

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины «Программирование на Python»

Выполнила:

Ковжого Елизавета Андреевна

2 курс, группа ИВТ-б-о-24-1,

09.03.01 «Информатика и вычислительная техника», направленность (профиль)
«Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р. А., доцент
департамента цифровых,
робототехнических систем и
электроники института
перспективной инженерии.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2025 г.

Тема: основы языка Python.

Цель: исследовать процесс установки и базовые возможности языка Python версии 3.x.

Порядок выполнения работы:

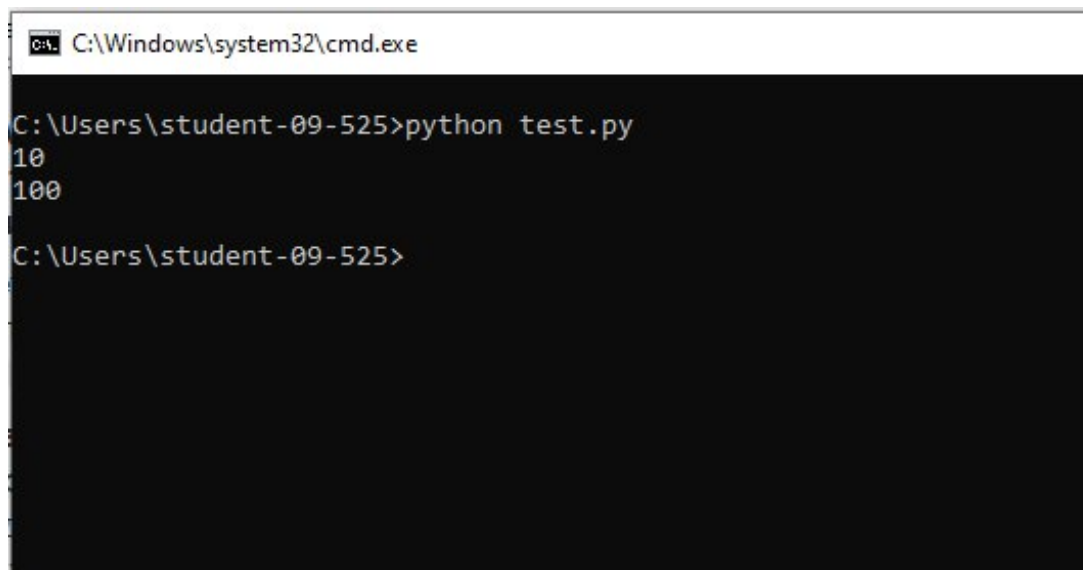
1. Последовательно установили на устройство Python, Anaconda и PyCharm, после чего проверили работоспособность.

```
SyntaxError: unterminated string literal (detected at line 1)
>>> print("Hello, world!")
Hello, world!
>>> exit_
```

Рис.1 — Проверка работоспособности Python

[illegible]

Рис. 2 — Проверка работоспособности Anaconda



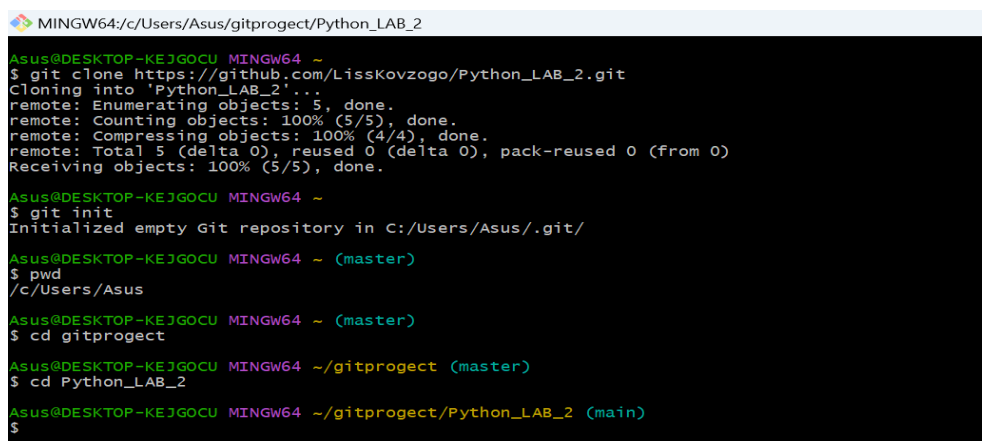
```
C:\Windows\system32\cmd.exe

C:\Users\student-09-525>python test.py
10
100

C:\Users\student-09-525>
```

