

**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 2 курса

группы ПИЖ-б-о-21-1

Пуценко Иван Алексеевич

Ставрополь 2022

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:

```

C:\Users\FonK\Desktop\ОПИ\labRab_4>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/FonK/Desktop/ОПИ/labRab_4/.git/hooks]

C:\Users\FonK\Desktop\ОПИ\labRab_4>git branch
* develop
  main

C:\Users\FonK\Desktop\ОПИ\labRab_4>

```

Рисунок 1 Организация репозитория согласно модели ветвления git-flow

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

"This is `имя`"

"It is `возраст`"

"(S)he live in `место\_жительства`"

The screenshot shows a code editor with a file explorer on the left and a console window at the bottom. The file explorer shows a project named 'labRab\_4' with files: .gitignore, arithmetic.py, individual.py, LICENSE, numbers.py, README.md, and user.py. The code editor displays the following Python code in user.py:

```

1 name = input("That is your name? ")
2 age = input("How old are you? ")
3 address = input("Where are you life? ")
4
5 print("This is", name)
6 print("It is", age)
7 print("He life in", address)
8

```

The console window shows the execution of the program with the following input and output:

```

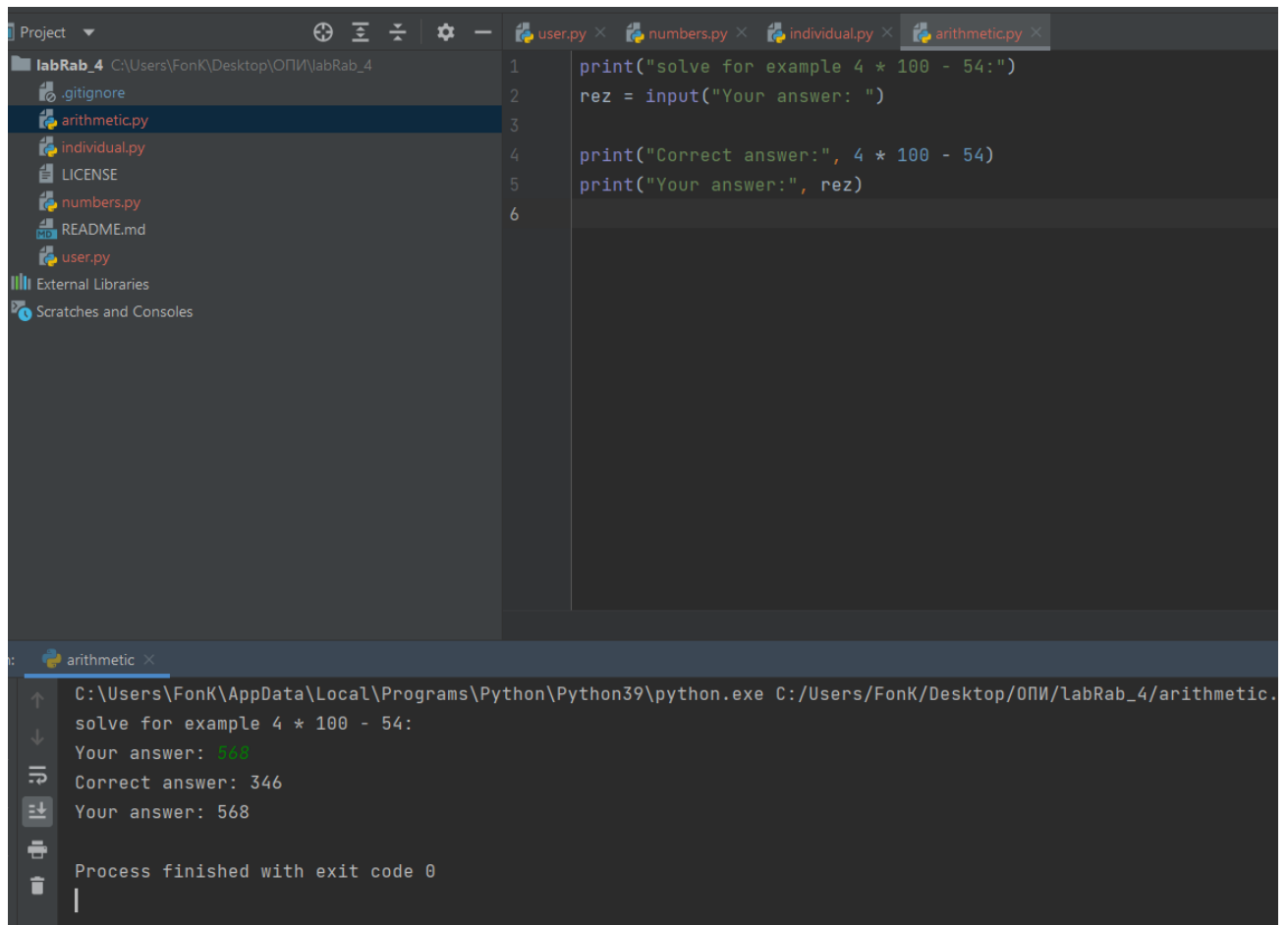
C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/labRab_4/user.py
That is your name? ivan
How old are you? 19
Where are you life? stavropol
This is ivan
It is 19
He life in stavropol

Process finished with exit code 0

