

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №4 по дисциплине
«Основы программной инженерии»

Выполнил:
Прокопов Дмитрий Владиславович,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

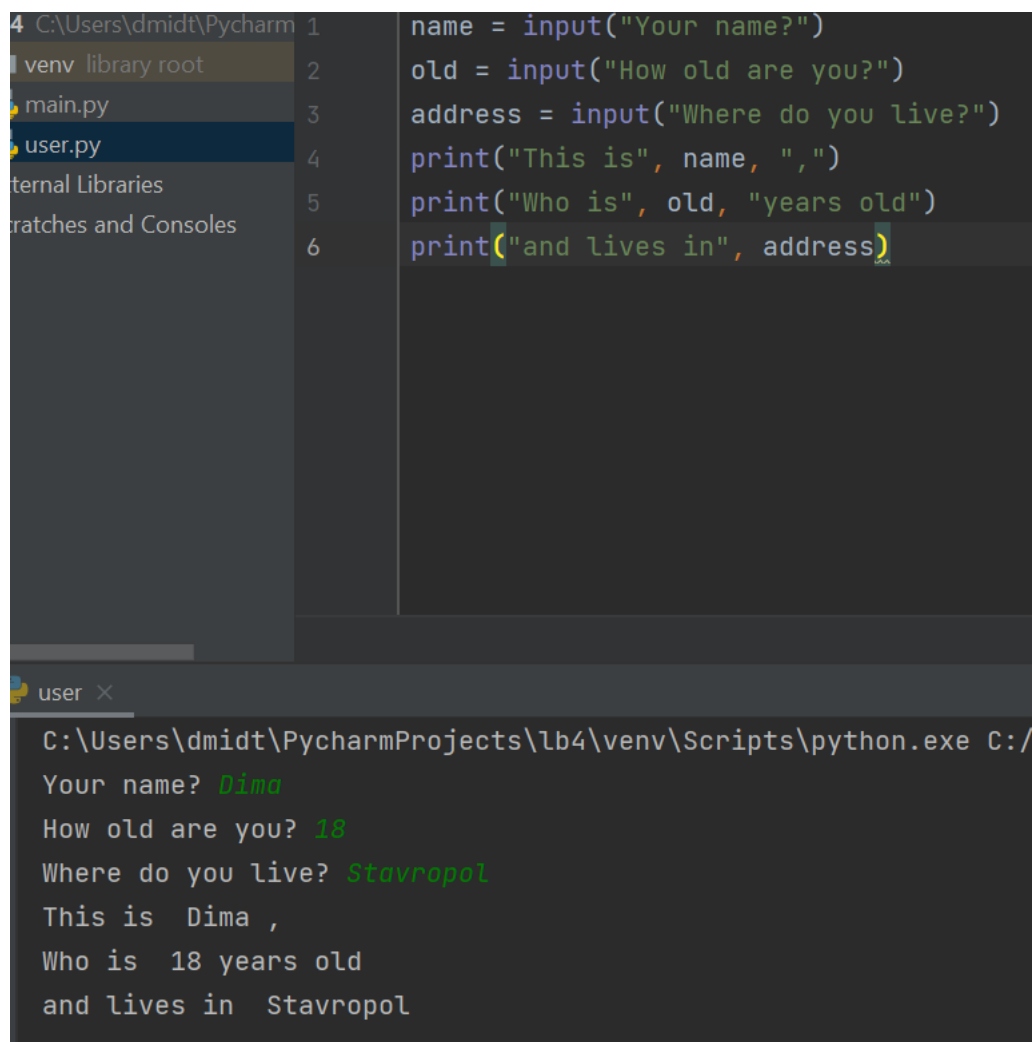
Ход работы

```
C:\Users\dmidt\lb\lb4>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/dmidt/lb/lb4/.git/hooks]
```

Рис. 1 – организация репозитория в соответствии с моделью git-flow.

