

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 №16
дисциплины «Основы программной инженерии»

Выполнила:
Панюкова Ксения Юрьевна
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., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучила теоретический материал работы

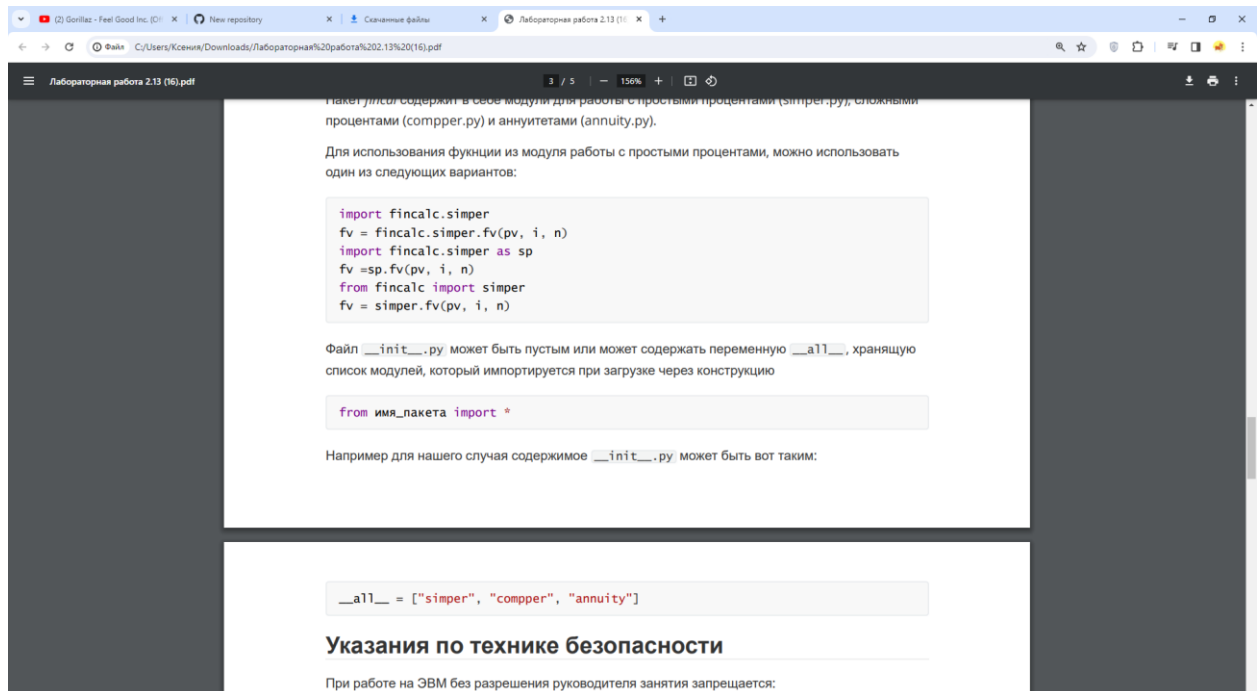



Рисунок 1.1 – Изучение материала для лабораторной работы

2. Создала 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 *
 MrFinichck / Repository name *
ModulesAndPackages
✔ ModulesAndPackages is available.

Great repository names are short and memorable. Need inspiration? How about **fuzzy-funicular** ?