Рис. 3 —Проверка работоспособности PyCharm

4. Создали новый репозиторий и клонировали их.



```
MINGW64/c:/Users/Asus/gitproject/Python_LAB_2

Asus@DESKTOP-KEJGOCU MINGW64 ~
$ git clone https://github.com/LissKovzogo/Python_LAB_2.git
Cloning into 'Python_LAB_2'...
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 (from 0)
Receiving objects: 100% (5/5), done.

Asus@DESKTOP-KEJGOCU MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/Asus/.git/

Asus@DESKTOP-KEJGOCU MINGW64 ~ (master)
$ pwd
/c:/Users/Asus

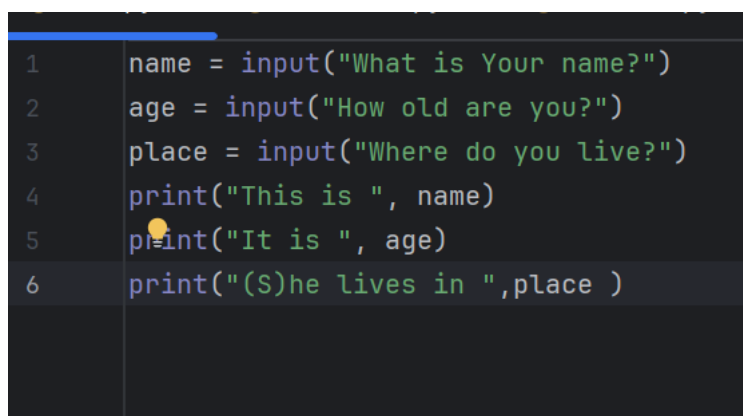
Asus@DESKTOP-KEJGOCU MINGW64 ~ (master)
$ cd gitproject

Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject (master)
$ cd Python_LAB_2

Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$
```

Рис. 4 — Клонирование репозитория

5. В соответствии с заданиями написали код и сохранили в файлы user.py, arithmetic.py,numbers.py,individul.py,individual2.py.



```
1 name = input("What is Your name?")
2 age = input("How old are you?")
3 place = input("Where do you live?")
4 print("This is ", name)
5 print("It is ", age)
6 print("(S)he lives in ",place )
```

Рис. 5 — user.py

```

a = input("Решите уравнение: 4 * 100 - 54 = ")
if a == "346":
    print("Верно")
else:
    print("Неверно")

```

Рис. 6 — arithmetic.py

```

firstf = int(input("Первое число "))
second = int(input("Второе число "))
third = int(input("Третье число "))
fourth = int(input("Четвёртое число "))
f = firstf + second
s = third + fourth
print(f'{{(f/s):.2f}}')

```

Рис. 7 — numbers.py

```

firstf = int(input("Первое число "))
second = int(input("Второе число "))
third = int(input("Третье число "))
fourth = int(input("Четвёртое число "))
f = firstf + second
s = third + fourth
print(f'{{(f/s):.2f}}')

```

Рис. 8 — individul.py

```

firstf = int(input("Первое число "))
second = int(input("Второе число "))
third = int(input("Третье число "))
fourth = int(input("Четвёртое число "))
f = firstf + second
s = third + fourth
print(f'{{(f/s):.2f}}')

```

Рис. 9 — individual2.py

6. Произвели коммит всех созданных файлов и выполнили слияние веток.

```
Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$ git add
warning: in the working copy of '.idea/inspectionProfiles/profiles_settings.xml', LF will be replaced by CRLF the next time Git touches it
Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .idea/.gitignore
    new file:   .idea/Python_LAB_2.iml
    new file:   .idea/inspectionProfiles/profiles_settings.xml
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/vcs.xml
    new file:   arithmetic.py
    new file:   individual1.py
    new file:   individual2.py
    new file:   numbers.py
    new file:   tst.py
    new file:   user.py

Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$ git commit -m"Commit python"
[main 075ef05] Commit python
12 files changed, 88 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/Python_LAB_2.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 arithmetic.py
create mode 100644 individual1.py
create mode 100644 individual2.py
create mode 100644 numbers.py
create mode 100644 tst.py
create mode 100644 user.py
Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$
```

Рис. 10 — Сохранение обновлений репозитория

7. Отправили изменения в удалённый репозиторий.

```
Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$ git push -u origin main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 20 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (15/15), 2.51 KiB | 856.00 KiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/LissKovzogo/Python_LAB_2.git
   e07ec13..075ef05  main -> main
branch 'main' set up to track 'origin/main'.

Asus@DESKTOP-KEJGOCU MINGW64 ~/gitproject/Python_LAB_2 (main)
$ git push
Everything up-to-date
```

