

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 практической работе №2.13**  
**Дисциплины «Программирование на Python»**

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

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

---

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических  
наук, доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

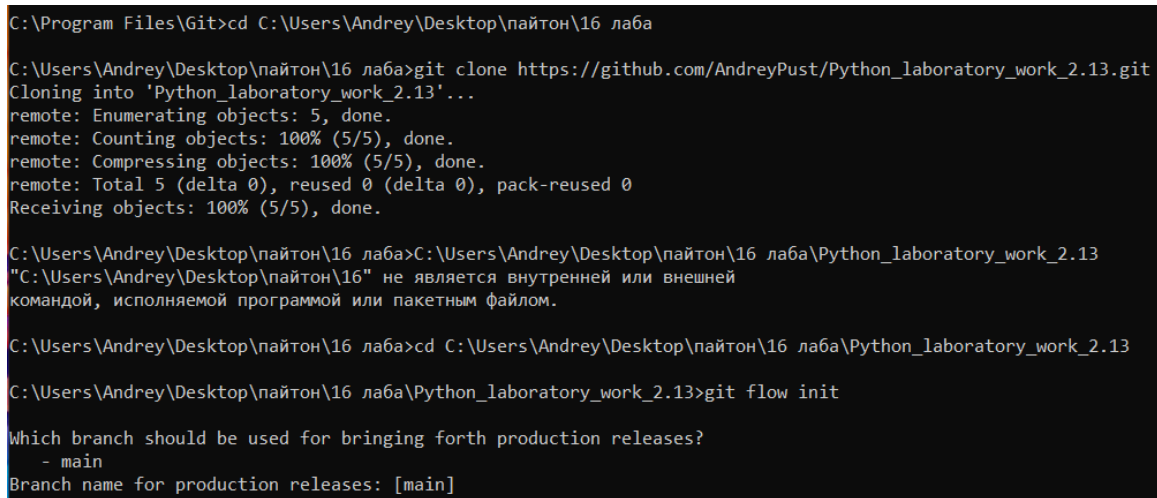
Ставрополь, 2023 г.

Тема: Модули и пакеты.

Цель: приобрести навыки по работе с модулями и пакетами языка программирования Python версии 3.x.

Ход работы:

Создание общедоступного репозитория на «GitHub», клонирование репозитория, редактирование файла «.gitignore», организация репозитория согласно модели ветвления «git-flow» (рис. 1).



```
C:\Program Files\Git>cd C:\Users\Andrey\Desktop\пайтон\16 лаба
C:\Users\Andrey\Desktop\пайтон\16 лаба>git clone https://github.com/AndreyPust/Python_laboratory_work_2.13.git
Cloning into 'Python_laboratory_work_2.13'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\Andrey\Desktop\пайтон\16 лаба>C:\Users\Andrey\Desktop\пайтон\16 лаба\Python_laboratory_work_2.13
"C:\Users\Andrey\Desktop\пайтон\16 лаба\Python_laboratory_work_2.13" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\Andrey\Desktop\пайтон\16 лаба>cd C:\Users\Andrey\Desktop\пайтон\16 лаба\Python_laboratory_work_2.13
C:\Users\Andrey\Desktop\пайтон\16 лаба\Python_laboratory_work_2.13>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
```

Рисунок 1 – Организация модели ветвления «git-flow».

Выполнение индивидуальных заданий:

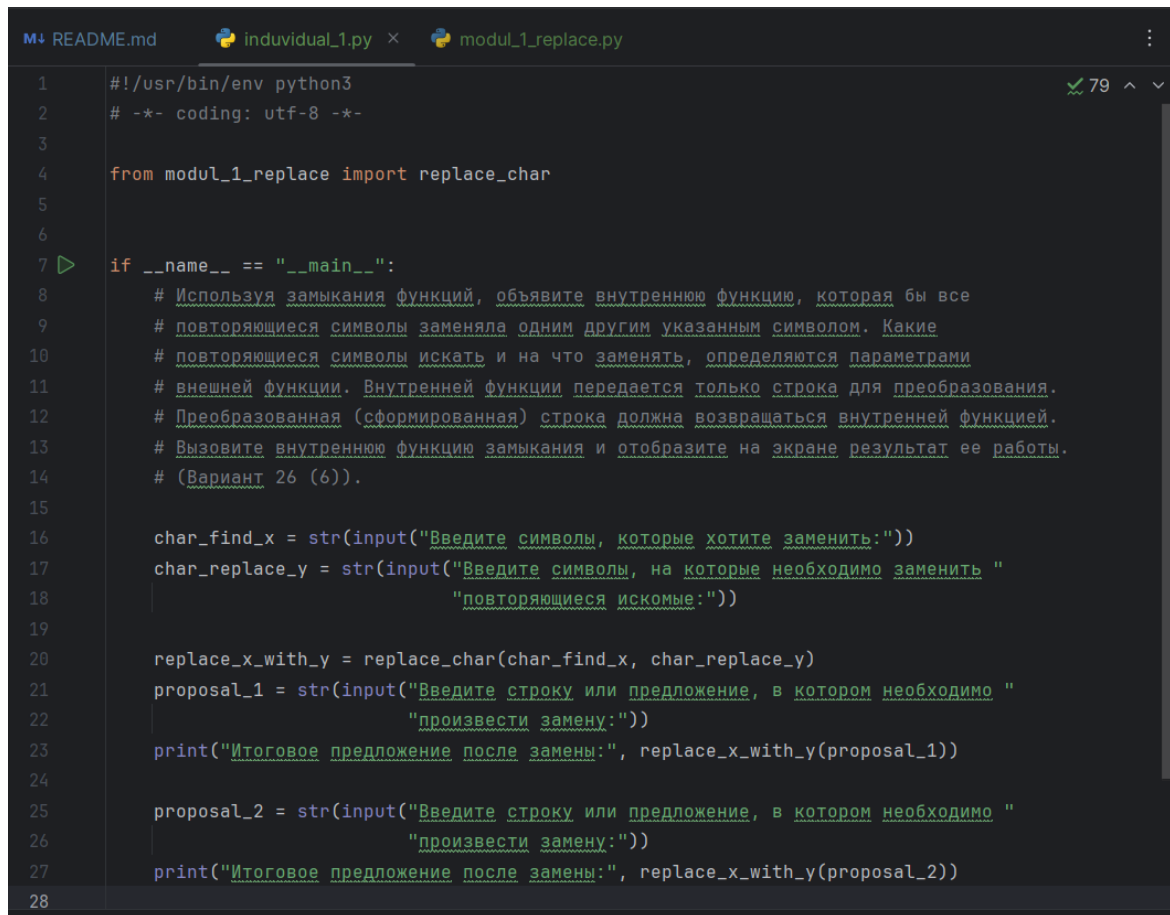
Задание 1.

Необходимо выполнить индивидуальное задание лабораторной работы 2.11, оформив все функции в нем в виде отдельного модуля. Разработанный модуль должен быть подключен в основную программу с помощью одного из вариантов команды «import».

Используя замыкания функций, объявите внутреннюю функцию, которая бы все повторяющиеся символы заменяла одним другим указанным символом. Какие повторяющиеся символы искать и на что заменять, определяются параметрами внешней функции. Внутренней функции передается только строка для преобразования. Преобразованная

(сформированная) строка должна возвращаться внутренней функцией. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы. (Вариант 26 (6)).

Код программы индивидуального задания, содержимое подключаемого отдельного модуля и результаты работы программы (рис. 2, 3, 4).