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

.gitignore template: Python

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

Choose a license

License: MIT License

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

 You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

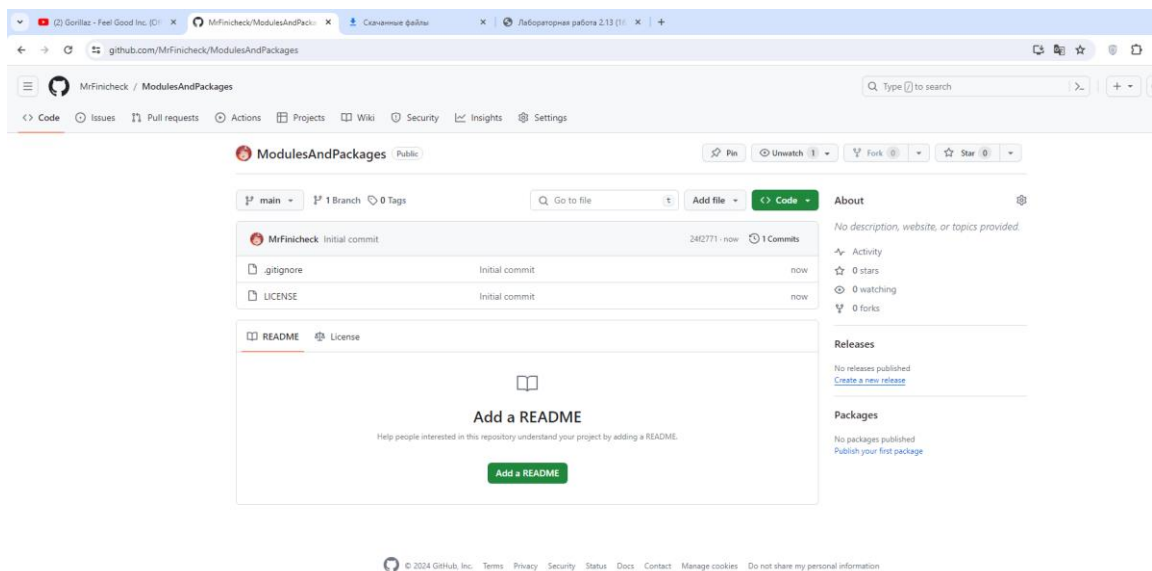


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

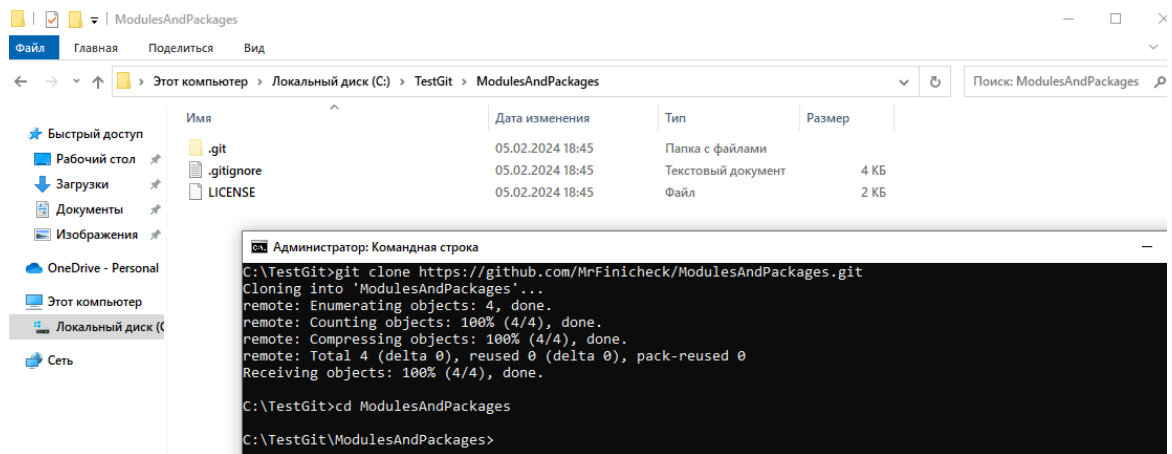


Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнила файл .gitignore необходимыми правилами для работы с IDE PyCharm

