

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Основы языка Python»**

**Отчет по лабораторной работе № 2.1**

**по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Коновалова В.Н. « » 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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


**Ход работы:**

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

## Create a new repository

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

Owner \*

 VictoriaKonovalova ▾

Repository name \*

/ Lr4 ✓

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

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 ▾

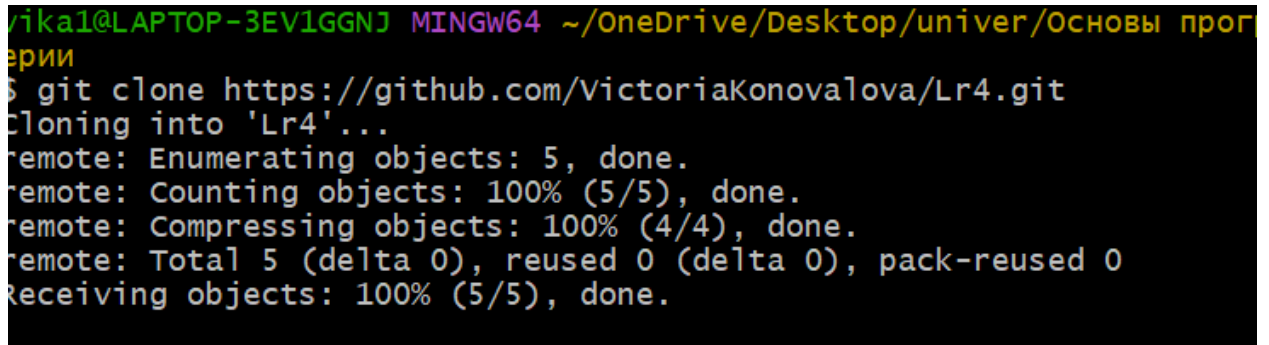
This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

**Create repository**

Рисунок 1 – Создание репозитория

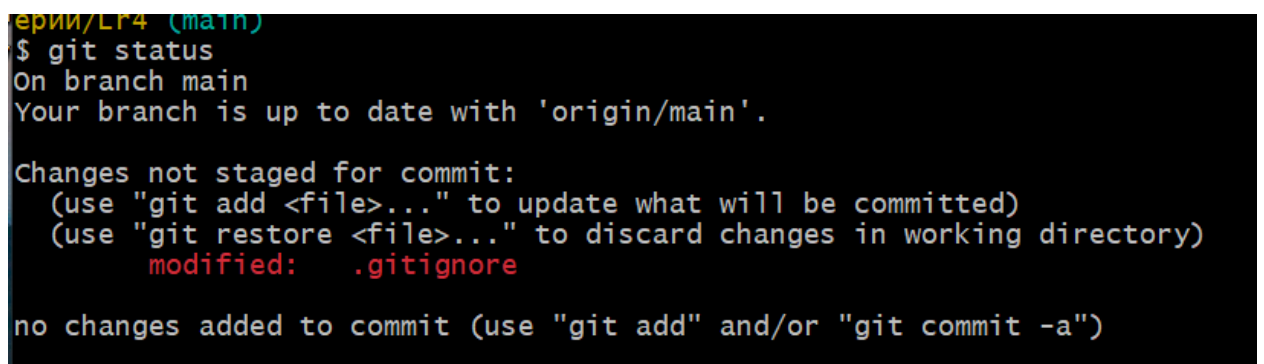
3. Выполните клонирование созданного репозитория.



```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/univer/Основы прог  
ерии  
$ git clone https://github.com/VictoriaKonovalova/Lr4.git  
Cloning into 'Lr4'...  
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.
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



