

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

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Магомедов Имран Борисович  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., кандидат технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

## Тема: Основы языка Python

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

### Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.


## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


---

*Required fields are marked with an asterisk (\*).*

**Owner \***


 **MarcusPlay** ▾

**Repository name \***


 **BSE\_Python\_1** is available.

Great repository names are short and memorable. Need inspiration? How about **solid-guacamole** ?

**Description** (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

**Initialize this repository with:**

☒ **Add a README file**


This is where you can write a long description for your project. [Learn more about READMEs.](#)


**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  `main` as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (main)
$ git branch
  develop
* main
```

6. Решите следующие задачи с помощью языка программирования Python3 и IDE PyCharm:

7. 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 live in `место\_жительства`"

```
1 # /usr/bin/env python3
2
3 def main(name, age, place):
4     print(f"This is {name}")
5     print(f"It is {age}")
6     print(f"(S)he live in {place}")
7
8
9 if __name__ == "__main__":
10     name = input("What is your name? ")
11     age = int(input("How old are you? "))
12     place = input("Where are you live? ")
13     main(name, age, place)
```

Вместо имя, возраст, место жительства должны быть данные, введенные пользователем. Примечание: можно писать фразы на русском языке, но, если вы планируете стать профессиональным программистом, привыкайте к английскому.

8. 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.

```
1 # /usr/bin/env python3
2
3 def main(answer):
4     print("Solve the example 4 * 100 - 54")
5     print(f"Your answer: {answer}")
6     print("Correct answer: 356")
7
8
9 if __name__ == "__main__":
10     answer = int(input("Enter your answer: "))
11     main(answer)
```

9. 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.

```
1 # /usr/bin/env python3
2
3 def main(args: list):
4     answer = (int(args[0]) + int(args[1])) / (int(args[2]) + int(args[3]))
5     print(f"({args[0]} + {args[1]}) / ({args[2]} + {args[3]}) = {answer: .2f}")
6
7
8 if __name__ == "__main__":
9     inp = input("Enter 4 numbers: ")
10     args = inp.split(' ')
11     main(args)
```

10. Напишите программу (файл *individual.py*) для решения индивидуального задания. Вариант индивидуального задания уточните у преподавателя.

```
1 # /usr/bin/env python3
2 # Известна стоимость монитора, системного блока, клавиатура и мышь.
3 # Сколько будут стоить 3 компьютера из этих элементов? N компьютеров?
4
5 def main(N):
6     ...monitor = 15_000
7     ...system_unit = 45_000
8     ...keyboard = 5_000
9     ...mouse = 2_500
10    ...print("Total price:", (N * (monitor + system_unit + keyboard + mouse)))
11
12
13 if __name__ == "__main__":
14     ...N = int(input("Enter count of PC: "))
15     ...main(N)
```

11. Выполните коммит файлов *user.py*, *arithmetic.py*, *numbers.py* и *individual.py* в репозиторий git в ветку для разработки.

```
Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   arithmetic.py
        modified:   hard_task.py
        modified:   individual.py
        modified:   number.py
        modified:   user.py

no changes added to commit (use "git add" and/or "git commit -a")

Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (develop)
$ git add .

Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (develop)
$ git commit -m "modified files"
[develop 2827a13] modified files
6 files changed, 60 insertions(+), 41 deletions(-)
```

12. Выполните слияние ветки для разработки с веткой *main*.

```

Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (main)
$ git merge develop
Updating f3b9b29..2827a13
Fast-forward
 .gitignore      | 1 +
 .vscode/settings.json | 6 +++++
 arithmetic.py    | 11 ++++++++
 hard_task.py     | 22 ++++++++
 individual.py    | 15 ++++++++
 number.py        | 11 ++++++++
 user.py          | 13 ++++++++
7 files changed, 79 insertions(+)
create mode 100644 .vscode/settings.json
create mode 100644 arithmetic.py
create mode 100644 hard_task.py
create mode 100644 individual.py
create mode 100644 number.py
create mode 100644 user.py

```

13. Отправьте сделанные изменения на сервер GitHub.

```

Admin@HOME-PC MINGW64 ~/Desktop/Work/BSE_Python_1 (main)
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarcusPlay/BSE_Python_1.git
 f3b9b29..2827a13  main -> main

```

## Индивидуальное задание

### Вариант 12

Известна стоимость монитора, системного блока, клавиатуры и мыши.  
 Сколько будут стоить 3 компьютера из этих элементов? N компьютеров?

```

1 # /usr/bin/env python3
2 # Известна стоимость монитора, системного блока, клавиатура и мышь.
3 # Сколько будут стоить 3 компьютера из этих элементов? N компьютеров?
4
5 def main(N):
6     ....monitor = 15_000
7     ....system_unit = 45_000
8     ....keyboard = 5_000
9     ....mouse = 2_500
10    ....print("Total price:", (N * (monitor + system_unit + keyboard + mouse)))
11
12
13 if __name__ == "__main__":
14     ....N = int(input("Enter count of PC: "))
15     ....main(N)

```

```

PS C:\Users\Admin\Desktop\Work\BSE_Python_1> & "C:/Program Files/Python312/python.exe"
Enter count of PC: 3
Total price: 202500

```

## Задание повышенной сложности

### Вариант 4

Даны целое число  $k$  ( $1 \leq k \leq 190$ ) и последовательность цифр 10111213...9899, в которой выписаны подряд все двузначные числа.

Определить:

1. номер пары цифр, в которую входит  $k$ -я цифра;
2. двузначное число, образованное парой цифр, в которую входит  $k$ -я цифра;
3.  $k$ -ю цифру, если известно, что:  $k$ -четное число;  $k$ -нечетное число.