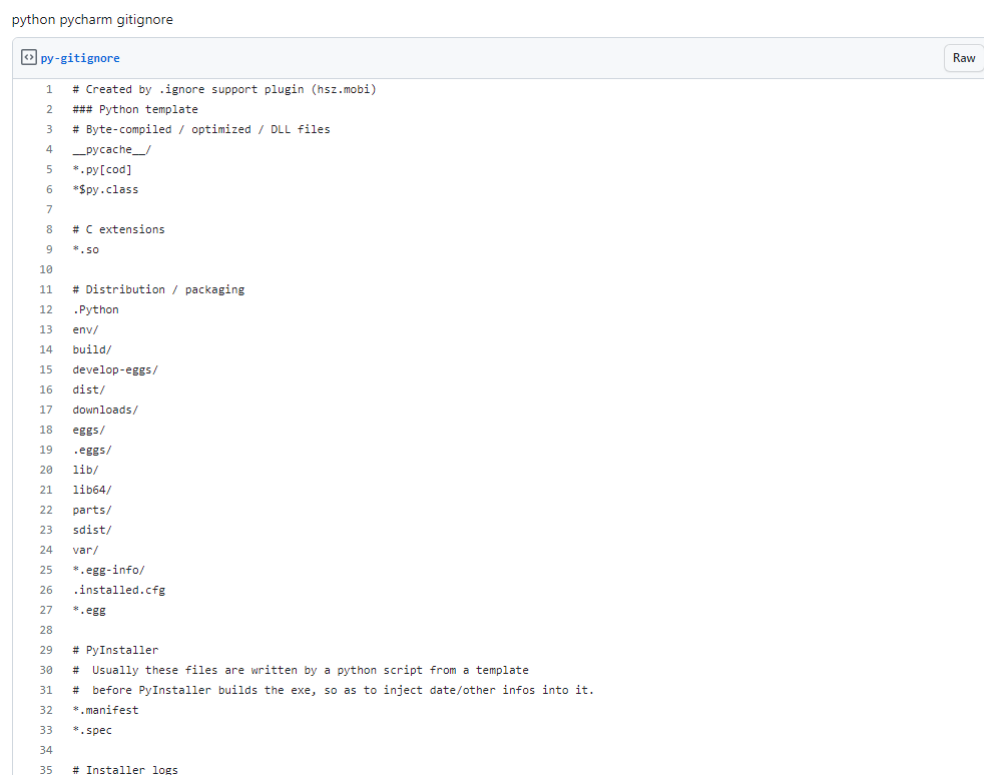


Рисунок 4.1 – .gitignore для IDE PyCharm

5. Организовала свой репозиторий в соответствии с моделью ветвления git-flow

```

C:\TestGit\ModulesAndPackages>git branch develop

C:\TestGit\ModulesAndPackages>git checkout develop
Switched to branch 'develop'

C:\TestGit\ModulesAndPackages>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinicheck/ModulesAndPackages/pull/new/develop
remote:
To https://github.com/MrFinicheck/ModulesAndPackages.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