```
ерии/Lr4 (main)  
$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
        modified:   .gitignore  
  
no changes added to commit (use "git add" and/or "git commit -a")
```

Рисунок 3 – Дополнение файла .gitignore

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

6. Создайте проект PyCharm в папке репозитория.

7. Решите следующие задачи с помощью языка программирования Python3 и IDE PyCharm:

8. Напишите 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:

"This is `имя`"

"It is `возраст`"

"(S)he lives in `место\_жительства`"

Вместо имя, возраст, место\_жительства должны быть данные, введенные пользователем.

Примечание: можно писать фразы на русском языке, но, если вы планируете стать профессиональным программистом, привыкайте к английскому.

Код программы:

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
home = input("Enter were you are live: ")
print("This is {0}, \n It is {1}, \n (S)he lives in {2} \n".format(name, age, home))
```

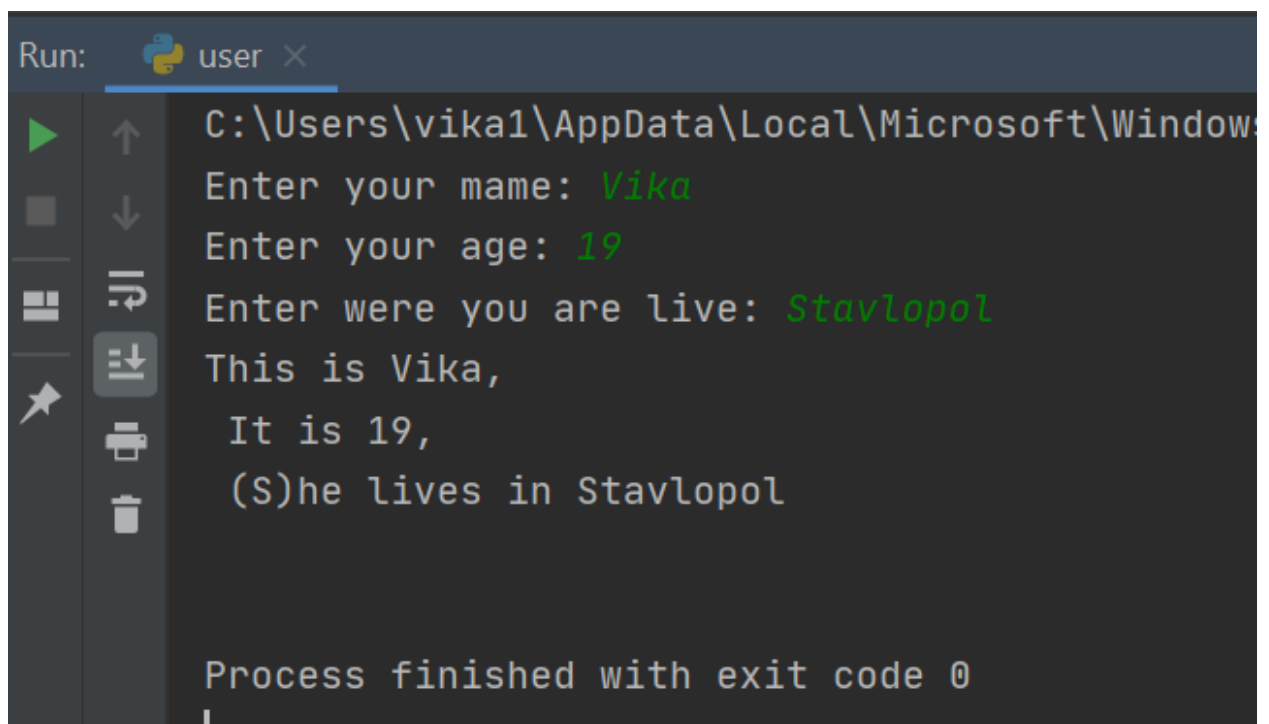


Рисунок 4 – Результат ее выполнения в IDE PyCharm

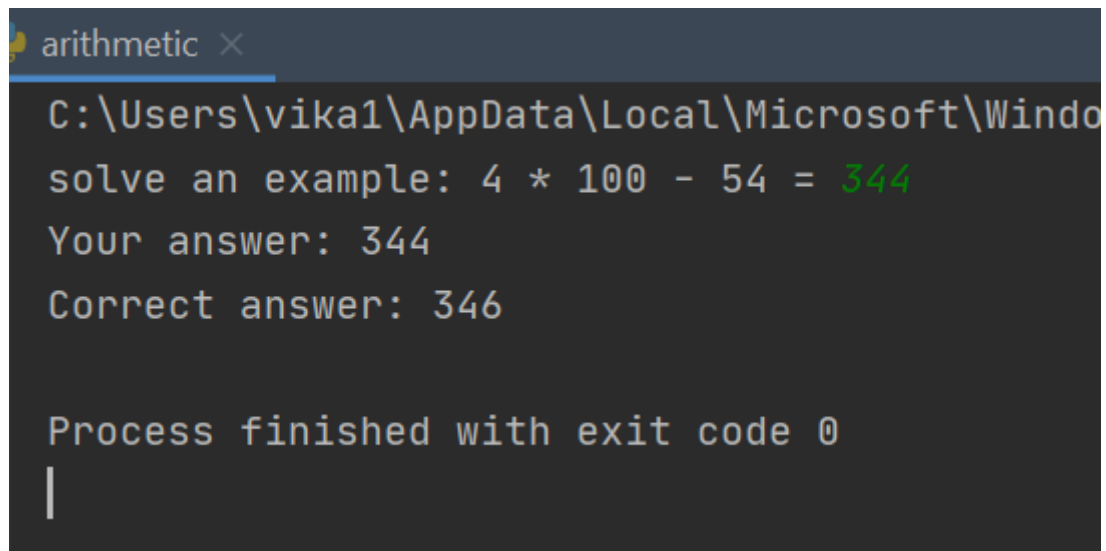
```
C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии\лаба 4\PyCharm>python user.py
Enter your name: Vika
Enter your age: 19
Enter were you are live: Stavropol