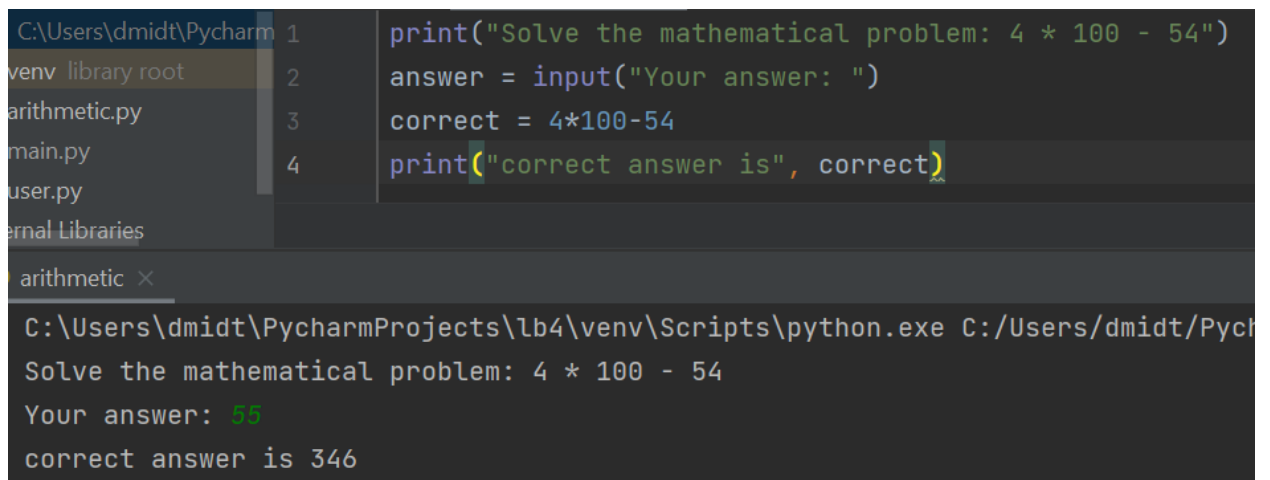


The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays the file structure of the 'lb4' project, including 'venv', 'library root', 'main.py', and 'user.py'. The 'user.py' file is selected and its code is visible in the main editor. The code is a simple Python script that prompts the user for their name, age, and address, and then prints the information in a formatted string. Below the editor, the 'Run' tool window shows the output of the program, indicating that it was executed successfully using the Python interpreter from the virtual environment.

```
1 name = input("Your name?")
2 old = input("How old are you?")
3 address = input("Where do you live?")
4 print("This is", name, ",")
5 print("Who is", old, "years old")
6 print("and lives in", address)
```

```
C:\Users\dmidt\PycharmProjects\lb4\venv\Scripts\python.exe C:/
Your name? Dima
How old are you? 18
Where do you live? Stavropol
This is Dima ,
Who is 18 years old
and lives in Stavropol
```

Рис. 2 – программа user.py



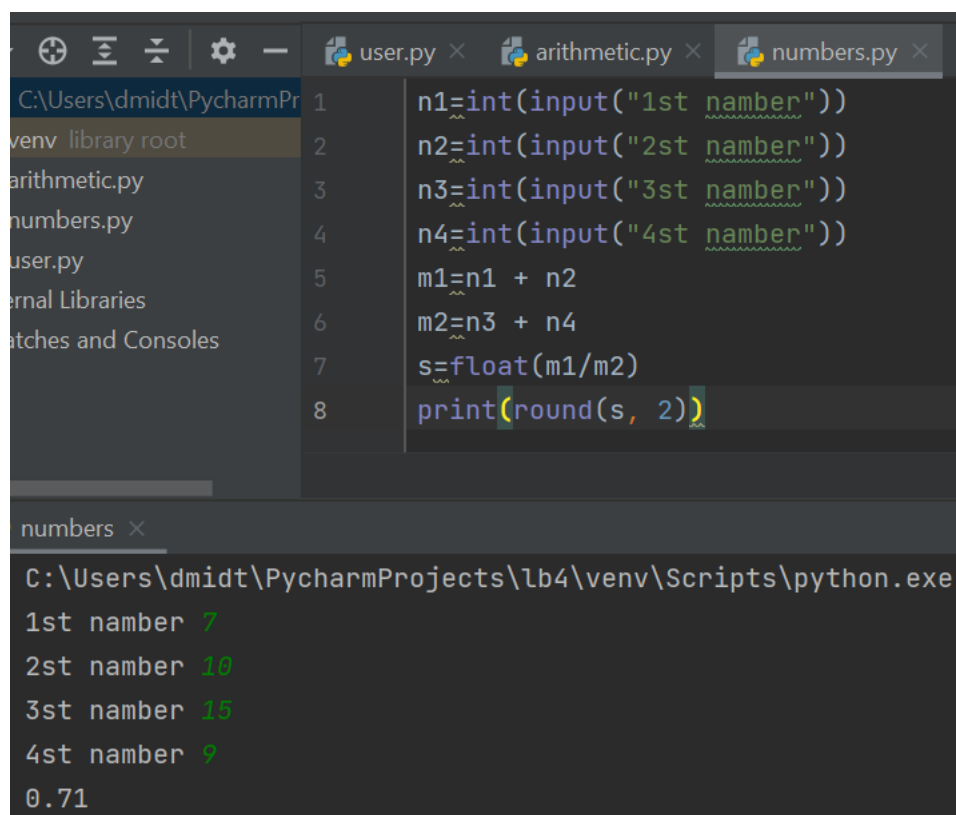
The screenshot shows the PyCharm IDE with the file explorer on the left displaying a project structure with files like `main.py`, `user.py`, and `arithmetic.py`. The `arithmetic.py` file is open in the editor, showing the following code:

```
1 print("Solve the mathematical problem: 4 * 100 - 54")
2 answer = input("Your answer: ")
3 correct = 4*100-54
4 print("correct answer is", correct)
```

