

**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

Кочкаров Умар Ахматович

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:

The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'umarkochkarov' and the 'Repository name' is 'lb2.1'. A green checkmark is next to the repository name. Below these fields, there is a 'Description (optional)' text area. Further down, there are two radio buttons for 'Public' and 'Private'. The 'Public' option is selected. Below this, there is a section 'Initialize this repository with:' with a checkbox for 'Add a README file' which is checked. There is also a section 'Add .gitignore' with a dropdown menu showing '.gitignore template: Python'. Below that is a 'Choose a license' section with a dropdown menu showing 'License: MIT License'. At the bottom, there is a green button labeled 'Create repository'.

Рисунок 1.1 Создание репозитория

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1
$ git clone https://github.com/umarkochkarov/lb2.1.git
Cloning into 'lb2.1'...
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 1.2 Клонирование репозитория на ПК

```

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Ла62.1/1b2.1 (main)
$ 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/erken/Desktop/Универ/Основы крос/Ла62.1/1b2.1/.git/hooks]

```

Рисунок 1.3 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:



```

1 name = input("What is your name? - ")
2 age = input('How old are you? - ')
3 location = input('Where are you live? - ')
4 print('This is', name)
5 print('It is', age)
6 print('(S)he live in', location)
7

```

Run - 1b2.1

```

C:\Users\erken\AppData\Local\Programs\Python\Python310\python.exe "C:/Users/erken/Desktop/Универ/Основы крос/Ла62.1/1b2.1/user.py"
What is your name? - Umar
How old are you? - 19
Where are you live? - Stavropol
This is Umar
It is 19
(S)he live in Stavropol

```

Рисунок 2. Программа user

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.



```

1 answer = input('Calculate example\n4*100-54=')
2 print('User answer:', answer)
3 print('Right answer:', 4*100-54)

```

Run - arithmetic.py

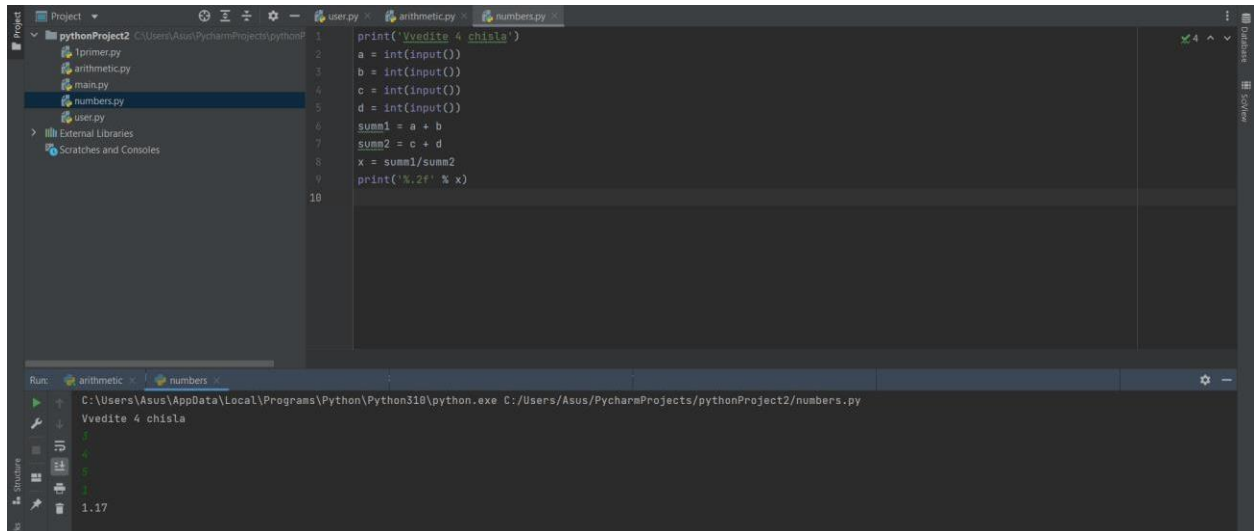
```

C:\Users\erken\AppData\Local\Programs\Python\Python310\python.exe "C:/Users/erken/Desktop/Универ/Основы крос/Ла62.1/1b2.1/arithmetic.py"
Calculate example
4*100-54=10
User answer: 10
Right answer: 346

```

Рисунок 3. Программа arithmetic

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 результат на экран с точностью до сотых.



The screenshot shows the PyCharm IDE with a project named 'pythonProject2'. The file explorer on the left shows a folder 'user.py' containing 'numbers.py'. The editor window displays the code for 'numbers.py':