This is Vika,
It is 19,
(S)he lives in Stavropol
```

Рисунок 5 – Результат выполнение программы в командной строке

9. Напишите программу (файл arithmetic.py), которая предлагала бы пользователю решить пример  $4 * 100 - 54$ . Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число.

Код программы:

```
answer = int(input("solve an example: 4 * 100 - 54 = "))  
print("Your answer: {0} \nCorrect answer: {1}".format(answer, 4*100-54))
```



```
arithmetic x  
C:\Users\vika1\AppData\Local\Microsoft\Wind  
solve an example: 4 * 100 - 54 = 344  
Your answer: 344  
Correct answer: 346  
  
Process finished with exit code 0  
|
```

Рисунок 6 – Результат ее выполнения в IDE PyCharm

```
C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии\лаба 4\PyCharm>python arithmetic.py  
solve an example: 4 * 100 - 54 = 344  
Your answer: 344  
Correct answer: 346
```

Рисунок 7 – Результат выполнение программы в командной строке

10. Запросите у пользователя четыре числа (файл numbers.py). Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.

Код программы:

```

a = float(input("Enter 1 number: "))
b = float(input("Enter 2 number: "))
c = float(input("Enter 3 number: "))
d = float(input("Enter 4 number: "))
answer = (a + b)/(c + d)
print("answer= %.2f"% (answer))