```

Рисунок 5.1 – Создание ветки develop от ветки main

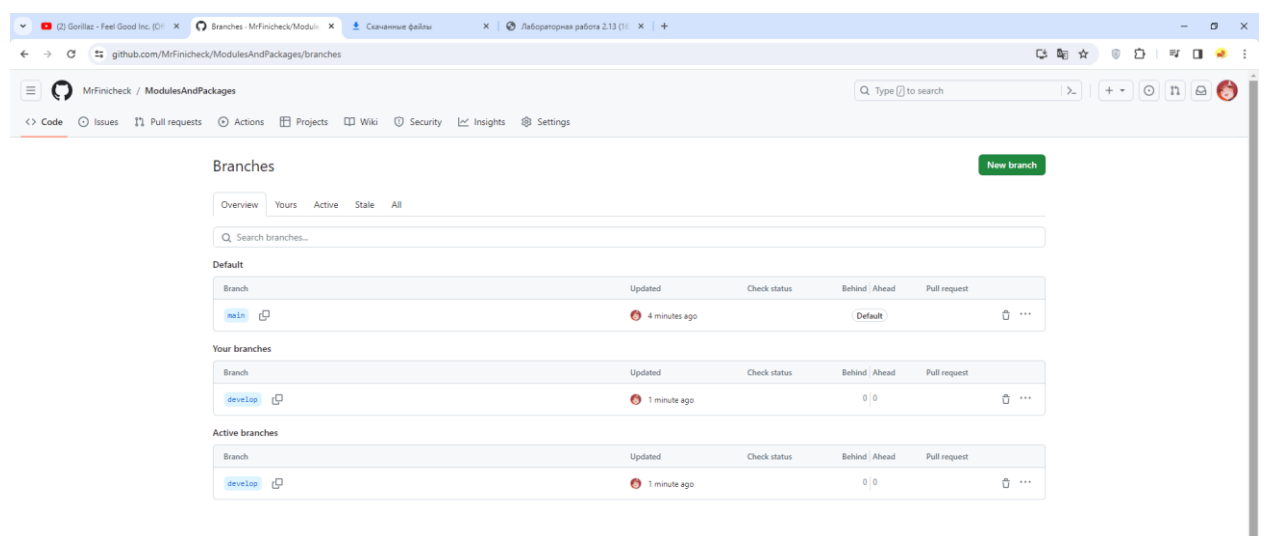


Рисунок 5.2 – Ветка develop на GitHub

6. Создала проект PyCharm в папке репозитория

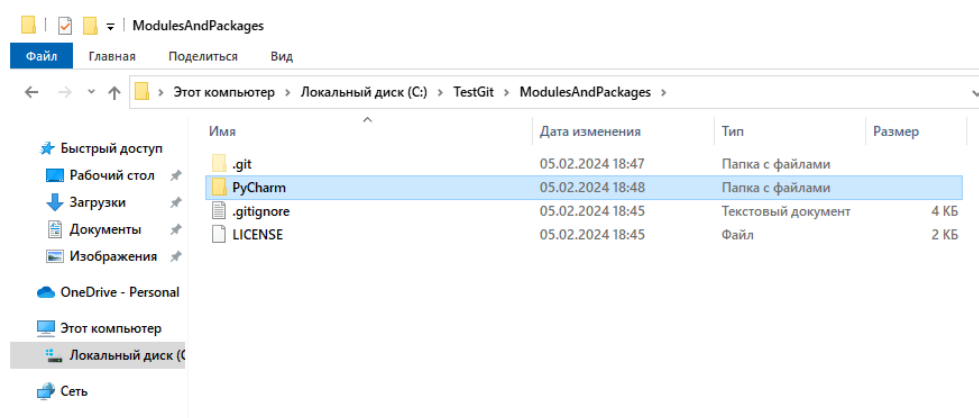
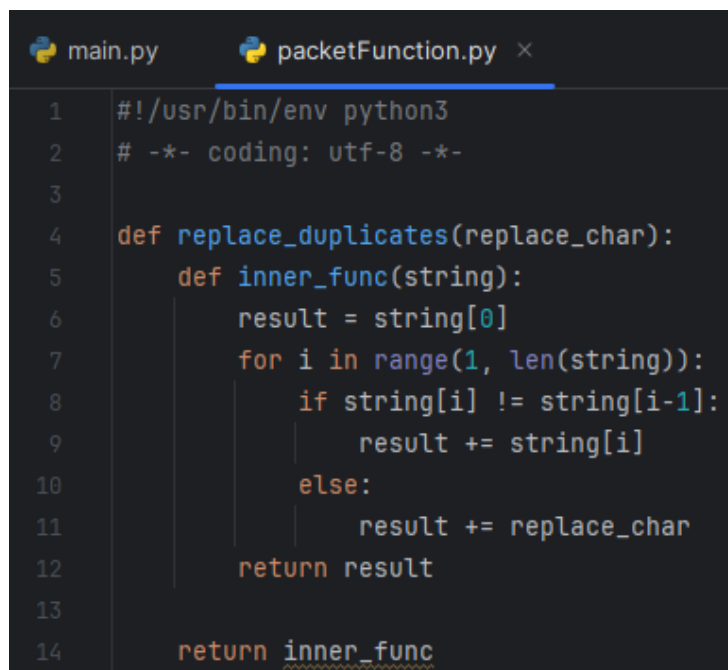


