Automation Cheat Sheet

Websites | WhatsApp | Emails
Excel | PDF | Folder Operation

udemy

Frank Andrade

File & Folder Operation

We can create folders and manipulate files in Python using Path.

Path

Import Path:

```
from pathlib import Path
```

Get current working directory:

```
>>> Path.cwd()
'/Users/frank/Projects/DataScience'
```

List directory content:

```
>>> list(Path().iterdir())
[PosixPath('script1.py'), PosixPath('script2.py')]
```

List directory content within a folder:

```
>>> list(Path('Dataset').iterdir())
```

Joining paths:

```
>>> from pathlib import Path, PurePath
>>> PurePath.joinpath(Path.cwd(), 'Dataset')
'/Users/frank/Projects/DataScience/Dataset'
```

Create a directory:

```
>>> Path('Dataset2').mkdir()
>>> Path('Dataset2').mkdir(exist_ok=True)
```

Rename a file:

```
>>> current_path = Path('Data')
>>> target_path = Path('Dataset')
>>> Path.rename(current_path, target_path)
```

Check existing file:

```
>>> check_path = Path('Dataset')
>>> check_path.exists() # True/False
```

Metadata:

```
>>> path = Path('test/expenses.csv')
>>> path.parts
('test', 'expenses.csv')
>>> path.name
expenses.csv
>>> path.stem
expenses
>>> path.suffix
.csv
```

Regex

We use regex to create patterns that help match text.

Metacharacters

\d	Digit (0-9)
\D	No digits (0-9)
\w	Word Character (a-z, A-Z, 0-9, _)
\W	Not a Word Character
\s	Whitespace (space, tab, new line)
\S	No Whitespace (space, tab, new line)
.	Any character except new line
\	Ignores any special character
^	Beginning of a string
$	End of a string

Quantifiers & Groups

*	0 or more (greedy)
+	1 or more (greedy)
?	0 or 1
{3}	Exact number
{n,}	More than n characters
{3,4}	Range of numbers (Min, Max)
( )	Group
[ ]	Matches characters in brackets
[^]	Matches characters not in brackets
|	Or

Other Metacharacters

\b	Word boundary
\B	No word boundary
\1	Reference

Table Extraction

We can use camelot to extract tables from PDFs and pandas to extract tables from some websites.

PDF

Import library:

```
import camelot
```

Read PDF:

```
tables=camelot.read_pdf('foo.pdf',
                        pages='1',
                   flavor='lattice')
```

Export tables:

```
tables.export('foo.csv',
           f='csv',
           compress=True)
```

Export first table to a CSV file:

```
tables[0].to_csv('foo.csv')
```

Print as a dataframe:

```
print(tables[0].df)
```

Websites

Import library:

```
import pandas as pd
```

Read table:

```
tables=pd.read_html('https://xyz.com')
```

Printing table:

```
print(tables[0])
```

Send Email & Message

With Python we can send emails and WhatsApp messages.

Email

Import libraries:

```
import smtplib
import ssl
from email.message import EmailMessage
```

Set variables:

```
email_sender = 'Write-sender-here'
email_password = 'Write-passwords-here'
email_receiver = 'Write-receiver-here'

subject = 'Check this out!'
body = """
I've just published a new video on YouTube
"""
```

Send email:

```
em = EmailMessage()
em['From'] = email_sender
em['To'] = email_receiver
em['Subject'] = subject
em.set_content(body)
context = ssl.create_default_context()

with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
 smtp.login(email_sender, email_password)
 smtp.sendmail(email_sender, email_receiver, em.as_string())
```

WhatsApp

Import libraries:

```
import pywhatkit
```

Send message to a contact:

```
# syntax: phone number with country code, message, hour and minutes
pywhatkit.sendwhatmsg('+1xxxxxxxx', 'Message 1', 18, 52)
```

Send message to a contact and close tab after 2 seconds:

```
# syntax: same as above plus wait_time, tab_close and close_time
pywhatkit.sendwhatmsg("+1xxxxxxx", "Message 2", 18, 55, 15, True, 2)
```

Send message to a group:

```
# syntax: group id, message, hour and minutes
pywhatkit.sendwhatmsg_to_group("write-id-here", "Message 3", 19, 2)
```

Create Reports

We can create an Excel report in Python using openpyxl.