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  from modul_1_replace import replace_char
5
6
7  if __name__ == "__main__":
8      # Используя замыкания функций, объявите внутреннюю функцию, которая бы все
9      # повторяющиеся символы заменяла одним другим указанным символом. Какие
10     # повторяющиеся символы искать и на что заменять, определяются параметрами
11     # внешней функции. Внутренней функции передается только строка для преобразования.
12     # Преобразованная (сформированная) строка должна возвращаться внутренней функцией.
13     # Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.
14     # (Вариант 26 (6)).
15
16     char_find_x = str(input("Введите символы, которые хотите заменить:"))
17     char_replace_y = str(input("Введите символы, на которые необходимо заменить "
18                               "повторяющиеся искомые:"))
19
20     replace_x_with_y = replace_char(char_find_x, char_replace_y)
21     proposal_1 = str(input("Введите строку или предложение, в котором необходимо "
22                           "произвести замену:"))
23     print("Итоговое предложение после замены:", replace_x_with_y(proposal_1))
24
25     proposal_2 = str(input("Введите строку или предложение, в котором необходимо "
26                           "произвести замену:"))
27     print("Итоговое предложение после замены:", replace_x_with_y(proposal_2))
28
```

Рисунок 2 – Код программы индивидуального задания с подключением нужного объекта из другого модуля.

```

def replace_char(char_find, char_replace):
    """
    Функция производит замену последовательности искоемых символов (char_find)
    на один указанный в переданных аргументах символ (char_replace).
    Для того, чтобы функции можно было передать новое предложение или строку
    при тех же искоемых и меняемых символах, в функции используется замыкание,
    поэтому в дальнейшем, при вызове функции ей передается всего один аргумент
    - новое предложение для замены.
    """
    new *
    def in_func(proposal):
        result_proposal = ''
        prev_char = ''
        count = 0
        len_proposal = len(proposal)

        # Произведем замену последовательностей нужных символов.
        for char in proposal:
            if count == len_proposal - 1:
                break
            if char != char_find:
                result_proposal += char
            if char == char_find and char == prev_char and proposal[count + 1] != prev_char:
                result_proposal += char_replace

            prev_char = char
            count += 1
        return result_proposal

    return in_func

```

Рисунок 3 – Код программы подключаемого модуля.

```

C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\Desktop\пайтон\16 лаба\
Введите символы, которые хотите заменить:1
Введите символы, на которые необходимо заменить повторяющиеся искомые:-
Введите строку или предложение, в котором необходимо произвести замену:qqq111qqq111www11eee
Итоговое предложение после замены: qq-qqq-www-ee
Введите строку или предложение, в котором необходимо произвести замену:0001110001110001110000
Итоговое предложение после замены: 000-000-000-000

```

Рисунок 4 – Результаты работы программы индивидуального задания №1.

## Задание 2.

Необходимо 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». Необходимо правильным образом настроить переменную «\_all\_» в файле «\_init\_.py» пакета.

Необходимо использовать словарь, содержащий следующие ключи: название пункта назначения, номер поезда, время отправления. Написать

программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по времени отправления поезда; вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры; если таких поездов нет, выдать на дисплей соответствующее сообщение (Вариант 26 (7)).

Код программы индивидуального задания, содержимое подключаемого пакета включая файл «\_\_init\_\_.py» и результаты работы программы (рис. 5, 6, 7, 8, 9, 10).

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

import sys
from package_2_functions import *

if __name__ == '__main__':
    # Необходимо использовать словарь, содержащий следующие ключи: название
    пункта
    # назначения, номер поезда, время отправления. Написать программу,
    выполняющую
    # следующие действия: ввод с клавиатуры данных в список, состоящий из
    словарей
    # заданной структуры; записи должны быть упорядочены по времени
    отправления поезда;
    # вывод на экран информации о поездах, направляющихся в пункт, название
    которого
    # введено с клавиатуры; если таких поездов нет, выдать на дисплей
    соответствующее сообщение.
    # Необходимо оформить команды в виде отдельных функций.

    # Список пунктов (станций).
    stations = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Вызов функции из модуля пакета для ввода информации о станции.
            get_station.get_station(stations)

        elif command.startswith('info '):
            # Разбить команду на части для выделения названия пункта.
            name_station = command.split(' ', maxsplit=1)
            name_station = name_station[1]
            # Вызвать функцию из модуля пакета для вывода информации о
            станции.
```

```

        info.info(stations, name_station)

    elif command == 'help':
        # вызов функции из модуля пакетов для получения информации по
        командам.
        help_command.help_command()

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