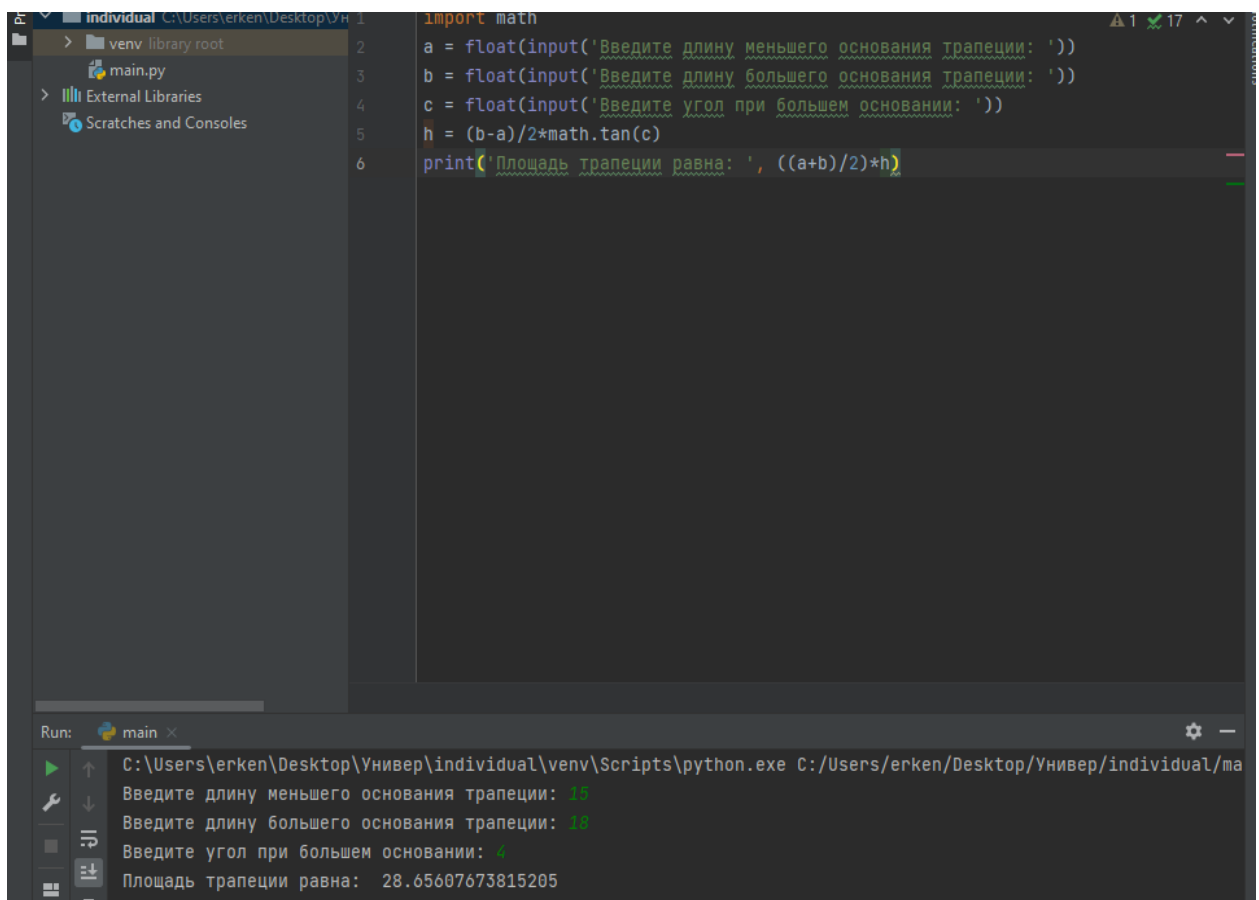
```
1 print('Введите 4 числа')
2 a = int(input())
3 b = int(input())
4 c = int(input())
5 d = int(input())
6 sum1 = a + b
7 sum2 = c + d
8 x = sum1/sum2
9 print('%2f' % x)
10
```

The Run window at the bottom shows the command: `C:\Users\Asus\AppData\Local\Programs\Python\Python310\python.exe C:/Users/Asus/PycharmProjects/pythonProject2/numbers.py`. The output is: `Введите 4 числа` followed by a blank line and the result `1.17`.

Рисунок 4. Программа numbers

5. Индивидуальное задание(вариант 8)

Даны основания равнобедренной трапеции и угол при большем основании. Найти площадь трапеции.



The screenshot shows the PyCharm IDE with a project named 'individual'. The file explorer on the left shows a folder 'venv' containing 'main.py'. The editor window displays the code for 'main.py':

```
1 import math
2 a = float(input('Введите длину меньшего основания трапеции: '))
3 b = float(input('Введите длину большего основания трапеции: '))
4 c = float(input('Введите угол при большем основании: '))
5 h = (b-a)/2*math.tan(c)
6 print('Площадь трапеции равна: ', ((a+b)/2)*h)
```

The Run window at the bottom shows the command: `C:\Users\erken\Desktop\Универ\individual\venv\Scripts\python.exe C:/Users/erken/Desktop/Универ/individual/main.py`. The output is: `Введите длину меньшего основания трапеции: 15`, `Введите длину большего основания трапеции: 18`, `Введите угол при большем основании: 4`, and `Площадь трапеции равна: 28.65607673815205`.

Рисунок 5. Программа individual

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

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (develop)
$ git branch
* develop
  main

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (develop)
$ 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

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (develop)
$ git add .

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (develop)
$ git commit -m "Added programmes"
[develop 97a4998] Added programmes
 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 6.1 Коммит изменений в ветку develop

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (main)
$ git merge develop
Updating f819e6d..97a4998
Fast-forward
 arithmetic.py | 3 +++
 individual.py | 6 +++++
 numbers.py    | 9 ++++++++
 user.py       | 6 +++++
 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 6.2 Слияние ветки develop с веткой main

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/Универ/Основы крос/Лаб2.1/1b2.1 (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 953 bytes | 953.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/umarkochkarov/1b2.1.git
 f819e6d..97a4998 main -> main
```

Рисунок 6.3 Push коммитов на удаленный сервер

umarkochkarov Added programms			97a4998 2 minutes ago	🕒 2 commits
📄	.gitignore	Initial commit		2 days ago
📄	LICENSE	Initial commit		2 days ago
📄	README.md	Initial commit		2 days ago
📄	arithmetic.py	Added programms		2 minutes ago
📄	individual.py	Added programms		2 minutes ago
📄	numbers.py	Added programms		2 minutes ago
📄	user.py	Added programms		2 minutes ago

Рисунок 6.3 Изменения на удаленном 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 отобразиться 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 с выбором 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 комплексно-сопряженного числа 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())`.