МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №2.19 по дисциплине основы программной инженерии

Выполнил:
Шальнев Владимир Сергеевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

Выполнение:

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


import collections
import pathlib


print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
```

Пример 1

```
Counter({'.py': 5})


Process finished with exit code 0
```

Результат работы примера 1

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

import pathlib


def tree(directory):
    print(f'+ {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = ' ' * depth
        print(f'{spacer}+ {path.name}')


tree(pathlib.Path.cwd())
```

Пример 2

```
+ F:\pythonProject\lab_2_7\examples
 + ex1.py
 + ex2.py
 + ex3.py
 + ex4.py
 + ex5.py
```

Результат работы примера 2

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


from datetime import datetime
import pathlib


time, file_path = max((f.stat().st_mtime, f) for f in
```

```
pathlib.Path.cwd().iterdir())
print(datetime.fromtimestamp(time), file_path)
```

Пример 3

```
2022-04-09 20:28:22.690603 F:\pythonProject\lab_2_7\examples\ex5.py
```

Результат работы примера 3

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

import pathlib


def unique_path(directory, name_pattern):
    counter = 0
    while True:
        counter += 1
        path = directory/name_pattern.format(counter)
        if not path.exists():
            return path


path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
print(path)
```

Пример 4

```
F:\pythonProject\lab_2_7\examples\test001.txt

Process finished with exit code 0
```

Результат работы примера 4

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

import pathlib


path = pathlib.PureWindowsPath(r'C:\Users\gahjelle\realpython\file.txt')
print(path.name)
print(path.parent)
print(path.exists())
```

Пример 5

```
file.txt
C:\Users\gahjelle\realpython
Traceback (most recent call last):
  File "F:\pythonProject\lab_2_7\examples\ex5.py", line 10, in <module>
    print(path.exists())
AttributeError: 'PureWindowsPath' object has no attribute 'exists'

Process finished with exit code 1
```

Результат работы примера 5

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


import sys
import argparse
import os
from datetime import datetime
import json
import jsonschema
from jsonschema import validate
from pathlib import Path


def main_loop():
    trains = []
    while True:
        command = get_command()
        if command == 'exit':
            break
        elif command == 'add':
            trains.append(add_train())
            if len(trains) > 1:
                trains.sort(key=lambda item: item.get('destination', ''))
        elif command == 'list':
            print_list(trains)
        elif command.startswith('select '):
            select_train(command, trains)
        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_trains(file_name, trains)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            trains = load_trains(file_name)
        elif command == 'help':
            print_help()
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


def get_command():
    return input(">>> ").lower()


def add_train(destination=None, number=None, time=None):
```

```
    print(destination)
    print(number)
    print(time)
    if destination is None \
            and number is None \
            and time is None:
        destination = input("Название пункта назначения? ")
        number = input("Номер поезда? ")
        time = input("Время отправления ЧЧ:ММ? ")
    train = {
        'destination': destination,
        'number': int(number),
        'time': datetime.strptime(time, '%H:%M'),
    }
    return train


def print_list(trains):
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 28,
        '-' * 14,
        '-' * 19
    )
    print(line)
    print(
        '| {:^4} | {:^28} | {:^14} | {:^19} |'.format(
            "No",
            "Название пункта назначения",
            "Номер поезда",
            "Время отправления"
        )
    )
    print(line)
    for idx, train in enumerate(trains, 1):
        print(
            '| {:>4} | {:<28} | {:<14} | {:>19} |'.format(
                idx,
                train.get('destination', ''),
                train.get('number', ''),
                train.get('time', 0).strftime("%H:%M")
            )
        )
    print(line)


def print_help():
    print("Список команд:\n")
    print("add - добавить отправление;")
    print("list - вывести список отправлений;")
    print("select <ЧЧ:ММ> - вывод на экран информации о "
          "поездах, отправляющихся после этого времени;")
    print("save <имя файла.json> - сохранить в файл")
    print("load <имя файла.json> - загрузить из файла")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")


def select_train(command, trains):
    count = 0
    parts = command.split(' ', maxsplit=1)
    time = datetime.strptime(parts[1], '%H:%M')
    for train in trains:
        if train.get("time") > time:
```

```
            count += 1
            print(
                '{:>4}: {} {}'.format(
                    count,
                    train.get('destination', ''),
                    train.get("number")
                )
            )
    if count == 0:
        print("Отправлений позже этого времени нет.")


def save_trains(file_name, trains):
    file_name = str(Path.home()) + '\\' + str(file_name)
    if file_name.split('.', maxsplit=1)[-1] != "json":
        print("Несоответствующий формат файла", file=sys.stderr)
        return False

    for i in trains:
        i['time'] = i['time'].time().strftime("%H:%M")
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(trains, fout, ensure_ascii=False, indent=4)
    return True


def validate_json(json_data):
    with open("schema.json", "r", encoding="utf-8") as json_schema:
        schema = json.load(json_schema)
    try:
        validate(instance=json_data, schema=schema)
    except jsonschema.exceptions.ValidationError as err:
        return False
    return True


def load_trains(file_name):
    file_name = str(Path.home()) + '\\' + str(file_name)
    if file_name.split('.', maxsplit=1)[-1] != "json":
        print("Несоответствующий формат файла", file=sys.stderr)
        return []
    print(file_name)
    if not os.path.exists(f"{file_name}"):
        print("Файл не существует", file=sys.stderr)
        return []

    with open(file_name, "r", encoding="utf-8") as fin:
        data = []
        try:
            data = json.load(fin)
            flag = validate_json(data)
        except Exception as e:
            print("Некоректный файл", file=sys.stderr)
            flag = False
        print(flag)
        if flag:
            for i in data:
                i['time'] = datetime.strptime(i['time'], '%H:%M')
            print(data)
            return data
        else:
            return []


def main(command_line=None):
```

```
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="Имя файла для хранения данных"
    )
    parser = argparse.ArgumentParser("trains")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Добавить новое отправление"
    )
    add.add_argument(
        "-trd",
        "--train_dest",
        action="store",
        required=True,
        help="Пункт назначения поезда"
    )
    add.add_argument(
        "-n",
        "--number",
        action="store",
        help="Номер поезда"
    )
    add.add_argument(
        "-t",
        "--time",
        action="store",
        required=True,
        help="Время отправления поезда"
    )
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Вывод информации о всех поездах"
    )
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Выбрать поезда, отправляющиеся "
             "позже указанного времени"
    )
    select.add_argument(
        "-t",
        "--time",
        action="store",
        required=True,
        help="Время отправления"
    )
    args = parser.parse_args(command_line)
    try:
        is_dirty = False
        if os.path.exists(str(Path.home()) + '\\' + str(args.filename)):
            trains = load_trains(args.filename)
        else:
            trains = []
        if args.command == "add":
```

```
            trains.append(add_train(
                args.train_dest,
                args.number,
                args.time
            ))
            is_dirty = True
        elif args.command == "display":
            print_list(trains)
        elif args.command == "select":
            select_train(f"select {args.time}", trains)
        if is_dirty:
            save_trains(args.filename, trains)
    except Exception as e:
        main_loop()


if __name__ == '__main__':
    main()
```