Below the editor, the console window shows the execution output:

```
C:\Users\dmidt\PycharmProjects\lb4\venv\Scripts\python.exe C:/Users/dmidt/PycharmProjects\lb4\venv\Scripts\python.exe C:/Users/dmidt/PycharmProjects\lb4\venv\Scripts\python.exe
Solve the mathematical problem: 4 * 100 - 54
Your answer: 55
correct answer is 346
```

Рис. 3 – программа `arithmetic.py`



The screenshot shows the PyCharm IDE with the file explorer on the left displaying a project structure with files like `main.py`, `user.py`, `arithmetic.py`, and `numbers.py`. The `numbers.py` file is open in the editor, showing the following code:

```
1 n1=int(input("1st number"))
2 n2=int(input("2st number"))
3 n3=int(input("3st number"))
4 n4=int(input("4st number"))
5 m1=n1 + n2
6 m2=n3 + n4
7 s=float(m1/m2)
8 print(round(s, 2))
```

Below the editor, the console window shows the execution output:

```
C:\Users\dmidt\PycharmProjects\lb4\venv\Scripts\python.exe
1st number 7
2st number 10
3st number 15
4st number 9
0.71
```

Рис. 4 – программа `numbers.py`

The screenshot shows the PyCharm IDE with the 'individual.py' file open. The file contains the following Python code:

```
1 s=int(input("population size "))
2 t=int(input("square "))
3 p=s/t
4 print("population density ", round(p, 2))
```

The console output shows the program being executed with the following inputs and results:

```
C:\Users\dmidt\PycharmProjects\lb4\venv\Scripts\python.exe C:/Users/dmidt/
population size 50000
square 1200
population density 41.67
```

Рис. 5 – программа individual.py

The screenshot shows the PyCharm IDE with the 'increased_complexity.py' file open. The file contains the following Python code:

```
1 a = int(input("three-digit number "))
2 b = int(input("two-digit number "))
3 a1 = a % 10
4 b1 = b % 10
5 a2 = (a % 100) // 10
6 b2 = b // 10
7 a3 = a // 100
8 t1 = (a1 + b1) % 10
9 t2 = (a2 + b2 + (a1 + b1) // 10) % 10
10 t3 = a3 + ((a2 + b2 + (a1 + b1) // 10) // 10)
11 print("number of hundreds", t3)
12 print("number of tens", t2)
13 print("number of units", t1)
```

The console output shows the program being executed with the following inputs and results:

```
C:\Users\dmidt\PycharmProjects\lb4\venv\Scripts\python.exe C:/Users/dmic
three-digit number 567
two-digit number 64
number of hundreds 6
number of tens 3
number of units 1
```

Рис. 6 – программа increased_complexity.py

19. Известны количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.

Рис. 8 – индивидуальное задание

3. Даны цифры двух десятичных целых чисел: трехзначного $a_3 a_2 a_1$ и двузначного $b_2 b_1$, где a_1 и b_1 – число единиц, a_2 и b_2 – число десятков, a_3 – число сотен. Получить цифры числа, равного сумме заданных чисел (известно, что это число трехзначное). Числа-слагаемые и число-результат не определять; условный оператор не использовать.

Рис. 9 – задание повышенной сложности

Ответы на вопросы:

1) 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.

В данном окне предлагается два варианта Install Now и Customize installation. При выборе Install Now, Python установится в папку по указанному пути. Помимо самого интерпретатора будет установлен IDLE (интегрированная среда разработки), pip (пакетный менеджер) и документация, а также будут созданы соответствующие ярлыки и установлены связи файлов, имеющие расширение .py с интерпретатором Python. Customize installation – это вариант настраиваемой установки. Опция Add python 3.5 to PATH нужна для того, чтобы появилась возможность запускать интерпретатор без указания полного пути до исполняемого файла при работе в командной строке.

3. Отметьте необходимые опции установки (доступно при выборе Customize installation)

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Рекомендуется выбрать все опции.