```

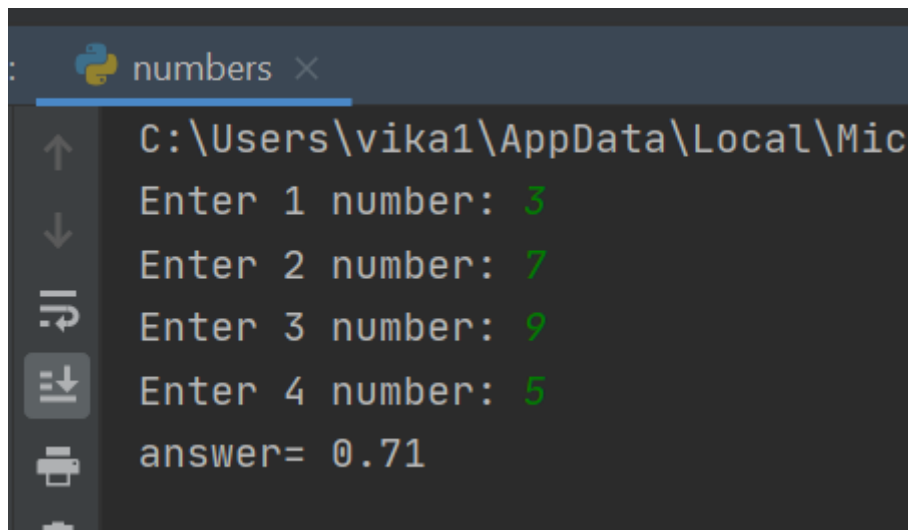


Рисунок 8 – Результат ее выполнения в IDE PyCharm

```

C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии\лаба 4\PyCharm>python numbers.py
Enter 1 number: 6
Enter 2 number: 4
Enter 3 number: 9
Enter 4 number: 24
answer= 0.30

```

Рисунок 9 – Результат выполнение программы в командной строке

11. Напишите программу (файл individual.py) для решения индивидуального задания.

13. Возраст Тани –  $X$  лет, а возраст Мити –  $Y$  лет. Найти их средний возраст, а также определить, на сколько отличается возраст каждого ребенка от среднего значения.

Код программы:

```

x = int(input())
y = int(input())
average = (x + y) / 2
T_diff = average - x
M_diff = average - y
print("average age: ", average)
print("Tanya: {0}\nMitya: {1}\n".format(abs(T_diff), abs(M_diff)))

```

```
individual x
C:\Users\vika1\AppData\Local\Microsoft\Windows\
16
3
average age: 9.5
Tanya: 6.5
Mitya: 6.5

Process finished with exit code 0
```

Рисунок 10 – Результат ее выполнения в IDE PyCharm

```
C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии\лаба 4\PyCharm>python individual.py
14
6
average age: 10.0
Tanya: 4.0
Mitya: 4.0
```

Рисунок 11 – Результат выполнение программы в командной строке

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

```
vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/
/Lr4 (develop)
$ git commit -m "task"
[develop 4854feb] task
4 files changed, 19 insertions(+)
create mode 100644 PyCharm/arithmetic.py
create mode 100644 PyCharm/individual.py
create mode 100644 PyCharm/numbers.py
create mode 100644 PyCharm/user.py
```