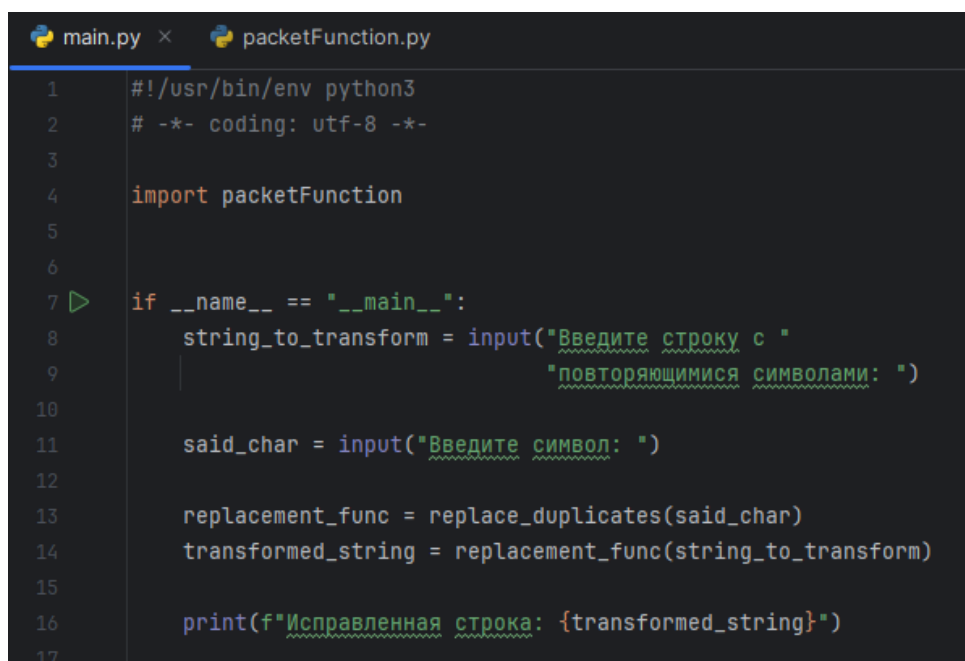
Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Выполнила индивидуальные задания. Привела в отчете скриншоты работы программ решения индивидуального задания.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def replace_duplicates(replace_char):
5      def inner_func(string):
6          result = string[0]
7          for i in range(1, len(string)):
8              if string[i] != string[i-1]:
9                  result += string[i]
10             else:
11                 result += replace_char
12             return result
13
14     return inner_func
```

Рисунок 7.1 – Код программы с функциями из лабораторной № 2.11



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import packetFunction
5
6
7  if __name__ == "__main__":
8      string_to_transform = input("Введите строку с "
9                                  "повторяющимися символами: ")
10
11     said_char = input("Введите символ: ")
12
13     replacement_func = replace_duplicates(said_char)
14     transformed_string = replacement_func(string_to_transform)
15
16     print(f"Исправленная строка: {transformed_string}")
17
```

Рисунок 7.2 – Основная программа, с импортированием функций из лабораторной № 2.11

```


    case 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить личность;")
        print("list - вывести список личностей;")
        print("select <месяц> - запросить личностей с этим месяцем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")

    case _:
        print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Рисунок 7.3 – Основная программа, с использованием функций пакета из лабораторной № 2.8