Documentation – установка документаций.

pip – установка пакетного менеджера pip.

tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).

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

Помимо указания пути, данное окно позволяет внести дополнительные изменения в процесс установки с помощью опций:

Install for all users – Установить для всех пользователей. Если не выбрать данную опцию, то будет предложен вариант инсталляции в папку пользователя, устанавливающего интерпретатор.

Associate files with Python – Связать файлы, имеющие расширение .py, с Python. При выборе данной опции будут внесены изменения в Windows, позволяющие запускать Python скрипты по двойному щелчку мыши.

Create shortcuts for installed applications – Создать ярлыки для запуска

приложений.

Add Python to environment variables – Добавить пути до интерпретатора Python в переменную PATH.

Precompile standard library – Провести прекомпиляцию стандартной библиотеки.

Последние два пункта связаны с загрузкой компонентов для отладки, их мы устанавливать не будем Linux: Чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить, набрав в терминале «python» или «python3»

В первом случае, вы запустите Python 2 во втором – Python 3. В будущем, скорее всего, во всех дистрибутивах Linux, включающих Python, будет входить только третья версия. 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) Для удобства запуска примеров и изучения языка Python, настоятельно рекомендуется установить на свой ПК пакет Anaconda. 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) 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», 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 выберете 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) При создании нового проекта нужно будет указать путь до него и интерпретатор.

5) После создания нового проекта нужно добавить python файл в проект.

6) Интерактивный режим – непосредственное выполнение команд одна за другой в консоли. Пакетный режим – запуск программы из файла.

7) Потому что тип переменной определяется непосредственно при выполнении программы.

8) К основным встроенным типам относятся:

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
 - int – целое число
 - float – число с плавающей точкой

- `complex` – комплексное число

4. Списки (Sequence Type)

- `list` – список

- tuple – кортеж
 - range – диапазон
5. Строки (Text Sequence Type)
 - str
 6. Бинарные списки (Binary Sequence Types)
 - bytes – байты
 - bytearray – массивы байт
 - memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer
 7. Множества (Set Types)
 - set – множество
 - frozenset – неизменяемое множество
 8. Словари (Mapping Types)
 - dict – словарь

9) Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана. Целочисленное значение 5 в рамках языка Python по сути своей является объектом. Объект, в данном случае – 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 (об этом чуть позже). 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, 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). Имя переменной не должно совпадать с ключевыми словами интерпретатора Python.

10) Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль `keyword` и воспользоваться командой `keyword.kwlist`.

11) Для того, чтобы посмотреть на объект с каким идентификатором ссылается данная переменная, можно использовать функцию `id()`. Тип переменной можно определить с помощью функции `type()`.

12) 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).

Как уже было сказано ранее, при создании переменной, вначале создается объект, который имеет уникальный идентификатор, тип и значение, после этого переменная может ссылаться на созданный объект.

Неизменяемость типа данных означает, что созданный объект больше не изменяется. Например, если мы объявим переменную $k = 15$, то будет

создан объект со значением 15, типа `int` и идентификатором, который можно узнать с помощью функции `id()`.

13) При обычном делении результатом операции будет вещественное число с плавающей точкой. При целочисленном делении результатом будет целое число, показывающее количество целых чисел b в числе a , к примеру.

14) 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 и мнимую части.

15) В стандартную поставку Python входит библиотека «`math`», в которой содержится большое количество часто используемых математических функций. Для работы с данным модулем его предварительно нужно импортировать. Библиотека «`cmath`» содержит в себе функции для работы с комплексными числами.

16) Через параметр «`sep`» можно указать отличный от пробела разделитель строк. Параметр «`end`» позволяет указывать, что делать, после вывода строки. По умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку.

17) 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). Нанулевое место подставится первый аргумент метода `format()`, на место с номером 1 – второй и т. д. «`f`-строки»: Форматирование, которое появилось в Python 3.6 (PEP 498). Этот способ похож на форматирование с помощью метода `format()`, но гибче, читабельней и быстрее. Пример:

```
>>> name = "Дмитрий"
>>> age = 25
>>> print(f"Меня зовут {name} Мне {age} лет.")
>>> Меня зовут Дмитрий. Мне 25 лет.
```

18) Даже, если ввести число, функция `input()` все равно вернет его строковое представление. Чтобы получить число, нужно использовать функции преобразования типов. Пример:

```
qtyOranges = int(input("Сколько апельсинов? "))  
priceOrange = float(input("Цена одного апельсина?"  
"))  
sumOranges = qtyOranges * priceOrang
```