

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**«Работа с файловой системе в Python3 с использованием модуля
pathlib»**

**Отчет по лабораторной работе № 2.19
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 2023г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2023

Цель работы: приобретение навыков по работе с файловой системой с помощью библиотеки `pathlib` языка программирования Python версии 3.x.

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл `.gitignore` необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления `git-flow`.
6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы. Создайте для них отдельные модули языка. Зафиксируйте изменения в репозитории.

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

import pathlib
import collections

# 1. Подсчет файлов
if __name__ == "__main__":
    print(collections.Counter(p.suffix for p in
        pathlib.Path.cwd().iterdir()))
```

```
/Users/svetik/Desktop/OPI/OPI_LR_2.19/Py
Counter({'': 5, '.cfg': 1, '.py': 1})

Process finished with exit code 0
```

Рисунок 1 – Результат работы программы

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

```
import pathlib
import collections
```

```
# Подсчет файлов 2
```

```
if __name__ == "__main__":
    print(collections.Counter(p.suffix for p in
pathlib.Path.cwd().glob('*.*')))
```

```
/Users/svetik/Desktop/OPI/OPI_LR_2.19/PyCharm/bi
Counter({'py': 3})
```

```
Process finished with exit code 0
```

Рисунок 2 – Результат работы программы

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

```
import pathlib
```

```
# 2.Показать дерево каталогов
```

```
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}')
```

```
if __name__ == "__main__":
    print(tree(pathlib.Path.cwd()))
```



```
+ wheel-0.38.4.v1
+ pr1.py
+ pr2.py
+ pyvenv.cfg
+ share
    + man
        + man1
            + ttx.1
None
Process finished with exit code 0
```

Рисунок 3 – Результат работы программы

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

import pathlib
from datetime import datetime

# Найти последний измененный файл
if __name__ == "__main__":
    time, file_path = max((f.stat().st_mtime, f)
                          for f in pathlib.Path.cwd().iterdir())
    print(datetime.fromtimestamp(time), file_path)
```

```
pr4 x
/Users/svetik/Desktop/OPI/OPI_LR_2.19/PyCharm/bin/python /Users/svetik/Desktop
2023-05-12 00:06:04.351898 /Users/svetik/Desktop/OPI/OPI_LR_2.19/PyCharm/pr4.p
Process finished with exit code 0
```

Рисунок 4 – Результат работы программы

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

import pathlib

# Создать уникальное имя файла
def unique_path(directory, name_pattern):
    counter = 0
    while True:
```

```

        counter += 1
        path = directory/name_pattern.format(counter)
        if not path.exists():
            return path

if __name__ == "__main__":
    path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
    print(path)

```

```

/Users/svetik/Desktop/OPI/OPI_LR_2.19/PyCharm/bin/python /Users/
/Users/svetik/Desktop/OPI/OPI_LR_2.19/PyCharm/test001.txt

Process finished with exit code 0

```

Рисунок 5 – Результат работы программы

8. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

Задание 1:

Для своего варианта лабораторной работы 2.17 добавьте возможность хранения файла данных в домашнем каталоге пользователя. Для выполнения операций с файлами необходимо использовать модуль `pathlib`.

```

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

import argparse
import json
import os.path
import sys
import pathlib

def get_way(ways, start, finish, num):
    """
    Запросить данные о маршруте.
    """
    # Создать словарь.
    ways.append(
        {
            'start': start,
            'finish': finish,

```

```

        'num': num,
    }
)
return ways

def display_way(numbers):
    """
    Отобразить список маршрутов.
    """
    if numbers:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^30} | {:^15} |'.format(
                "No",
                "Название начального маршрута",
                "Название конечного маршрута",
                "Номер маршрута"
            )
        )
        print(line)

        # Вывести данные о всех маршрутах.
        for idx, way in enumerate(numbers, 1):
            print(
                '| {:>4} | {:<30} | {:<30} | {:>15} |'.format(
                    idx,
                    way.get('start', ''),
                    way.get('finish', ''),
                    way.get('num', 0)
                )
            )
            print(line)
    else:
        print("Список пуст.")