Excel

Create workbook:

```
from openpyxl import Workbook

wb = Workbook()  # create workbook
ws = wb.active  # grab active worksheet
ws['C1'] = 10  # assign data to a cell
wb.save("report.xlsx")  # save workbook
```

Working with existing workbook:

```
from openpyxl import load_workbook

wb = load_workbook('report_2021.xlsx')
sheet = wb['Report'] # grab worksheet "Report"
```

Cell references:

```
min_column = wb.active.min_column
max_column = wb.active.max_column
min_row = wb.active.min_row
max_row = wb.active.max_row
```

Create Barchart:

```
from openpyxl.chart import BarChart, Reference
barchart = BarChart()
```

Locate data:

```
data = Reference(sheet,
                 min_col=min_column+1,
                 max_col=max_column,
                 min_row=min_row,
                 max_row=max_row)
```

Locate categories:

```
categories = Reference(sheet,
                       min_col=min_column,
                       max_col=min_column,
                       min_row=min_row+1,
                       max_row=max_row)
```

Add data and categories:

```
barchart.add_data(data, titles_from_data=True)
barchart.set_categories(categories)
```

Add chart:

```
sheet.add_chart(barchart, "B12")
```

Save existent workbook:

```
wb.save('report_2021.xlsx')
```

Web Automation

Web automation is the process of automating web actions like clicking on buttons, selecting elements within dropdowns, etc. The most popular tool to do this in Python is Selenium.

Selenium 4

Note that there are a few changes between Selenium 3.x versions and Selenium 4.

Import libraries:

```
 from selenium import webdriver
 from selenium.webdriver.chrome.service import Service

web="www.google.com"
path='introduce chromedriver path'
service = Service(executable_path=path)
driver = webdriver.Chrome(service=service)
driver.get(web)
```

Find an element

```
 driver.find_element(by="id", value="...")
```

Find elements

```
 driver.find_elements(by="xpath", value="...")
```

Quit driver

```
 driver.quit()
```

Getting the text

```
 data = element.text
```

Implicit Waits

```
 import time
 time.sleep(2)
```

Explicit Waits

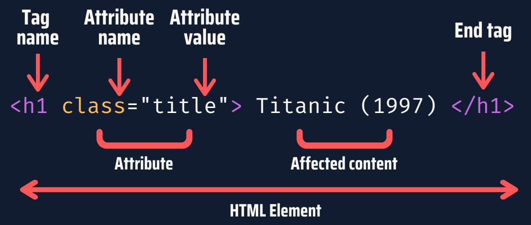
```
 from selenium.webdriver.common.by import By
 from selenium.webdriver.support.ui import WebDriverWait
 from selenium.webdriver.support import expected_conditions as EC

 WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.ID, 'id_name')))
 #Wait 5 seconds until an element is clickable
```

Options: Headless mode, change window size

```
 from selenium.webdriver.chrome.options import Options
 options = Options()
 options.headless = True
 options.add_argument('window-size=1920x1080')
 driver=webdriver.Chrome(service=service,options=options)
```

HTML for Web Automation

Let's take a look at the HTML element syntax.

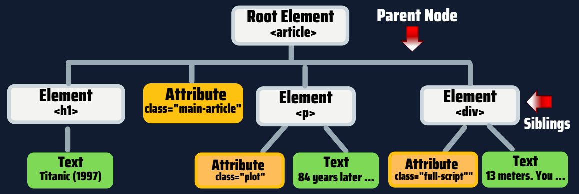


This is a single HTML element, but the HTML code behind a website has hundreds of them.

HTML code example

```
<article class="main-article">
 <h1> Titanic (1997) </h1>
 <p class="plot"> 84 years later ... </p>
 <div class="full-script"> 13 meters. You ... </div>
</article>
```

The HTML code is structured with "nodes". Each rectangle below represents a node (element, attribute and text nodes)



- The "root node" is the top node. In this example, <article> is the root.
- Every node has exactly one "parent", except the root. The <h1> node's parent is the <article> node.
- "Siblings" are nodes with the same parent.
- One of the best ways to find an element is building its XPath

XPath

We need to learn how to build an XPath to properly work with Selenium.

XPath Syntax

An XPath usually contains a tag name, attribute name, and attribute value.

```
//tagName[@AttributeName="Value"]
```

Let's check some examples to locate the article, title, and transcript elements of the HTML code we used before.

```
//article[@class="main-article"]
//h1
//div[@class="full-script"]
```

XPath Functions and Operators

XPath functions

```
//tag[contains(@AttributeName, "Value")]
```

XPath Operators: and, or

```
//tag[(expression 1) and (expression 2)]
```

XPath Special Characters

/	Selects the children from the node set on the left side of this character
//	Specifies that the matching node set should be located at any level within the document
.	Specifies the current context should be used (refers to present node)
..	Refers to a parent node
*	A wildcard character that selects all elements or attributes regardless of names
@	Select an attribute
()	Grouping an XPath expression
[n]	Indicates that a node with index "n" should be selected

More Coming Soon ...
(stay tuned!)

Below there are my guides, tutorials and all my Python and data science courses:
- Medium Guides
- YouTube Tutorials
- Data Science Course
- Web Scraping Course (I teach Selenium here)
- Make Money Using Your Programming & Data Science Skills

Made by Frank Andrade frank-andrade.medium.com