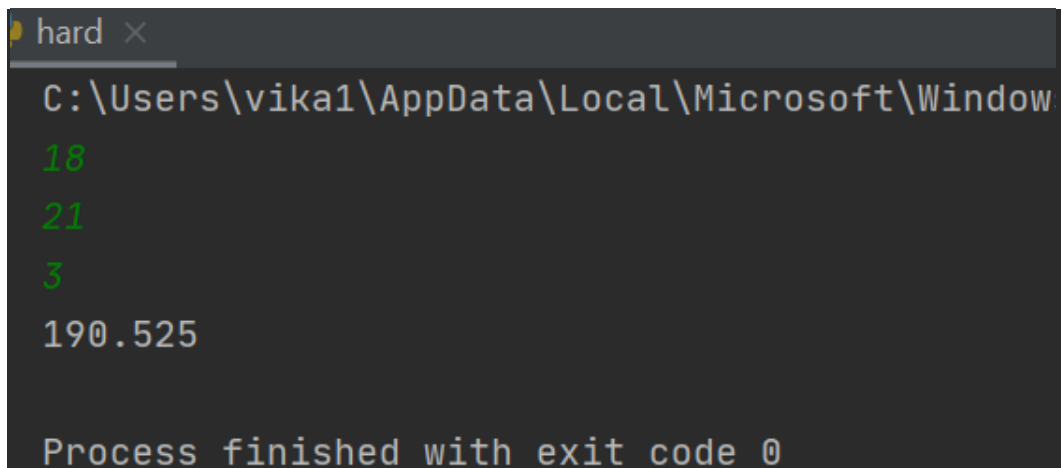
Рисунок 12 – Коммит основных заданий

## Задачи повышенной сложности

5. Даны целые числа  $h, m, s$  ( $0 < h \leq 23, 0 \leq m \leq 59, 0 \leq s \leq 59$ ), указывающие момент времени: « $h$  часов,  $m$  минут,  $s$  секунд». Определить угол (в градусах) между положением часовой стрелки в начале суток и в указанный момент времени.

Код программы:

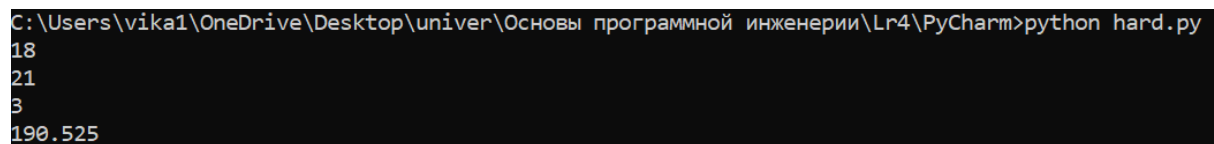
```
h = int(input())
m = int(input())
s = int(input())
if (h >= 12):
    h -= 12
angle = (h * 30 + m / 2 + s / 120)
print(angle)
```



```
hard x
C:\Users\vika1\AppData\Local\Microsoft\Windows
18
21
3
190.525

Process finished with exit code 0
```

Рисунок 13 – Результат ее выполнения в IDE PyCharm



```
C:\Users\vika1\OneDrive\Desktop\univer\Основы программной инженерии\Lr4\PyCharm>python hard.py
18
21
3
190.525
```

Рисунок 14 – Результат выполнение программы в командной строке



```

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive
/Lr4 (develop)
$ git commit -m "hard"
[develop b3dabf3] hard
1 file changed, 7 insertions(+)
create mode 100644 PyCharm/hard.py

```

Рисунок 15 – Коммит индивидуального задания

13. Выполните слияние ветки для разработки с веткой master.

```

vika1@LAPTOP-3EV1GGNJ MINGW64 ~/OneDrive/Desktop/Lr4 (main)
$ git merge develop
Updating f7e3693..b3dabf3
Fast-forward
 PyCharm/arithmetic.py | 2 ++
 PyCharm/hard.py       | 7 +++++++
 PyCharm/individual.py | 7 +++++++
 PyCharm/numbers.py   | 6 ++++++
 PyCharm/user.py       | 4 +++++
5 files changed, 26 insertions(+)
create mode 100644 PyCharm/arithmetic.py
create mode 100644 PyCharm/hard.py
create mode 100644 PyCharm/individual.py
create mode 100644 PyCharm/numbers.py
create mode 100644 PyCharm/user.py

```

Рисунок 16 – Слияние веток

14. Отправьте сделанные изменения на сервер GitHub.

Вопросы для защиты работы

1. Опишите основные этапы установки Python в Windows и Linux.
1. Для установки интерпретатора Python первое, что нужно сделать – это скачать дистрибутив. Загрузить его можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>.
2. Запустить скачанный установочный файл.
3. Выбрать способ установки.

4. Отметить необходимые опции установки (доступно при выборе Customize installation).

5. Выберите место установки (доступно при выборе Customize installation).

При установке для Linux, в случае ошибки необходимо либо собрать Python из исходников либо 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 является дистрибутивом языков программирования таких как Python и R для науки о данных и машинного обучения, а Python — это язык программирования высокого уровня общего назначения.