def find_way(numbers, nw):
    """
    Выбрать маршрут с данным номером.
    """
    # Список маршрутов
    result = []
    for h in numbers:
        if nw in str(h.values()):
            result.append(h)

    # Возвратить список выбранных маршрутов.

```

```
    return result

def save_ways(file_name, ways):
    """
    Сохранить номера всех маршрутов в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(ways, fout, ensure_ascii=False, indent=4)

def load_ways(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("ways")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления маршрута.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new student"
    )

    add.add_argument(
        "-s",
        "--start",
        action="store",
        required=True,
        help="Start Route"
    )
)
```

```
add.add_argument(
    "-f",
    "--finish",
    action="store",
    help="Final Route"
)

add.add_argument(
    "-n",
    "--num",
    action="store",
    required=True,
    help="Route number"
)

# Создать субпарсер для отображения всех маршрутов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all ways"
)

# Создать субпарсер для поиска маршрутов.
find = subparsers.add_parser(
    "find",
    parents=[file_parser],
    help="find the ways"
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)
path = pathlib.Path.home() / args.filename

# Загрузить все маршруты из файла, если файл существует.
is_dirty = False
if path.exists():
    ways = load_ways(path)
else:
    ways = []

# Добавить маршрут.
if args.command == "add":
    ways = get_way(
        ways,
        args.start,
        args.finish,
        args.num
    )
    is_dirty = True

# Отобразить всех студентов.
elif args.command == "display":
    display_way(ways)

# Выбрать требуемых студентов.
```




```
elif args.command == "find":
    found = find_way(ways)
    display_way(found)

# Сохранить данные в файл, если список студентов был изменен.
if is_dirty:
    save_ways(path, ways)

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

```
/Library/Frameworks/Python.framework/Versions/3.11/Resources/Python.app/Contents/MacOS/Python: can't open file '/Users/svetik/Des
Errno 2] No such file or directory
(PyCharm) (base) svetik@MacBook-Air-Svetik PyCharm % python ind_1.py add ind_1.json --start="hhshs" --finish="fjjfjf" --num=44

(PyCharm) (base) svetik@MacBook-Air-Svetik PyCharm % python ind_1.py display ind_1.json

+-----+-----+-----+-----+
| No | Название начального маршрута | Название конечного маршрута | Номер маршрута |
+-----+-----+-----+-----+
| 1 | hhshs | fjjfjf | 44 |
+-----+-----+-----+-----+
(PyCharm) (base) svetik@MacBook-Air-Svetik PyCharm %
```

Рисунок 6 – Результат работы программы

Задание 2: Разработайте аналог утилиты tree в Linux. Используйте возможности модуля argparse для управления отображением дерева каталогов файловой системы. Добавьте дополнительные уникальные возможности в данный программный продукт.

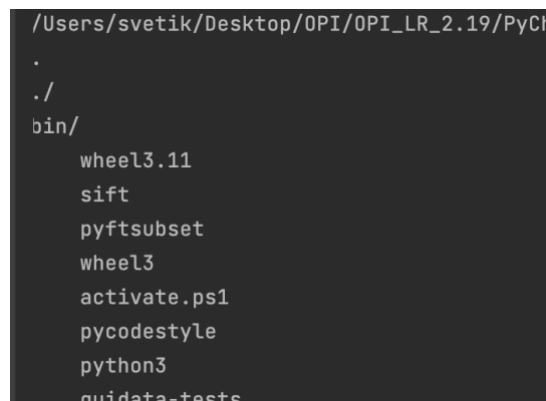
```
import os
import argparse

def print_tree(dir_path, level=0):
    """
    Выводит дерево каталогов и файлов начиная с указанной директории
    """
    print(' ' * level * 4 + os.path.basename(dir_path) + '/')
    level += 1

    for item in os.listdir(dir_path):
        item_path = os.path.join(dir_path, item)
        # Если это папка, вызываем функцию рекурсивно для неё, а если файл,
        # выводим
        # его имя с отступом в зависимости от уровня вложенности
        if os.path.isdir(item_path):
            print_tree(item_path, level)
        else:
            print(' ' * level * 4 + item)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Утилита вывода дерева каталогов файловой системы')
    parser.add_argument('dir', metavar='DIR', type=str, nargs='?',
                        default='.', help='Директория, для которой нужно вывести дерево')
    parser.add_argument('-d', '--depth', type=int, default=-1, help='Глубина рекурсии')
    args = parser.parse_args()

    print(args.dir)
    print_tree(args.dir, args.depth)
```



```
/Users/svetik/Desktop/OPI/OPI_LR_2.19/PyC
.
./
bin/
  wheel3.11
  sift
  pyftsubset
  wheel3
  activate.ps1
  pycodestyle
  python3
  quidata-tests
```

