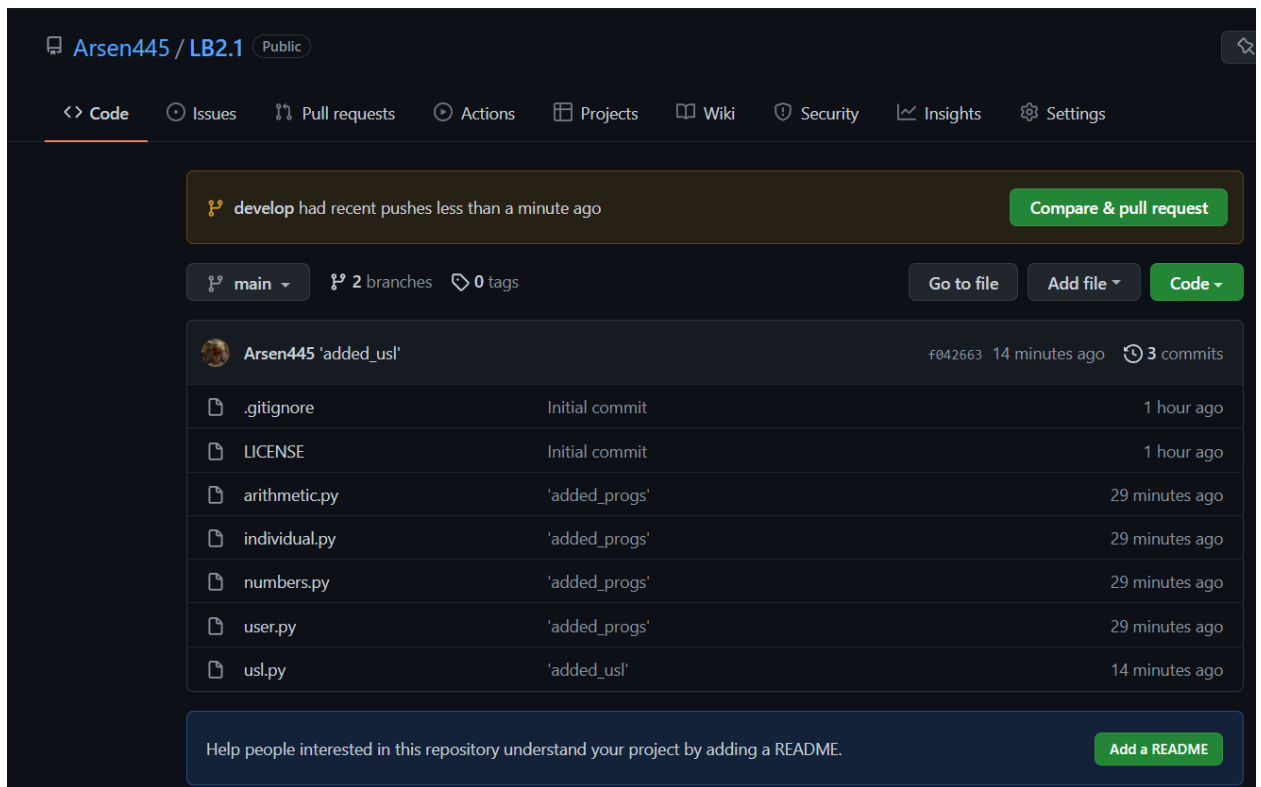


Рисунок 7.4 Изменения на уд. сервере

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

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

Windows: Оsn. 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 устновки;
- 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 число 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 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 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())`.