```

Рисунок 2 Программа user в репозитории

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



The screenshot shows a Python IDE with a project named 'labRab\_4'. The file explorer on the left lists files: .gitignore, arithmetic.py, individual.py, LICENSE, numbers.py, README.md, and user.py. The main editor displays the code for arithmetic.py:

```
1 print("solve for example 4 * 100 - 54:")
2 rez = input("Your answer: ")
3
4 print("Correct answer:", 4 * 100 - 54)
5 print("Your answer:", rez)
6
```

Below the editor, the console output shows the program's execution:

```
C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/LabRab_4/arithmetic.py
solve for example 4 * 100 - 54:
Your answer: 568
Correct answer: 346
Your answer: 568
Process finished with exit code 0
```

Рисунок 3 Программа arithmetic.py

4. 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.

The screenshot shows an IDE with a project named 'labRab\_4'. The file explorer on the left lists files: .gitignore, arithmetic.py, individual.py, LICENSE, numbers.py (selected), README.md, user.py, External Libraries, and Scratches and Consoles. The editor displays the code for 'numbers.py' with line numbers 1 to 12. The code prompts the user to enter 4 numbers, calculates the sum of the first two and the last two, and then prints the result of the division of these sums, formatted to two decimal places. The console at the bottom shows the execution of 'python.exe' for 'numbers.py', with input values 4, 6, 8, and 9, and an output of 0.59. The process finished with exit code 0.

```
1 print("enter 4 any numbers:")
2 n1 = int(input())
3 n2 = int(input())
4 n3 = int(input())
5 n4 = int(input())
6
7 sum1 = n1 + n2
8 sum2 = n3 + n4
9 rez = sum1 / sum2
10
11 print(format(rez, ".2"))
12
```

numbers ×  
C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/labRab\_4/numb  
enter 4 any numbers:  
4  
6  
8  
9  
0.59  
Process finished with exit code 0

Рисунок 4 Программа numbers.py

5. Написал программу для индивидуального задания:

*Даны катеты прямоугольного треугольника. Найти его периметр.(вариант 23)*

The screenshot shows the same IDE with the file explorer listing 'individual.py' as the selected file. The editor displays the code for 'individual.py' with line numbers 1 to 8. The code prompts the user to enter two catheti, calculates the hypotenuse using the Pythagorean theorem, and then prints the perimeter of the right-angled triangle. The console at the bottom shows the execution of 'python.exe' for 'individual.py', with input values 12 and 0, and an output of 12.0. The process finished with exit code 0.

```
1 cat1 = int(input())
2 cat2 = int(input())
3
4 hup = pow(pow(cat1, 2) + pow(cat2, 2), 1 / 2)
5 P = cat1 + cat2 + hup
6
7 print(P)
8
```

individual ×  
C:\Users\FonK\AppData\Local\Programs\Python\Python39\python.exe C:/Users/FonK/Desktop/ОПИ/labRab\_4/individual.  
12.0  
Process finished with exit code 0

Рисунок 5 Программа индивидуального задания

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

```
C:\Users\FonK\Desktop\ОПИ\labRab_4>git add .  
  
C:\Users\FonK\Desktop\ОПИ\labRab_4>git commit -m "main programms"  
[develop f4a86a0] main programms  
5 files changed, 31 insertions(+)  
create mode 100644 arithmetic.py  
create mode 100644 individual.py  
create mode 100644 numbers.py  
create mode 100644 user.py
```

Рисунок 6 Коммит изменений в ветку develop

```
C:\Users\FonK\Desktop\ОПИ\labRab_4>git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.  
  
C:\Users\FonK\Desktop\ОПИ\labRab_4>git merge develop  
Updating a5163b8..f4a86a0  
Fast-forward  
.gitignore      | 1 +  
arithmetic.py   | 5 +++++  
individual.py   | 7 +++++++  
numbers.py      | 11 ++++++++  
user.py         | 7 +++++++  
5 files changed, 31 insertions(+)  
create mode 100644 arithmetic.py  
create mode 100644 individual.py  
create mode 100644 numbers.py  
create mode 100644 user.py  
  
C:\Users\FonK\Desktop\ОПИ\labRab_4>
```

Рисунок 7 Слияние ветки develop с веткой main

```
C:\Users\FonK\Desktop\ОПИ\labRab_4>git push  
Enumerating objects: 9, done.  
Counting objects: 100% (9/9), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (7/7), done.  
Writing objects: 100% (7/7), 913 bytes | 182.00 KiB/s, done.  
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/kvakaet/labRab_4.git  
a5163b8..f4a86a0 main -> main
```

Рисунок 8 push коммитов на уд. репозиторий

### **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 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 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 (frozenset).

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  $\pi$ .

`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.

**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())`.