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

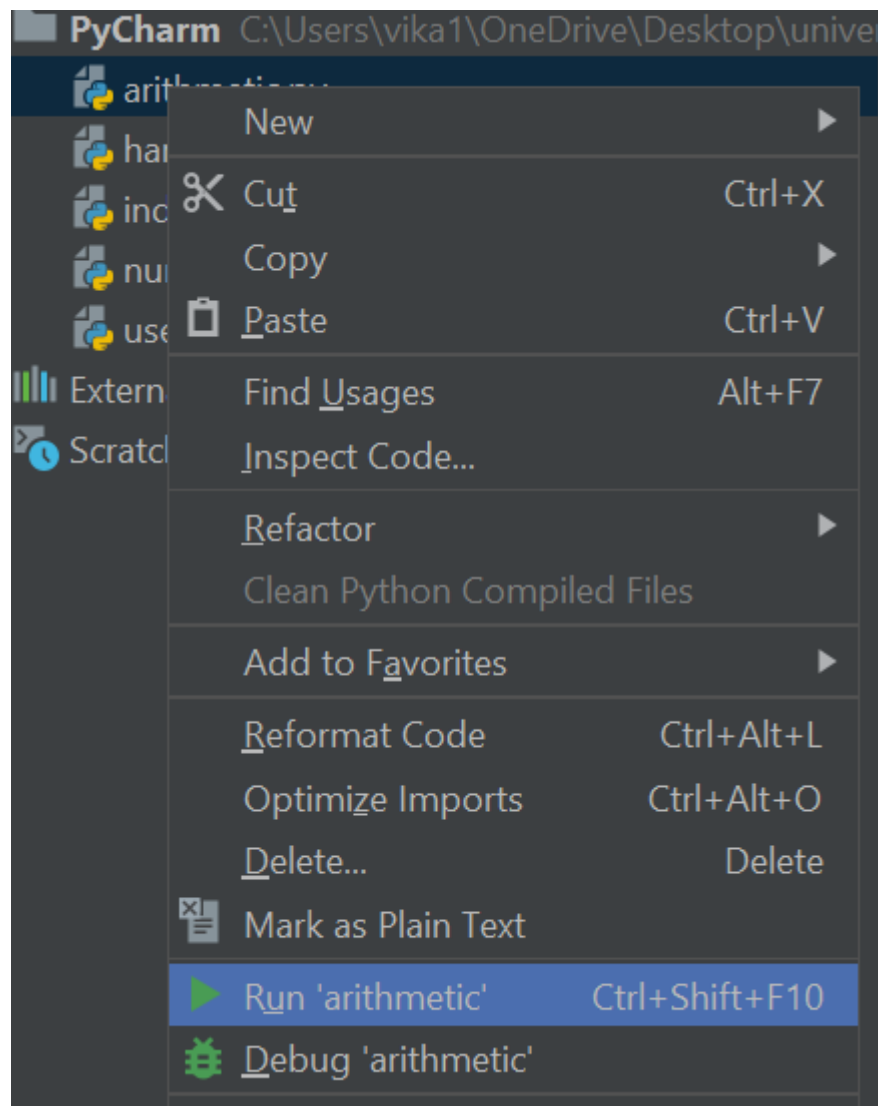
Создайте ноутбук для разработки, для этого нажмите на кнопку New (в правом углу окна) и в появившемся списке выберите Python. В результате будет создана новая страница в браузере с ноутбуком. Введите в первой ячейке команду «`print("Hello, World!")`» и нажмите Alt+Enter на клавиатуре. Ниже ячейки должна появиться соответствующая надпись.

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

При создании нового проекта в PyCharm есть возможность выбрать путь до проекта и интерпретатор.

5. Как осуществить запуск программы с помощью IDE PyCharm?

Правой кнопкой в любом месте или по файлу слева и выбрать из появившегося меню пункт «Run»



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

Интерактивный режим позволяет вводить команды, которые сразу же будут обрабатываться, это можно использовать в качестве калькулятора.

Пакетный режим позволяет запустить файл с исходным кодом.

7. Почему язык программирования Python называется языком динамической типизации?

Это означает, что одна и та же переменная в разное время может ссылаться на данные разного типа.

