

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

**Отчет по лабораторной работе №2.19 по дисциплине:  
Основы программной инженерии**

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2022 г.

## ВЫПОЛНЕНИЕ:

### 1. Проблема с обработкой пути к файлу Python

```
import glob
import os
import shutil

for file_name in glob.glob('*.txt'):
    new_path = os.path.join('archive', file_name)
    shutil.move(file_name, new_path)
```

### 2. Подсчет файлов.

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

import pathlib
import collections

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

if \_\_name\_\_ == "\_\_main\_\_"

ex1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.19/ex1.py  
Counter({' .py': 3, '': 1})

### 3. Показ дерева 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__":
    tree(pathlib.Path.cwd())

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

ex2 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.19/ex2.py  
+ D:\УЧЕБА\ОПИ\2.19  
+ .idea  
+ .gitignore  
+ 2.19.iml  
+ inspectionProfiles  
+ profiles\_settings.xml  
+ Project\_Default.xml  
+ misc.xml  
+ modules.xml  
+ workspace.xml  
+ ex1.py  
+ ex2.py  
+ ex3.py

### 4. Нахождение последнего измененного файла.

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

from datetime import datetime
import pathlib

def last_changed(directory):
    time, file_path = max(
        (f.stat().st_mtime, f) for f in directory.iterdir()
    )
    print(datetime.fromtimestamp(time), file_path)

if __name__ == "__main__":
    last_changed(pathlib.Path.cwd())

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

ex3 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.19/ex3.py  
2022-04-10 20:42:34.585019 D:\УЧЕБА\ОПИ\2.19\ex3.py

## 5. Создание уникального имени файла.

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

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

ex4 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.19/ex4.py  
D:\УЧЕБА\ОПИ\2.19\test001.txt

### Выполнение 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`.

```
PS D:\УЧЕБА\ОПИ\2.19> py ind1.py add ind1.json --surname="Petrov" --name="Ilya" --number="6785" --date_obj="07.03.2004"
```

ind1.json – Блокнот

Файл Правка Формат Вид Справка

```
[
  {
    "surname": "Petrov",
    "name": "Ilya",
    "number": "6785",
    "date_obj": "07.03.2004"
  }
]
```

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.

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.19/ind2.py
This program is showing files and directories on C: drive or CWD
Where do you want to work? h - home, c - CWD: h
Type the directory name: .ipython
Enter the level of search if needed (-1) if don't: 2
Do you want to print only the directories? d - yes: y
.ipython
├── extensions
├── nbextensions
└── profile_default
    ├── db
    ├── history.sqlite
    ├── log
    ├── pid
    ├── security
    └── startup
```

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?

Данный PEP предлагает включить в стандартную библиотеку модуль стороннего разработчика – pathlib. Включение предлагается под предварительной меткой, как описано в PEP 411. Поэтому изменения в 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?

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()
```

Вывод: `PosixPath('/home/gahjelle/realpython/')`

Путь также может быть явно создан из его строкового представления:

```
pathlib.Path(r'C:\Users\gahjelle\realpython\file.txt')
```

Вывод: `WindowsPath('C:/Users/gahjelle/realpython/file.txt')` Объединение путей: с помощью «`\`» или `.joinpath()` `pathlib.Path.home().joinpath('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').resolve()
```

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

### 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? 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))
```

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

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:

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

Тем не менее, это оставляет дверь открытой для возможного состояния гонки. Другой процесс может добавить файл по пути `destination` между выполнением оператора `if` и метода `.replace()` . Если это вызывает озабоченность, более безопасный способ - открыть путь назначения для

создания exclusive и явно скопировать исходные данные:

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

```
fid.write(source.read_bytes())
```

Приведенный выше код вызовет `FileExistsError`, если `destination` уже существует. Технически это копирует файл. Чтобы выполнить перемещение, просто удалите `source` после завершения копирования.

Когда вы переименовываете файлы, полезными методами могут быть `.with_name()` и `.with_suffix()`. Они оба возвращают исходный путь, но с замененным именем или суффиксом соответственно.

```
path PosixPath('/home/gahjelle/realpython/test001.txt').path.with_suffix('.py')
```

```
PosixPath('/home/gahjelle/realpython/test001.py')
```

```
path.replace(path.with_suffix('.py'))
```

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 `collections.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?

```
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:

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

`NotImplementedError: cannot instantiate 'WindowsPath' on your system`

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'
```

Windows использует «\» , а Mac и Linux используют «/» в качестве разделителя. Это различие может привести к трудно обнаруживаемым ошибкам.