Dealing with file systems

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Computer User

- log files
- build artifacts
- directory trees
- structured data
- unstructured data
- ML models

Filesystem

- File system is a hierarchy
- The Unix `tree` command

```
??? Makefile
??? README.md
??? demos
?   ??? flask-sklearn
?   ?   ??? Dockerfile
?   ?   ??? Makefile
?   ?   ??? README.md
?   ?   ??? app.py
?   ?   ??? ml_prediction.joblib
```

Human User

- config files
- user profile data
- business documents
- code
- data science projects
- ML models

Leaning into os.walk

- `os.walk` returns:
  - `root`
  - `dirs`
  - `files`
- Returns a generator

```
# generator only returns a result at a time
foo = os.walk("/tmp")
type(foo)
```

```
generator
```

Finding file extensions

- splitting off a file extension

```
fullpath = "/tmp/somestuff/data.csv"
_, ext = os.path.splitext(fullpath)
```

```
'.csv'
```

Let's practice.

COMMAND LINE AUTOMATION IN PYTHON

Find files matching a pattern

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Using Path.glob()

- `Path.glob()`
  - finds patterns in directories
  - yields matches
  - can recursively search

Simple glob patterns

```
from pathlib import Path
```

```
path = Path("data")
list(path.glob("*.csv"))
```

```
[PosixPath('mydata.csv'), PosixPath('yourdata.csv')]
```

Recursive glob patterns

```
from pathlib import Path
```

```
path = Path("data")
list(path.glob("**/*.csv"))
```

```
[PosixPath('data/one.csv'), PosixPath('data/moredata/two.csv')]
```

Using os.walk to find patterns

- `os.walk` pattern matching
  - more explicit
  - can explicitly look at directories or files
  - doesn't return `Path` object

```
import os
result = os.walk("/tmp")
# consume the generator
next(result)
# Find your pattern here....
```

Using fnmatch

- Supports Unix shell wildcard matches
- Can be converted to regular expression

```
if fnmatch.fnmatch(file, "*.csv"):
log.info(f"Found match {file}")
```

Converting fnmatch to regular expression

- fnmatch.translate converts pattern to regex

```
import fnmatch, re
regex = fnmatch.translate('*.csv')
pattern = re.compile(regex)
print(pattern)
```

```
re.compile(r'(?s:.*\.csv)\Z', re.UNICODE)
```

```
pattern.match("titanic.csv")
```

```
<re.Match object; span=(0, 11), match='titanic.csv'>
```

Let's practice!

COMMAND LINE AUTOMATION IN PYTHON

High-level file and directory operations

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Two powerful modules

- `shutil` : high-level file operations
  - copy tree
  - delete tree
  - archive tree
- `tempfile` : generates temporary files and directories

Using shutil.copytree

- Can recursively copy a tree of files and folders

```
from shutil import copytree, ignore_patterns
```

- Can ignore patterns

```
copytree(source, destination, ignore=ignore_patterns('*.txt',
  '*.excel'))
```

copytree in action

```
In [1]: pwd
Out[1]: '/private/tmp'
In [2]: !mkdir sometree && touch sometree/somefile.txt
In [3]: from shutil import copytree
In [5]: copytree("sometree", "newtree")
Out[5]: 'newtree'
In [6]: !ls -l newtree/
total 0
-rw-r--r--  1 noahgift  wheel  0 May 19 20:08 somefile.txt
```

Using shutil.rmtree

- Can recursively delete tree of files and folders

```
from shutil import rmtree
```

```
rmtree(source, destination)
```

Using shutil.make_archive

- Archiving a tree with `make_archive`

```
from shutil import make_archive
```

```
make_archive("somearchive", "gztar", "inside_tmp_dir")
```

```
'/tmp/somearchive.tar.gz'
```

Automation Takeaways

- Use the Python standard library
- If an automation tasks requires a lot of code
  - The approach may be incorrect
  - Consult the Python standard library
  - Look at 3rd party Python libraries
- The less code you write, the less bugs you have

Practicing high-level automation

COMMAND LINE AUTOMATION IN PYTHON

Using pathlib

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Using pathlib.Path

```
from pathlib import Path
```

- Make a path object

```
path = Path("/usr/bin")
```

- List items in directory as object

```
list(path.glob("*"))[0:4]
```

```
[PosixPath('/usr/bin/link'),
 PosixPath('/usr/bin/tput'),
```

Working with PosixPath objects

```
mypath.cwd()
```

```
PosixPath('/app')
```

```
mypath.exists()
```

```
True
```

More PosixPath

```
mypath.as_posix()
```

```
'/usr/bin/link'
```

Open a file with pathlib

- Open a `Makefile` from a path object

```
from pathlib import Path
some_file = Path("Makefile")
```

- Print the last line of the `Makefile`

```
with some_file.open() as file_to_read:
    print(file_to_read.readlines()[-1:])
```

```
['all: install lint test\n']
```

Create a directory with pathlib

- Path objects can create directories

```
from pathlib import Path
tmp = Path("/tmp/inside_tmp_dir")
tmp.mkdir()
```

- Contents of the directory

```
ls -l /tmp/
```

```
inside_tmp_dir/
```

Write text with pathlib

- `write_text()` is a serious shortcut

```
write_path = Path("/tmp/some_random_file.txt")
write_path.write_text("Wow")
```

```
3
```

```
print(write_path.read_text())
```

```
'Wow'
```

Rename a file with pathlib

- renaming a file with `pathlib`

```
from pathlib import Path
# Create a Path object
modify_file = Path("/tmp/some_random_file.txt")
#rename file
modify_file.rename("/tmp/some_random_file_renamed.txt")
```

```
ls /tmp
```

```
some_random_file_renamed.txt
```

Practicing with pathlib

COMMAND LINE AUTOMATION IN PYTHON