```

Рисунок 5 – Код программы индивидуального задания с подключенным пакетом.

```

1 usage new *
def help_command():
    """
    Функция вывода справочной информации о доступных командах, ничего не возвращает.
    """

    print("Список команд:\n")
    print("add - добавить станцию;")
    print("info <станция> - запросить информацию о станции;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

    return None

```

Рисунок 6 – Содержимое модуля «help\_command».

```

1 usage new *
def info(stations, name_station):
    """
    Функция вывода информации о введенной станции, о поездах и их времени (если они есть).
    Функция ничего не возвращает.
    """

    count = 0
    for station in stations:
        if station.get('name') == name_station:
            count += 1
            print("Номер поезда пункта: ", station.get('train'),
                  "Время отправления: ", station.get('time'))

    # Если счетчик равен 0, то станции не найдены.
    if count == 0:
        print("Станции не найдены.")

    return None

```

Рисунок 7 – Содержимое модуля «info».

```

__all__ = ["get_station", "info", "help_command"]

```

Рисунок 8 – Содержимое модуля «\_\_init\_\_».

```

1 usage new*
def get_station(stations):
    """
    Функция запроса данных о станции.
    """
    name = input("Название пункта: ")
    # Создать словарь.
    station = {'name': name}
    print("Добавить поезд? n/y")
    cucle = input()
    if cucle == 'n':
        station['train'] = 'Поездов нет'
        station['time'] = ' '
    else:
        train = input("Номер поезда: ")
        dep_time = input("Время отправления поезда: ")
        # Добавить в словарь поезд и время.
        station['train'] = train
        station['time'] = dep_time

    # Добавить словарь в список.
    stations.append(station)

    # Отсортировать список в случае необходимости по времени поезда.
    if len(stations) > 1:
        stations.sort(key=lambda item: item.get('time', ''))

    return stations

```

Рисунок 9 – Содержимое модуля «get\_station».

```

>>> help
Список команд:

add - добавить станцию;
info <станция> - запросить информацию о станции;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Название пункта: первинск
Добавить поезд? n/y
y
Номер поезда: 001
Время отправления поезда: 10:00
>>> add
Название пункта: второвинск
Добавить поезд? n/y
y
Номер поезда: 002
Время отправления поезда: 12:00
>>> add
Название пункта: третьинск
Добавить поезд? n/y
y
Номер поезда: 003
Время отправления поезда: 9:00
>>> info второвинск
Номер поезда пункта: 002 Время отправления: 12:00
>>> exit

```

Рисунок 10 – Результаты работы программы индивидуального задания.

## Ответы на контрольные вопросы:

1. Модуль – это файл с расширением «.py», предназначенный для хранения часто используемых функций, классов, констант и т.д. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для импортирования их в другие программы. Модули могут быть написаны не только на Python, но и на других языках (например C).

2. Для импортирования модуля можно воспользоваться конструкцией «import имя\_модуля». Если необходимо установить модулю псевдоним, то используется конструкция «import имя\_модуля as новое\_имя». Если нужно импортировать только конкретные объекты, то используется «from имя\_модуля import имя\_объекта», а если нужно импортировать все, то «from имя\_модуля import \*».

3. Пакет – 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). Для импортирования пакетов используется тот же синтаксис, что и для работы с модулями, например – «from имя\_пакета import \*».

4. Файл «\_\_init\_\_.py» может быть пустым или может содержать переменную «\_\_all\_\_», хранящую список модулей, который импортируется при загрузке через конструкцию «from имя\_пакета import \*».

5. Переменная «\_\_all\_\_» хранит список модулей, который импортируется при загрузке через конструкцию «from имя\_пакета import \*».

Вывод: в ходе выполнения лабораторной работы была более подробно изучена работа с модулями и пакетами в языке программирования Python. Также были изучены способы импортирования требуемых модулей, создания псевдонимов для модулей, импортирование отдельных объектов из модуля. Были изучены способы создания модулей и пакетов, понятия модулей и пакетов.