Задание №1

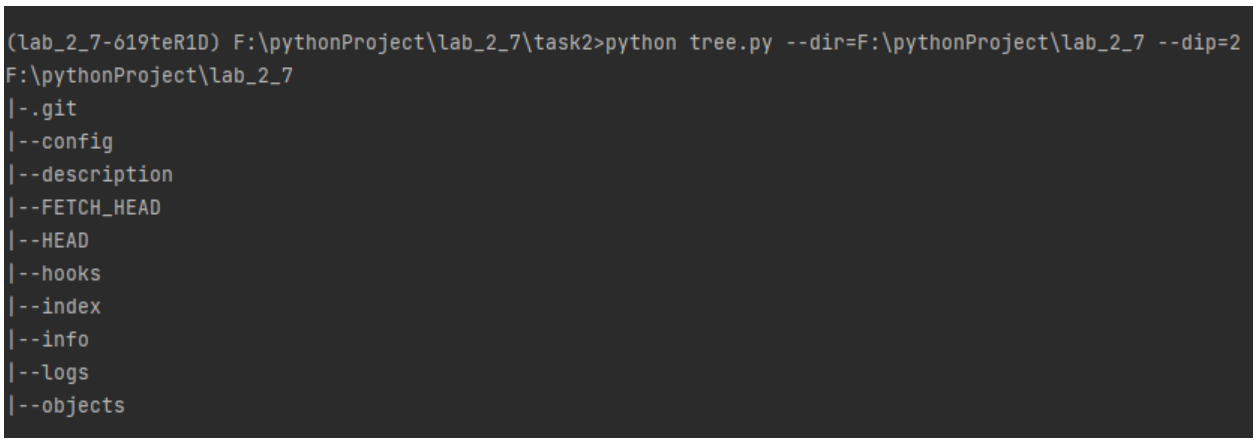
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import click
from pathlib import *