```

# /usr/bin/env python3
# Даны целое число  $k$  ( $1 \leq k \leq 190$ ) и последовательность цифр 10111213...9899,
# в которой выписаны подряд все двузначные числа. Определить:
# 1. номер пары цифр, в которую входит  $k$ -я цифра;
# 2. двузначное число, образованное парой цифр, в которую входит  $k$ -я цифра;
# 3.  $k$ -ю цифру, если известно, что:  $k$ -четное число;  $k$ -нечетное число.

def find_name(k):
    numbers = [x for x in range(10, 100)]

    if k % 2 == 0:
        print("The number of a pair of digits:", k//2 - 1)
        print("Two-digit number:", numbers[k//2 - 1])
        print("k-th digit:", numbers[k//2 - 1] % 10)
    else:
        print("The number of a pair of digits:", k//2)
        print("Two-digit number:", numbers[k//2])
        print("k-th digit:", numbers[k//2] // 10)

if __name__ == "__main__":
    k = input("enter k (1 <= k <= 190): ")
    find_name(k=int(k))

```

## Контрольные вопросы

### 1. Установка Python в Windows и Linux:

- **Windows:**

- Скачайте исполняемый файл установки Python с официального сайта (python.org).
- Запустите установочный файл.
- Выберите "Добавить Python в PATH" во время установки.
- Следуйте инструкциям установщика.

- **Linux:**

- Многие дистрибутивы Linux уже поставляются с Python. большинстве случаев вы можете установить Python с помощью менеджера пакетов, например, apt-get (для Debian/Ubuntu) или yum (для CentOS).



- Пример установки на Ubuntu: `sudo apt-get install python3`.

## **2. Отличие Anaconda от официального Python:**

- Anaconda — это пакет, который включает в себя Python и множество научных библиотек для анализа данных и машинного обучения.
- Официальный Python — это базовый интерпретатор без дополнительных библиотек.
- Anaconda предназначена в основном для научных и аналитических задач.

## **3. Проверка работоспособности Anaconda:**

- Запустите командную строку/терминал и введите `conda list` для просмотра установленных пакетов.
- Запустите Python, например, введя `python` в командной строке, и убедитесь, что он успешно запускается.

## **4. Задание интерпретатора в PyCharm:**

- Откройте настройки проекта в PyCharm.
- Перейдите в раздел Project: <ваш проект> Python Interpreter.
- Выберите интерпретатор Python, который вы хотите использовать для проекта.

## **5. Запуск программы в PyCharm:**

- Откройте ваш проект в PyCharm.
- Откройте файл с кодом.
- Нажмите правой кнопкой мыши на файле и выберите "Run" или "Run File".

## **6. Интерактивный и пакетный режимы Python:**

- Интерактивный режим - взаимодействие с Python в реальном времени, ввод команд в интерпретатор.
- Пакетный режим - выполнение скриптов Python из файлов.

## **7. Динамическая типизация в Python:**

- Переменные в Python не связаны строго с определенными типами данных и могут изменять свой тип в процессе выполнения программы.

## **8. Основные типы в Python:**

- Числа (int, float, complex)
- Строки (str)
- Списки (list)
- Кортежи (tuple)
- Множества (set)
- Словари (dict)
- Булевы значения (bool)
- None (отсутствие значения)

## **9. Создание объектов в памяти:**

- Объекты создаются при присваивании значений переменным. Они хранятся в памяти компьютера.
- Процесс объявления новых переменных включает создание имени и связывание его с объектом в памяти.

## **10. Получение списка ключевых слов в Python:**

- Используйте модуль keyword:

```
import keyword  
  
print(keyword.kwlist)
```

## **11. Функции id() и type():**

- id() возвращает уникальный идентификатор объекта в памяти.
- type() возвращает тип объекта.

## **12. Изменяемые и неизменяемые типы в Python:**

- Изменяемые (mutable) типы, например, списки и словари, могут быть изменены после создания.
- Неизменяемые (immutable) типы, например, числа и строки, не могут быть изменены после создания.

### **13. Операции деления и целочисленного деления:**

- / - обычное деление, всегда возвращает число с плавающей запятой.
- // - целочисленное деление, возвращает целое число (округленное вниз).

### **14. Работа с комплексными числами:**

- Python поддерживает комплексные числа, представленные как  $a + bj$ , где  $a$  и  $b$  - действительные числа,  $j$  - мнимая единица.

### **15. Библиотека math и cmath:**

- math - библиотека для работы с математическими функциями (например, sin, cos, sqrt).
- cmath - библиотека для комплексной математики.

### **16. Параметры sep и end в print():**

- sep определяет разделитель между аргументами print.
- end определяет символ, добавляемый в конце вывода.

### **17. Метод format() и форматирование строк:**

- format() используется для форматирования строк, заменяя заполнители {} значениями.
- Существуют также f-строки, которые представляют улучшенный способ форматирования строк в Python 3.6 и выше.

### **18. Ввод значений с консоли:**

- Используйте input() для ввода строк с консоли. Пример:  
name = input("Введите ваше имя: ")  
age = int(input("Введите ваш возраст: "))