Рисунок 9 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.

Вопросы для защиты работы:

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.split('\\', 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?

PEP 428 - "The pathlib module - representing file system paths as objects" регламентирует использование модуля pathlib в Python.

3. Как осуществляется создание путей средствами модуля pathlib?

Создание путей средствами модуля pathlib осуществляется с помощью класса `pathlib.Path`. Прежде всего, существуют classmethods наподобие `cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог вашего пользователя). Путь также может быть явно создан из его строкового представления.

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

Чтобы получить путь дочернего элемента файловой системы с помощью модуля `pathlib` в Python, можно использовать метод `joinpath()`.

Допустим, у вас есть объект `Path`, представляющий путь к родительскому каталогу, и вы хотите получить путь к дочернему элементу `child_dir` в этом каталоге. Для этого можно вызвать метод `joinpath()` на объекте

Path, передав в него имя дочернего элемента в качестве аргумента.

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

Для получения пути к родительским элементам файловой системы с помощью модуля pathlib в Python, можно использовать атрибут `parent` объекта `Path`. Этот атрибут возвращает объект `Path`, представляющий родительский каталог текущего элемента.

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

Он позволяет выполнять операции с файлами с помощью объектов `Path`, которые представляют пути файловой системы.

```
from pathlib import Path
```

```
# создание файла file_path =  
Path('/path/to/myfile.txt').touch()
```

```
# чтение содержимого файла  
file_path =  
Path('/path/to/myfile.txt') with  
file_path.open() as f:    contents = f.read()
```

Запись в файл:

```
# запись в файл file_path =  
Path('/path/to/myfile.txt') with  
file_path.open(mode='w') as f:  
f.write('Hello, world!')
```

Переименование файла:

```
# переименование файла
file_path = Path('/path/to/myfile.txt')
new_file_path = Path('/path/to/newfile.txt')
file_path.rename(new_file_path)
```

```
# удаление файла
file_path = Path('/path/to/myfile.txt') file_path.unlink()
```

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

Модуль pathlib в Python позволяет выделить различные компоненты пути файловой системы, такие как имя файла, расширение файла, родительский каталог и т.д.

```
from pathlib import Path
```

```
# получение имени файла
file_path = Path('/path/to/myfile.txt')
file_name = file_path.name print(file_name) #
'myfile.txt'
```

```
# получение расширения файла
file_path = Path('/path/to/myfile.txt')
file_ext = file_path.suffix print(file_ext) #
'.txt'
```

```
# получение родительского каталога
file_path = Path('/path/to/myfile.txt') parent_dir =
file_path.parent print(parent_dir) # '/path/to'
```

```
# получение всех компонентов пути file_path
= Path('/path/to/myfile.txt') components =
file_path.parts print(components) # ('/', 'path', 'to',
'myfile.txt')
```

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

Модуль `pathlib` также предоставляет методы для перемещения и удаления каталогов, такие как `Path.rename()` и `Path.rmdir()`. 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.

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. Более гибкие списки файлов могут быть созданы с помощью методов `.glob()` и `.rglob()` (рекурсивный глоб).

10. Как отобразить дерево каталогов файловой системы?

Для отображения дерева каталогов файловой системы в Python можно использовать модуль `pathlib` и рекурсивную функцию `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?

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.

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`.

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.