

**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 программной инженерии»  
Тема: «Основы ветвления GIT»

Выполнил: студентка  
2 курса группы  
Пиж-б-о-21-1  
Логвинов Иван  
Васильевич

Ставрополь 2022

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

Владелец \*      Имя репозитория \*

Иван Логвинов11 / Lab\_2.1 ✓

Отличные названия репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет **потенциальной системы?**

Описание (необязательно)

☒ **Общедоступный**  
Любой пользователь Интернета может увидеть этот репозиторий. Вы сами выбираете, кто может взять на себя обязательства.

☐ **Личное**  
Вы выбираете, кто может видеть и фиксировать в этом репозитории.

**Инициализируйте этот репозиторий с помощью:**  
Пропустите этот шаг, если вы импортируете существующий репозиторий.

☒ **Добавьте файл README**  
Здесь вы можете написать длинное описание для вашего проекта. [Узнать больше.](#)

**Добавить .gitignore**  
Выберите, какие файлы не отслеживать, из списка шаблонов. [Узнать больше.](#)

.шаблон gitignore: Python ▼

**Выберите лицензию**  
Лицензия сообщает другим пользователям, что они могут и чего не могут делать с вашим кодом. [Узнать больше.](#)

Лицензия: MIT License ▼

**Вы создаете общедоступный репозиторий в своем личном кабинете.**

**Создать репозиторий**

Рисунок 1 – был создан общедоступный репозиторий на Github с лицензией MIT и языком программирования Python

```
Командная строка
Microsoft Windows [Version 10.0.18363.418]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Иван>cd desktop

C:\Users\Иван\Desktop>git clone https://github.com/IvanLogvinov11/Lab_2.1.git
Cloning into 'Lab_2.1'...
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:\Users\Иван\Desktop>
```

Рисунок 2 – Клонирование репозитория

```
C:\Users\Иван\Desktop>cd Lab_2.1

C:\Users\Иван\Desktop\Lab_2.1>git branch ForWork

C:\Users\Иван\Desktop\Lab_2.1>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\Иван\Desktop\Lab_2.1>git checkout ForWork
Switched to branch 'ForWork'

C:\Users\Иван\Desktop\Lab_2.1>
```

Рисунок 3 – Организация репозитория в соответствии с моделью ветвления git-flow

1. 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, местожительства 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("What's your name?:\n")
age = input("What's your age?:\n")
locBirth = input("Where were you born?\n:")

print("\nThen your name is: ", name)
print("Then your age is: ", age)
print("Then you were born in: ", locBirth)
```

```
"C:\Users\Иван\Desktop\Lab_2.1\Новая папка\venv\Scripts\python.exe" "C:/Users/Иван/Desktop/
What's your name?:
ivan
What's your age?:
19
Where were you born?
:stav

Then your name is:  ivan
Then your age is:  19
Then you were born in:  stav

Process finished with exit code 0
|
```

Рисунок 4 – Результат работы программы

2. 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:

```
sol = input("what is the answer?\n4*100-54 = ")

corsol = 346
print("the correct solution is: ", corsol)
print("your answer is: ", sol)
```

```
C:\Users\Иван\PycharmProjects\user\venv\Scripts\python.exe C:/Users/Иван/PycharmProjects/user/arithmetic.py
what is the answer?
4*100-54 = 346
the correct solution is: 346
your answer is: 346

Process finished with exit code 0
|
```

Рисунок 5 – Результат работы программы

3. 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:

```
user.py x arithmetic.py x numbers.py x
1 print("enter four numb: ")
2 numb1 = int(input())
3 numb2 = int(input())
4 numb3 = int(input())
5 numb4 = int(input())
6
7 sum1 = numb1 + numb2
8 sum2 = numb3 + numb4
9 div = sum1 / sum2
10
11 print("we have made some calculation\nthe result is: " + format(div, ".3"))
12
```

```
user x numbers x
C:\Users\Иван\PycharmProjects\user\venv\Scripts\python.exe C:/Users/Иван/PycharmProjects/user/numbers.py
enter four numb:
1
2
3
4
we have made some calculation
the result is: 0.429