8. Какие существуют основные типы в языке программирования Python?

1. None (неопределенное значение переменной)

2. Логические переменные (Boolean Type)

3. Числа (Numeric Type)

3.1. int – целое число

3.2. float – число с плавающей точкой

3.3. complex – комплексное число

4. Списки (Sequence Type)

4.1. list – список

4.2. tuple – кортеж

4.3. range – диапазон

5. Строки (Text Sequence Type )

6. Бинарные списки (Binary Sequence Types)

6.1. bytes – байты

6.2. bytearray – массивы байт

6.3. memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer

7. Множества (Set Types)

7.1. set – множество

7.2. frozenset – неизменяемое множество

8. Словари (Mapping Types)

8.1. dict – словарь

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

При создании переменной, вначале создается объект, который имеет уникальный идентификатор, тип и значение, после этого переменная может ссылаться на созданный объект.

Создание переменных и объектов в Python происходит с помощью оператора присваивания («=»). Записывается имя переменной, оператор и значение, например, «a = 5». Множественное присваивание (позиционное присваивание) в Python реализуется следующим образом: «a, b, c = 5, 3, 1»

10. Как получить список ключевых слов в Python?

Нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

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

id() – возвращает уникальный идентификатор объекта в программе.

type() – возвращает тип переменной.

12. Что такое изменяемые и неизменяемые типы в Python.

Изменяемый объект можно изменить после его создания, а неизменяемый – нет. Во втором случае фактически мы не изменяем значение переменной, а создаем другой объект с тем же именем и присваиваем ему другое значение. Мы связываем имя переменной с новым значением. Теперь, при ее вызове, он будет выводить новое значение и ссылаться на новое местоположение.

13. Чем отличаются операции деления и целочисленного деления?

Во втором случае не учитывается остаток от деления.

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.

15. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`.

Библиотека `math` содержит большое количество часто используемых математических функций, такие как округление, логарифмы, факториал, модуль числа, экспонента ( $e^x$ ), степень, квадратный корень, синус/косинус, числа  $\pi$  и  $e$  и т.д.

Модуль `cmath` – предоставляет функции для работы с комплексными числами. Из отличных функций можно выделить преобразование к полярным координатам и преобразование из полярных координат.

**`cmath.isfinite(x)`** - True, если действительная и мнимая части конечны.

**`cmath.isinf(x)`** - True, если либо действительная, либо мнимая часть бесконечна.

**`cmath.isnan(x)`** - True, если либо действительная, либо мнимая часть NaN.

16. Каково назначение именованных параметров `sep` и `end` в функции `print()`?

`Sep()` устанавливает отличный от пробела разделитель строк.

`End()` указывает, что делать, после вывода строки (по умолчанию стоит переход на новую 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()` позволяет форматировать выводимые строки. Рассмотрим пример:

```
print("This is a {0}. It's {1}.".format("ball", "red"))
```

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 также может осуществляться в старом (Си-) стиле. Он схож с тем, как происходит вывод на экран в языке C. Пример:

```
print("It's %s, %d. Level: %f" % (pupil, old, grade))
```

Здесь вместо трех комбинаций символов `%s`, `%d`, `%f` подставляются значения переменных `pupil`, `old`, `grade`. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число.

F-Строки являются упрощенной версией метода `format()`. F-strings являются строковыми литералами с «f» в начале и фигурные скобки, содержащие выражения, которые в дальнейшем будут заменены своими значениями. Пример:

```
print(f"Hello, {name}. You are {2*8}.") //name – переменная
```

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

С помощью функции `input()` можно получить вводимые с консоли данные. Однако они будут принадлежать к строковому типу, чтоб получить число нужно использовать функции преобразования типов:

```
c = float(input("Enter temperature (Celsius) : "))
```

Выводы: в ходе выполнения работы исследование процесса установки и базовых возможностей языка Python, получен опыт работы с IDE PyCharm, Anaconda, консоль.