Рис. 11 — Отправление на сервер GitHub

Адрес
созданного
репозитория:
https://github.com/LissKovzogo/Python_LAB_2.git

Контрольные вопросы:

1. Основные этапы установки Python

Windows:

- Скачать установщик с python.org
- Запустить установщик, отметить "Add Python to PATH"
- Выбрать опцию "Install Now" или "Customize installation"
- Завершить установку
- Проверить в командной строке: `python --version`

Linux:

- Для Ubuntu/Debian: `sudo apt update && sudo apt install python3`

`python3-pip`

- Для CentOS/RHEL: `sudo yum install python3`
- Для Fedora: `sudo dnf install python3`
- Проверить: `python3 --version`

2. Отличие Anaconda от официального Python

Anaconda:

- Включает Python + 1500+ популярных научных пакетов
- Собственный менеджер пакетов `conda`
- Упрощенное управление виртуальными окружениями
- Готовое окружение для data science

Официальный Python:

- Только интерпретатор Python
- Менеджер пакетов `pip`
- Требуется ручная установка дополнительных библиотек

3. Проверка работоспособности Anaconda

`bash`

`conda --version`

`conda list`

`python --version`

`jupyter notebook`

4. Настройка интерпретатора в PyCharm

File → Settings → Project → Python Interpreter

Settings → Add → Выбрать нужный интерпретатор

Можно выбрать: System Python, Virtualenv, Conda, Docker

5. Запуск программы в PyCharm

Правой кнопкой по файлу → Run 'filename'

6. Режимы работы Python

Интерактивный режим:

Команды выполняются сразу после ввода

Запуск: python или python3

Подходит для тестирования и экспериментов

Пакетный режим:

Выполнение программы из файла

Запуск: python script.py

Для полноценных приложений

7. Динамическая типизация

Python определяет типы переменных во время выполнения, а не компиляции:

```
python
```

```
x = 5    # int
```

```
x = "hello" # str - без ошибки
```

8. Основные типы данных

Числовые: int, float, complex, bool

Последовательности: str, list, tuple, range

Множества: set, frozenset

Словари: dict

Бинарные: bytes, bytearray, memoryview

NoneType: None

9. Создание объектов и переменных

Создание объекта:

```
python
```

`x = 5` *# Создается объект int со значением 5*

10. Список ключевых слов

python

`import keyword`

`print(keyword.kwlist)`

11. Функции `id()` и `type()`

`id()` - возвращает уникальный идентификатор объекта

`type()` - возвращает тип объекта

12. Изменяемые vs неизменяемые типы

Неизменяемые (immutable):

`int`, `float`, `str`, `tuple`, `frozenset`

Нельзя изменить после создания

Изменяемые (mutable):

`list`, `dict`, `set`, `bytearray`

Можно изменять содержимое

13. Операции деления

`/` - обычное деление (возвращает `float`)

`//` - целочисленное деление (возвращает `int`)

`%` - остаток от деления

14. Комплексные числа

python

`z = 3 + 4j`

`print(z.real)` *# 3.0*

`print(z.imag)` *# 4.0*

`print(z.conjugate())` *# (3-4j)*

15. Модули `math` и `cmath`

`math` - математические функции для вещественных чисел:

python

`import math`


```
math.sqrt(16) # 4.0
```

```
math.sin(math.pi/2) # 1.0
```

cmath - для комплексных чисел:

```
python
```

```
import cmath
```

```
cmath.sqrt(-1) # 1j
```

16. Параметры sep и end в print()

sep - разделитель между аргументами (по умолчанию пробел)

end - что печатать в конце (по умолчанию '\n')

```
python
```

```
print(1, 2, 3, sep='-', end='!') # 1-2-3!
```

17. Форматирование строк

Метод format():

```
python
```

```
"{ } {}".format("Hello", "World")
```

```
"{1} {0}".format("World", "Hello")
```

f-строки (Python 3.6+):

```
python
```

```
name = "Alice"
```

```
age = 25
```

```
f"Name: {name}, Age: {age}"
```

Другие способы:

%-форматирование: "Name: %s" % name

Конкатенация: "Hello " + name

18. Ввод числовых переменных

```
python
```

```
# Целое число
```

```
x = int(input("Введите целое число: "))
```

```
# Вещественное число
y = float(input("Введите вещественное число: "))

# С обработкой ошибок
try:
    num = int(input("Введите число: "))
except ValueError:
    print("Это не число!")
```