Институт цифрового развития
Кафедра инфокоммуникаций

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

**ОТЧЕТ**
**по лабораторной работе №19**
**дисциплины**
**«Основы программной инженерии»**

Выполнил:
Мизин Глеб Егорович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Примеры

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

import pathlib
import collections

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

Рисунок 1 – Пример 1

```python
#!/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}")


if __name__ == "__main__":
    # Показать дерево каталогов
    print(tree(pathlib.Path.cwd()))
```

Рисунок 2 – Пример 2

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

import pathlib
from datetime import datetime

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

Рисунок 3 – Пример 3

```python
#!/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


if __name__ == "__main__":
    # Создать уникальное имя файла
    # {:03d} - принимает целое число в качестве аргумента,
    # задает минимальную ширину 3
    # и заполняет оставшиеся места нулями
    path = unique_path(pathlib.Path.cwd(), "test{:03d}.txt")
    print(path)
```

Рисунок 4 – Пример 4

Задание №1:

```
158         path = pathlib.Path.home() / args.filename
159
160         is_dirty = False
161         if path.exists():
162             requisites = load_workers(path)
163         else:
164             requisites = []
165
```

Рисунок 5 – изменённый код для задания №1

Задание №2

```
9   def tree_size(path, seen, head="", tail=""):
10      if path.resolve() not in seen:
11          seen.add(path.resolve())
12          size = float('{:.3f}'.format(os.path.getsize(path)/1024))
13          print(head + path.name + " " + str(size) + "KB")
14          dirs = []
15          files = []
16          if path.is_dir():
17              dirs = sorted(filter(Path.is_dir, path.iterdir()))
18              files = sorted(filter(Path.is_file, path.iterdir()))
19          entries = dirs + files
20          for i, entry in enumerate(entries):
21              if i < len(entries) - 1:
22                  tree_size(entry, seen, tail + "├──", tail + "│   ")
23              else:
24                  tree_size(entry, seen, tail + "└──", tail + "    ")
25
26
27  def tree_a(path, seen, head="", tail=""):
28      if path.resolve() not in seen:
29          seen.add(path.resolve())
30          print(head + path.name)
31          dirs = []
32          files = []
33          if path.is_dir():
34              dirs = sorted(filter(Path.is_dir, path.iterdir()))
35              files = sorted(filter(Path.is_file, path.iterdir()))
```

Рисунок 6 – Код задания №2

Контрольные вопросы

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.

3. Как осуществляется создание путей средствами модуля pathlib ?

Все, что вам действительно нужно знать, это класс `pathlib.Path`. Есть несколько разных способов создания пути. Прежде всего, существуют classmethods наподобие `.cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог вашего пользователя):

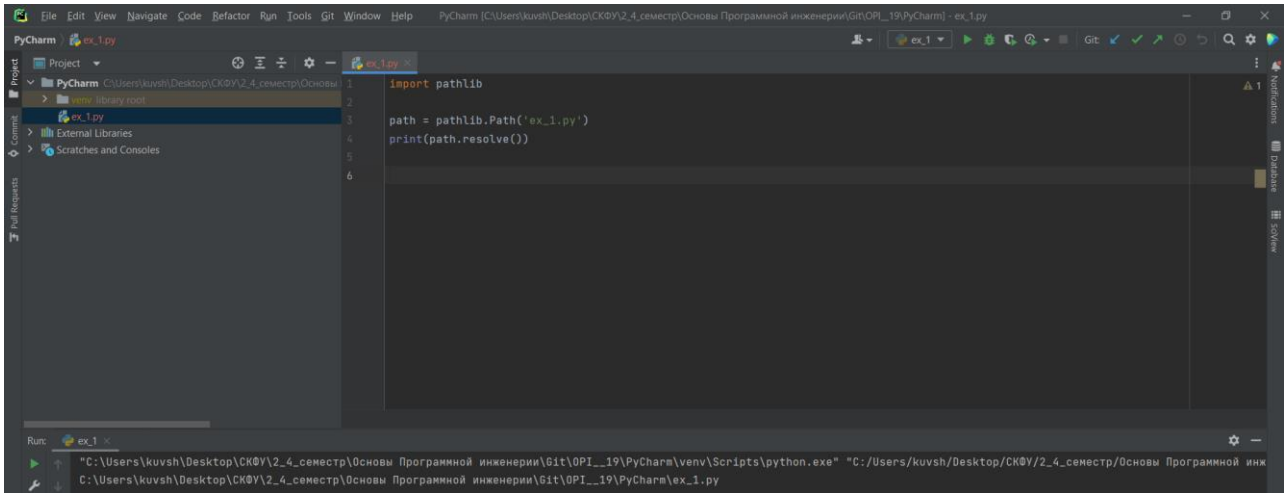
```
>>> import pathlib
>>> pathlib.Path.cwd()
PosixPath('/home/gahjelle/realpython/')
```

