

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 2.13
по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-22-1
Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2024

Тема: Модули и пакеты.

Цель работы: приобретение навыков по работе с модулями и пакетами языка программирования Python версии 3.x.

Ход выполнения работы:

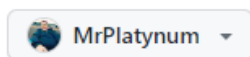
1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и язык программирования Python:

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 *



Repository name *

ProgrammEngineering16

✓ ProgrammEngineering16 is available.

Great repository names are short and memorable. Need inspiration? How about [musical-computing-machine](#) ?

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 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering16.git
Cloning into 'ProgrammEngineering16'...
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.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер

```
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Выполнить индивидуальное задание лабораторной работы 2.11, оформив все функции программы в виде отдельного модуля. Разработанный модуль должен быть подключен в основную программу с помощью одного из вариантов команды `import`. Номер варианта уточнить у преподавателя:

Листинг индивидуального задания 1:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from replace_chars_module import replace_repeated_chars
```

```

if __name__ == "__main__":
    replace_char = input('Введите символ, на который необходимо заменить
повторяющиеся символы: ')
    replace_func = replace_repeated_chars(replace_char)

    input_string = input('Введите строку: ')

    result = replace_func(input_string)
    print(result)

```

Листинг модуля replace_chars_module:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

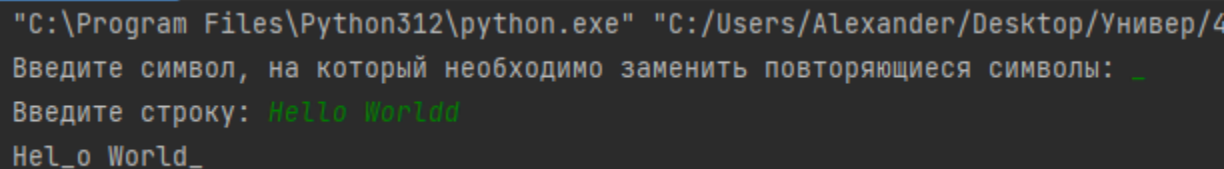
def replace_repeated_chars(replace_char):
    def inner_function(input_string):
        result = ''
        prev_char = ''

        for char in input_string:
            if char != prev_char:
                result += char
                prev_char = char
            else:
                result += replace_char

        return result

    return inner_function

```



```

"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4
Введите символ, на который необходимо заменить повторяющиеся символы: _
Введите строку: Hello World
Hel_o World_

```

Рисунок 5 – Индивидуальное задание №1

3. Выполнить индивидуальное задание лабораторной работы 2.8, оформив все классы программы в виде отдельного пакета. Разработанный пакет должен быть подключен в основную программу с помощью одного из вариантов команды `import`. Настроить соответствующим образом переменную `__all__` в файле `__init__.py` пакета. Номер варианта уточнить у преподавателя.

Листинг индивидуального задания 2:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from train_system import add_train, list_trains, select_trains,
display_help

```

```

def main():
    """Основная функция управления программой."""
    trains = []

    while True:
        command = input(">>> ").lower()

        match command:
            case 'exit':
                break
            case 'add':
                add_train(trains)
            case 'list':
                list_trains(trains)
            case 'select':
                selected = select_trains(trains, input("Введите время для
поиска поездов (в формате ЧЧ:ММ): "))
                if isinstance(selected, list):
                    list_trains(selected)
                else:
                    print(selected)
            case 'help':
                display_help()
            case _:
                print(f"Неизвестная команда {command}")

if __name__ == '__main__':
    main()

```

Листинг модуля train_function:

```

from datetime import time

def add_train(trains):
    """Добавляет информацию о поезде."""
    destination = input("Название пункта назначения: ")
    train_number = input("Номер поезда: ")
    departure_time_str = input("Время отправления (в формате ЧЧ:ММ): ")

    hours, minutes = map(int, departure_time_str.split(':'))

    departure_time = time(hours, minutes)

    train = {
        'название пункта назначения': destination,
        'номер поезда': train_number,
        'время отправления': departure_time,
    }

    trains.append(train)
    trains.sort(key=lambda x: x['название пункта назначения'])

def list_trains(trains):
    """Выводит список всех поездов."""
    line = f'+--{"-" * 35}+--{"-" * 15}+--{"-" * 25}+-'
    print(line)