```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from infoPack import *

```

Рисунок 7.4 – Импортирование всех функций пакета из лабораторной № 2.8

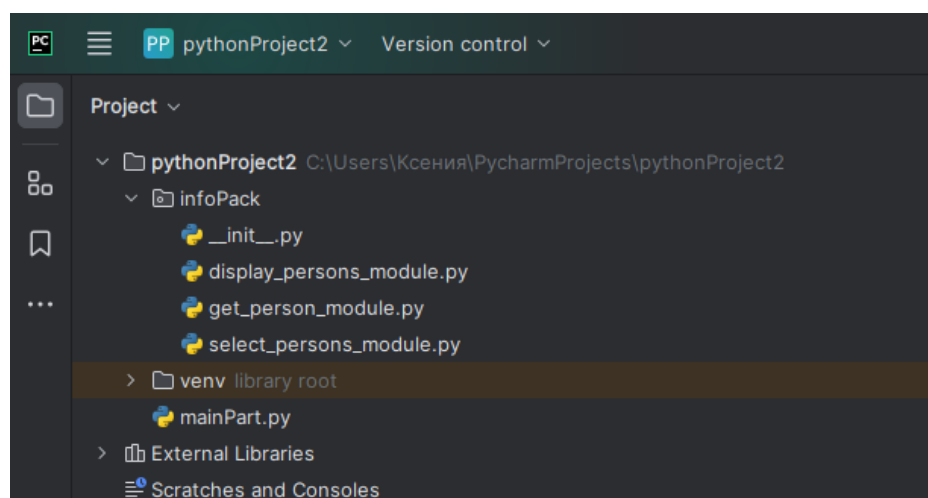


Рисунок 7.5 – Функции, занесённые в пакет из лабораторной № 2.8

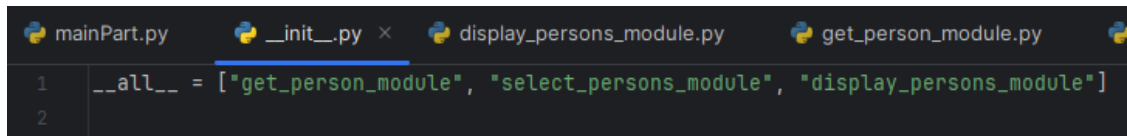


Рисунок 7.6 – Переменная `__all__`, хранящая список модулей, который импортируется при загрузке через конструкцию

8. Зафиксировала сделанные изменения в репозитории.

```
C:\TestGit\ModulesAndPackages>git add PyCharm

C:\TestGit\ModulesAndPackages>git commit -m "add files"
[develop 3956d4f] add files
 7 files changed, 188 insertions(+)
 create mode 100644 PyCharm/Task1/main.py
 create mode 100644 PyCharm/Task1/packetFunction.py
 create mode 100644 PyCharm/Task2/infoPack/__init__.py
 create mode 100644 PyCharm/Task2/infoPack/display_persons_module.py
 create mode 100644 PyCharm/Task2/infoPack/get_person_module.py
 create mode 100644 PyCharm/Task2/infoPack/select_persons_module.py
 create mode 100644 PyCharm/Task2/mainPart.py
```

Рисунок 8.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Что является модулем языка Python?

Под модулем в Python понимается файл с расширением `.py`. Модули предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т. п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы. Стоит заметить, что модули могут быть написаны не только на языке Python, но и на других языках (например C).

2. Какие существуют способы подключения модулей в языке Python?

Самый простой способ импортировать модуль в Python это воспользоваться конструкцией: `import имя_модуля`. За один раз можно импортировать сразу несколько модулей, для этого их нужно перечислить через запятую после слова `import`. Если вы хотите задать псевдоним для модуля в вашей программе, можно воспользоваться вот таким синтаксисом: `import`

имя_модуля as новое_имя. Используя любой из вышеперечисленных подходов, при вызове функции из импортированного модуля, вам всегда придется указывать имя модуля (или псевдоним). Для того, чтобы этого избежать делайте импорт через конструкцию `from ... import...`

3. Что является пакетом языка Python?

Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл `__init__.py`. Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку).

4. Каково назначение файла `__init__.py` ?

Файл `__init__.py` может быть пустым или может содержать переменную `__all__`, хранящую список модулей, который импортируется при загрузке через конструкцию.

5. Каково назначение переменной `__all__` файла `__init__.py` ?

Переменная `__all__` в файле `__init__.py` в Python хранит список модулей, который импортируется при загрузке через конструкцию