Третий способ построения пути - это соединение частей пути с помощью специального оператора `/`. Оператор прямой косой черты используется независимо от фактического разделителя пути на платформе:

```
>>> pathlib.Path.home()/'python'/'scripts'/'test.py'
PosixPath('/home/gahjelle/python/scripts/test.py')
```

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

```
>>> path = pathlib.Path('test.md')
>>> path.resolve()
```




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

Различные части пути удобно доступны как свойства. Основные примеры включают в себя:

- `.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

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

## Чтение и запись файлов

Традиционно для чтения или записи файла в Python использовалась встроенная функция `open()`. Это все еще верно, поскольку функция `open()` может напрямую использовать объекты `Path`. Следующий пример находит все заголовки в файле Markdown и печатает их:

```
path = pathlib.Path.cwd() / 'test.md'
with open(path, mode='r') as fid:
    headers = [line.strip() for line in fid if line.startswith('#')]
print('\n'.join(headers))
```

Эквивалентной альтернативой является вызов `.open()` для объекта `Path`:

```
with path.open(mode='r') as fid:
    ...
```

Фактически, `Path.open()` вызывает встроенную функцию `open()` за кулисами. Какой вариант вы используете, это в основном дело вкуса.

Для простого чтения и записи файлов в библиотеке `pathlib` есть несколько удобных методов:

- `.read_text()`: открыть путь в текстовом режиме и вернуть содержимое в виде строки.
- `.read_bytes()`: открыть путь в двоичном/байтовом режиме и вернуть содержимое в виде строки байтов.
- `.write_text()`: открыть путь и записать в него строковые данные.
- `.write_bytes()`: открыть путь в двоичном/байтовом режиме и записать в него данные.

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

Различные части пути удобно доступны как свойства. Основные примеры включают в себя:

- `.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

Вот эти свойства в действии:

```
>>> path
PosixPath('/home/gahjelle/realpython/test.md')
>>> path.name
'test.md'
```

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

Через `pathlib` вы также получаете доступ к базовым операциям на уровне файловой системы, таким как перемещение, обновление и даже удаление файлов. По большей части эти методы не выдают предупреждение и не ждут подтверждения, прежде чем информация или файлы будут потеряны. Будьте осторожны при использовании этих методов.

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:

```
if not destination.exists():
    source.replace(destination)
```

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())
Counter({'.md': 2, '.txt': 4, '.pdf': 2, '.py': 1})
```

Более гибкие списки файлов могут быть созданы с помощью методов `.glob()` и `.rglob()` (рекурсивный глоб). Например, `pathlib.Path.cwd().glob('*.txt')` возвращает все файлы с суффиксом `.txt` в текущем каталоге. Следующее только подсчитывает типы файлов, начинающиеся с `p`:

```
>>> import collections
>>> collections.Counter(p.suffix for p in pathlib.Path.cwd().glob('*.p*'))
Counter({'.pdf': 2, '.py': 1})
```

## 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')
```

Если каталог уже содержит файлы `test001.txt` и `test002.txt`, приведенный выше код установит для `path` значение `test003.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')
NotImplementedError: cannot instantiate 'WindowsPath' on your system
```