```

```

    print(f"| {'Название пункта назначения':^35} | {'Номер поезда':^15} |  

{'Время отправления':^25} |")

    for train in trains:
        print(line)
        departure_time = train['время отправления'].strftime('%H:%M')
        print(
            f"| {train['название пункта назначения':^35} | {train['номер  

поезда':^15} | {departure_time:^25} |")
        print(line)

def select_trains(trains, search_time_str):
    """Выводит поезда, отправляющиеся после указанного времени."""
    found = False
    result = []

    hours, minutes = map(int, search_time_str.split(':'))

    search_time = time(hours, minutes)

    print(f"Поезда, отправляющиеся после {search_time}:")
    for train in trains:
        train_time = train['время отправления']
        if train_time > search_time:
            result.append(train)
            found = True

    if found:
        return result

    if not found:
        return "Нет поездов, отправляющихся после указанного времени."

def display_help():
    """Выводит справку о доступных командах."""
    print("Список команд:\n")
    print("add - добавить информацию о поезде;")
    print("list - вывести список всех поездов;")
    print("select <время> - вывести поезда, отправляющиеся после указанного  

времени;")
    print("exit - завершить работу с программой.")

```

```
"C:\Program Files\Python312\python.exe" "C:/Users/Alexander/Desktop/Универ/4 семестр/Основы программн
>>> add
Название пункта назначения: Moscow
Номер поезда: 123
Время отправления (в формате ЧЧ:ММ): 12:30
>>> add
Название пункта назначения: SPB
Номер поезда: 23
Время отправления (в формате ЧЧ:ММ): 13:00
>>> list
+-----+-----+-----+
| Название пункта назначения | Номер поезда | Время отправления |
+-----+-----+-----+
| Moscow | 123 | 12:30 |
+-----+-----+-----+
| SPB | 23 | 13:00 |
+-----+-----+-----+
>>> select
Введите время для поиска поездов (в формате ЧЧ:ММ): 12:30
Поезда, отправляющиеся после 12:30:00:
+-----+-----+-----+
| Название пункта назначения | Номер поезда | Время отправления |
+-----+-----+-----+
| SPB | 23 | 13:00 |
+-----+-----+-----+
>>>
```

Рисунок 6 – Индивидуальное задание №2

4. Зафиксируем сделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (develop)
$ git log --oneline
1039478 (HEAD -> develop) Добавление индивидуального задания 2
864943e Добавление индивидуального задания 1
ed9fa9c (origin/main, origin/HEAD, main) Initial commit

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (develop)
$ .....
```

Рисунок 7 – Коммиты ветки develop во время выполнения лабораторной работы

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (develop)
$ git checkout main
Switched to branch 'main'
M       .gitignore
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (main)
$ git merge develop
Updating ed9fa9c..1039478
Fast-forward
 individual1.py          | 13 ++++++
 individual2.py          | 34 ++++++
 replace_chars_module.py | 18 ++++++
 train_system/__init__.py | 1 +
 train_system/train_functions.py | 67 ++++++
 5 files changed, 133 insertions(+)
 create mode 100644 individual1.py
 create mode 100644 individual2.py
 create mode 100644 replace_chars_module.py
 create mode 100644 train_system/__init__.py
 create mode 100644 train_system/train_functions.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (main)
$

```

Рисунок 8 – Слияние ветки develop в ветку main

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 2.66 KiB | 2.66 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MrPlatynum/ProgrammEngineering16.git
   ed9fa9c..1039478  main -> main

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering16 (main)
$

```

Рисунок 9 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Что является модулем языка в Python?

В Python модуль - это файл с расширением `.py`, содержащий код на языке Python. Модуль может содержать функции, классы и переменные, и его можно использовать для организации кода и повторного использования.

2. Какие существуют способы подключения модулей в языке Python?

Есть несколько способов подключения модулей:

Использование ключевого слова `import` (например, `import module_name`).

Использование ключевого слова `from` для импорта конкретных элементов из модуля (например, `from module_name import function_name`).

Использование псевдонимов с ключевым словом `as` (например, `import module_name as alias`).

3. Что является пакетом языка Python?

Пакет в Python - это способ организации модулей в иерархическую структуру внутри директории. Пакеты позволяют логически группировать связанный код для более удобного управления проектами.

4. Каково назначение файла `__init__.py`?

Файл `__init__.py` в директории Python указывает, что эта директория должна рассматриваться как пакет, а не просто коллекция модулей. Этот файл может быть пустым, но его наличие делает директорию пригодной для использования в качестве пакета.

5. Каково назначение переменной `__all__` в файле `__init__.py`?

Переменная `__all__` в файле `__init__.py` используется для определения списка модулей, которые будут импортированы при использовании оператора `from package_name import *`. Если `__all__` не определен, будут импортированы все модули, кроме тех, чьи имена начинаются с подчеркивания. Если `__all__` определен, будут импортированы только модули, указанные в этом списке.