@click.command()
@click.option(
    '--dir',
    default=Path.cwd(),
    help='Ввод директории',
    type=str
)
@click.option(
    '--dip',
    default=-1,
    help='Уровень вложенности для отображения в выводе',
    type=int
)
def display_tree(dir, dip):
    print(f"{dir}")
    for path in sorted(Path(dir).rglob('*')):
        depth = len(path.relative_to(dir).parts)
        if depth <= dip or dip == -1:
            spacer = '-' * depth
            print(f'|{spacer}{path.name}')


if __name__ == '__main__':
    display_tree()
```

Задание №2



Пример работы задания №2

Ответы на вопросы:

1. Какие существовали средства для работы с файловой системой до Python 3.4?

До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк:

>>> path.rsplit('\\', maxsplit=1)[0]

либо с помощью модуля os.path :

>>> os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt'))

2. Что регламентирует PEP 428?

Данный PEP предлагает включить в стандартную библиотеку модуль стороннего разработчика, pathlib [1]. Включение предлагается под предварительной меткой, как описано в PEP 411.

Поэтому изменения API могут быть сделаны либо в рамках процесса PEP, либо после принятия в стандартную библиотеку (и до тех пор, пока предварительная метка не будет снята).

Цель этой библиотеки - предоставить простую иерархию классов для работы с путями файловой системы и обычными операциями, которые пользователи выполняют над ними.

3. Как осуществляется создание путей средствами модуля pathlib?

Есть несколько разных способов создания пути. Прежде всего, существуют classmethods наподобие. cwd() (текущий рабочий каталог) и .home() (домашний каталог вашего пользователя):

>>> import pathlib

>>> pathlib.Path.cwd()

PosixPath('/home/gahjelle/realpython/')

4. Как получить путь дочернего элемента файловой системы с помощью модуля pathlib?

5. Как получить путь к родительским элементам файловой системы с помощью модуля pathlib?

path.parent

6. Как выполняются операции с файлами с помощью модуля pathlib?

Path(dir).rglob('*')

7. Как можно выделить компоненты пути файловой системы с помощью модуля pathlib?

- .name : имя файла без какого-либо каталога
- .parent : каталог, содержащий файл, или родительский каталог, если путь является
- каталогом
- .stem : имя файла без суффикса
- .suffix : расширение файла
- .anchor : часть пути перед каталогами

8. Как выполнить перемещение и удаление файлов с помощью модуля pathlib?

Чтобы переместить файл, используйте .replace() . Обратите внимание, что если место назначения уже существует, .replace() перезапишет его. К сожалению, pathlib явно не поддерживает безопасное перемещение файлов. Чтобы избежать возможной перезаписи пути назначения, проще всего проверить, существует ли место назначения перед заменой:

```
with destination.open(mode='xb') as fid:
    fid.write(source.read_bytes())
```

Каталоги и файлы могут быть удалены с помощью .rmdir() и .unlink() соответственно.

9. Как выполнить подсчет файлов в файловой системе?

Есть несколько разных способов перечислить много файлов. Самым простым является метод .iterdir() , который перебирает все файлы в данном каталоге. В следующем примере комбинируется .iterdir() с классом collection.Counter для подсчета количества файлов каждого типа в текущем каталоге:

```
>>> import collections
>>> collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir())
```

10. Как отобразить дерево каталогов файловой системы?

В следующем примере определяется функция tree() , которая будет печатать визуальное дерево, представляющее иерархию файлов, с корнем в данном каталоге. Здесь мы также хотим перечислить подкаталоги, поэтому мы используем метод .rglob() :

```
def tree(directory):
    print(f'+ {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = ' ' * depth
        print(f'{spacer}+ {path.name}')
```

11. Как создать уникальное имя файла?

Последний пример покажет, как создать уникальное нумерованное имя файла на основе шаблона. Сначала укажите шаблон для имени файла с местом для счетчика. Затем проверьте существование пути к файлу, созданного путем соединения каталога и имени файла (со значением счетчика). Если он уже существует, увеличьте счетчик и попробуйте снова:

```
def unique_path(directory, name_pattern):
    counter = 0
    while True:
        counter += 1
        path = directory/name_pattern.format(counter)
        if not path.exists():
            return path

path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
```

12. Каковы отличия в использовании модуля pathlib для различных операционных систем?

Ранее мы отмечали, что когда мы создавали экземпляр pathlib.Path , возвращался либо объект WindowsPath , либо PosixPath . Тип объекта будет зависеть от операционной системы, которую вы используете. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить WindowsPath или PosixPath , но вы будете ограничивать свой код только этой системой без каких-либо преимуществ. Такой конкретный путь не может быть использован в другой системе:

```
>>> pathlib.WindowsPath('test.md')
```

В некоторых случаях может потребоваться представление пути без доступа к базовой файловой системе (в этом случае также может иметь смысл представлять путь Windows в системе, отличной от Windows, или наоборот). Это можно сделать с помощью объектов PurePath .

```
>>> path =
pathlib.PureWindowsPath(r'C:\Users\gahjelle\realpython\file.txt')
>>> path.name
'file.txt'
>>> path.parent
PureWindowsPath('C:/Users/gahjelle/realpython')
>>> path.exists()
AttributeError: 'PureWindowsPath' object has no attribute 'exists'
```

Вы можете напрямую создать экземпляр PureWindowsPath или PurePosixPath во всех системах. Создание экземпляра PurePath вернет один из этих объектов в зависимости от используемой операционной системы.