Process finished with exit code 0
|
```

Рисунок 6 – Результат работы программы

4. Напишите программу (файл individual.py) для решения

индивидуального задания. Вариант индивидуального задания уточните у преподавателя.

Даны основания и высота равнобедренной трапеции. Найти периметр трапеции.

Код программы:

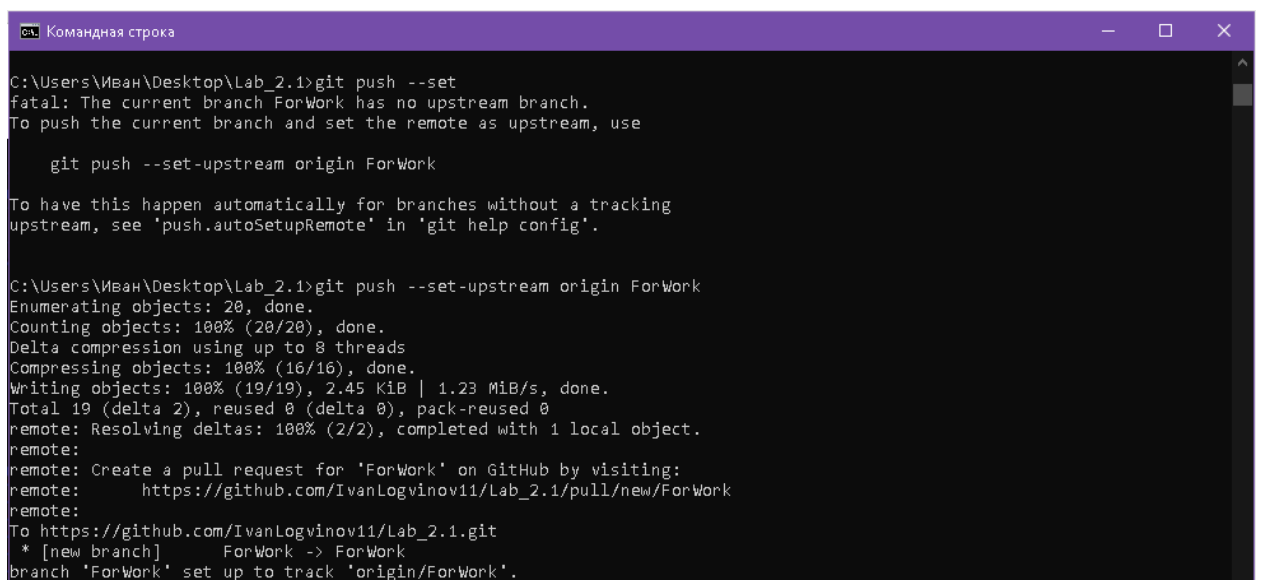
```
from math import sqrt

a = float(input("Длина большего основания:"))
b = float(input("Длина меньшего основания:"))
h = float(input("Высота:"))
print("P =", a + b + 2 * sqrt(sqrt(h) + sqrt(a - b) / 4))
```

```
"C:\Users\Иван\Desktop\Lab_2.1\Новая па
Длина большего основания:12
Длина меньшего основания:11
Высота:4
P = 26.0

Process finished with exit code 0
|
```

Рисунок 7 – Результат выполнения программы



```
Командная строка
C:\Users\Иван\Desktop\Lab_2.1>git push --set
fatal: The current branch ForWork has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin ForWork

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

C:\Users\Иван\Desktop\Lab_2.1>git push --set-upstream origin ForWork
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (19/19), 2.45 KiB | 1.23 MiB/s, done.
Total 19 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
remote:
remote: Create a pull request for 'ForWork' on GitHub by visiting:
remote:   https://github.com/IvanLogvinov11/Lab_2.1/pull/new/ForWork
remote:
To https://github.com/IvanLogvinov11/Lab_2.1.git
 * [new branch]      ForWork -> ForWork
branch 'ForWork' set up to track 'origin/ForWork'.
```

Рисунок 8 – Коммит изменений ветки ForWork

```
Командная строка
C:\Users\Иван\Desktop\Lab_2.1>git merge ForWork
Updating 0d3836c..3ea4658
Fast-forward
.../.idea/.gitignore" | 3 +++
.../.idea/inspectionProfiles/profiles_settings.xml" | 6 +++++
.../.idea/misc.xml" | 4 ++++
.../.idea/modules.xml" | 8 +++++++
.../.idea/vcs.xml" | 6 ++++++
...\.260\321\217 \320\277\320\260\320\277\320\272\320\260.iml" | 10 ++++++++
.../arithmetic.py" | 5 +++++
.../ind.py" | 6 ++++++
.../numbers.py" | 11 ++++++++
.../user.py" | 7 ++++++
10 files changed, 66 insertions(+)
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea/.gitignore"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea/inspectionP
rofiles/profiles_settings.xml"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea/misc.xml"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea/modules.xml"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea/vcs.xml"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/.idea\320\235\32
0\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260.iml"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/arithmetic.py"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/ind.py"
create mode 100644 "\320\235\320\276\320\262\320\260\321\217 \320\277\320\260\320\277\320\272\320\260/numbers.py"
```

Рисунок 8 – Слияние ветки ForWork с веткой main

```
C:\Users\Иван\Desktop\Lab_2.1>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/IvanLogvinov11/Lab_2.1.git
    0d3836c..3ea4658  main -> main

C:\Users\Иван\Desktop\Lab_2.1>
```

Рисунок 9 – Push на удаленный сервер

Вывод: в результате лабораторной работы исследования процесса установки и базовых возможностей языка Python, был получен опыт установки таких программ как Anaconda, PyCharm, Python, также были получены знания работы с PyCharm

#### Контрольные вопросы

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

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

## 5. Выбирать место установки

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?

Пакет Anaconda содержит версии языка Python 2 и 3, набор наиболее часто используемых библиотек и удобную среду разработки

3. Как осуществить проверку работоспособности пакета 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 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. 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, выберете "Add..".
- 2) Далее выберете "System Interpreter";
- 3) Нажмите на 3 точки "..." справа от поля в выбор интерпретатора;
- 4) Укажите путь до интерпретатора



5. Как осуществить запуск программы с помощью IDE PyCharm?  
Shift+F10

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

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.

В проектном.

В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью.

7. Почему язык программирования Python называется языком динамической типизации?

В нем проверка типа происходит во время выполнения, а не компиляции

8. Какие существуют основные типы в языке программирования 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

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?

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 сопряжённого 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`.

`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 пи.

`math.e` - число е.

16. Каково назначение именных 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.

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())`.