

ODSC EUROPE 2021

BASIC PYTHON FOR DATA PROCESSING

JAIIME BUELTA

ABOUT ME

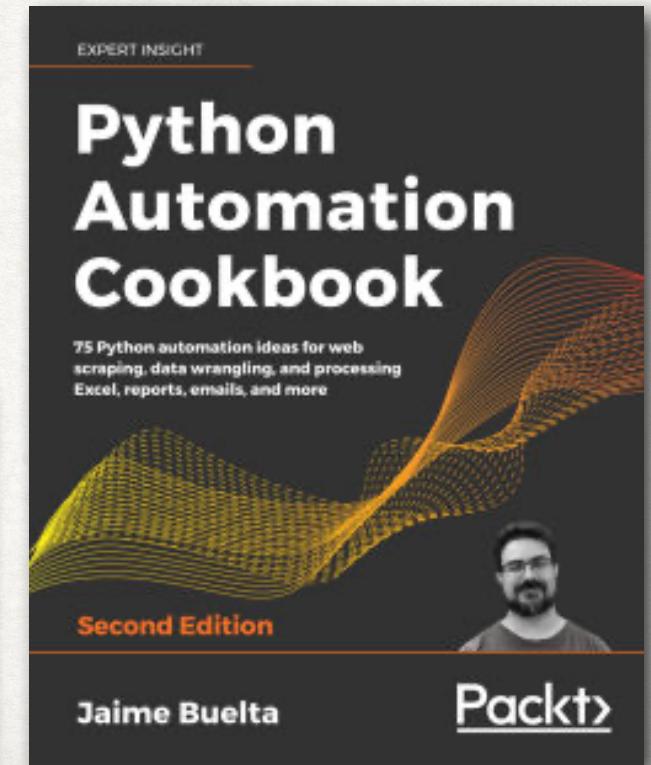
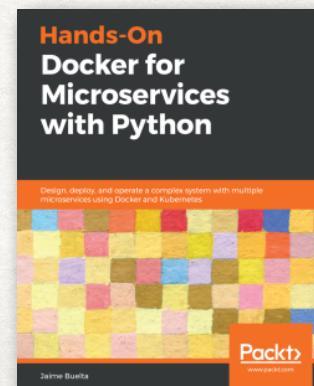
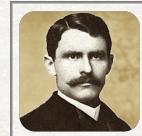
- Software Architect in Double Yard
- Python developer since 2010
- Book author ([Amazon](#))
- Podcast [wrongsideoflife.com](#)



buelta.com



@jaimebuelta



AIM OF THE WORKSHOP

- Get an introduction of the Python possibilities
- Start a journey in useful elements for dealing with data
 - Data input from files
 - Data output to files
 - Useful third-party modules like Matplotlib, Pandas, sh...

SUMMARY

1. Basic Python

2. Dealing with files

3. Efficient data treatment

PREREQUISITES

INSTALL PYTHON 3

Likely already installed! Check running



```
$ python3 --version  
Python 3.9.5
```

- <https://www.python.org/downloads/>

WHY USING PYTHON

- Easy to use. Easy to create prototype and experiment
- Multilanguage support
- Powerful standard library
- Fantastic third-party library
- Integrated environments

GITHUB REPO

https://github.com/jaimebuelta/odsc_2021_python_workshop

BASIC PYTHON

VIRTUAL ENVIRONMENTS



```
$ python3 -m venv venv  
$ source ./venv/bin/activate  
(venv) $
```

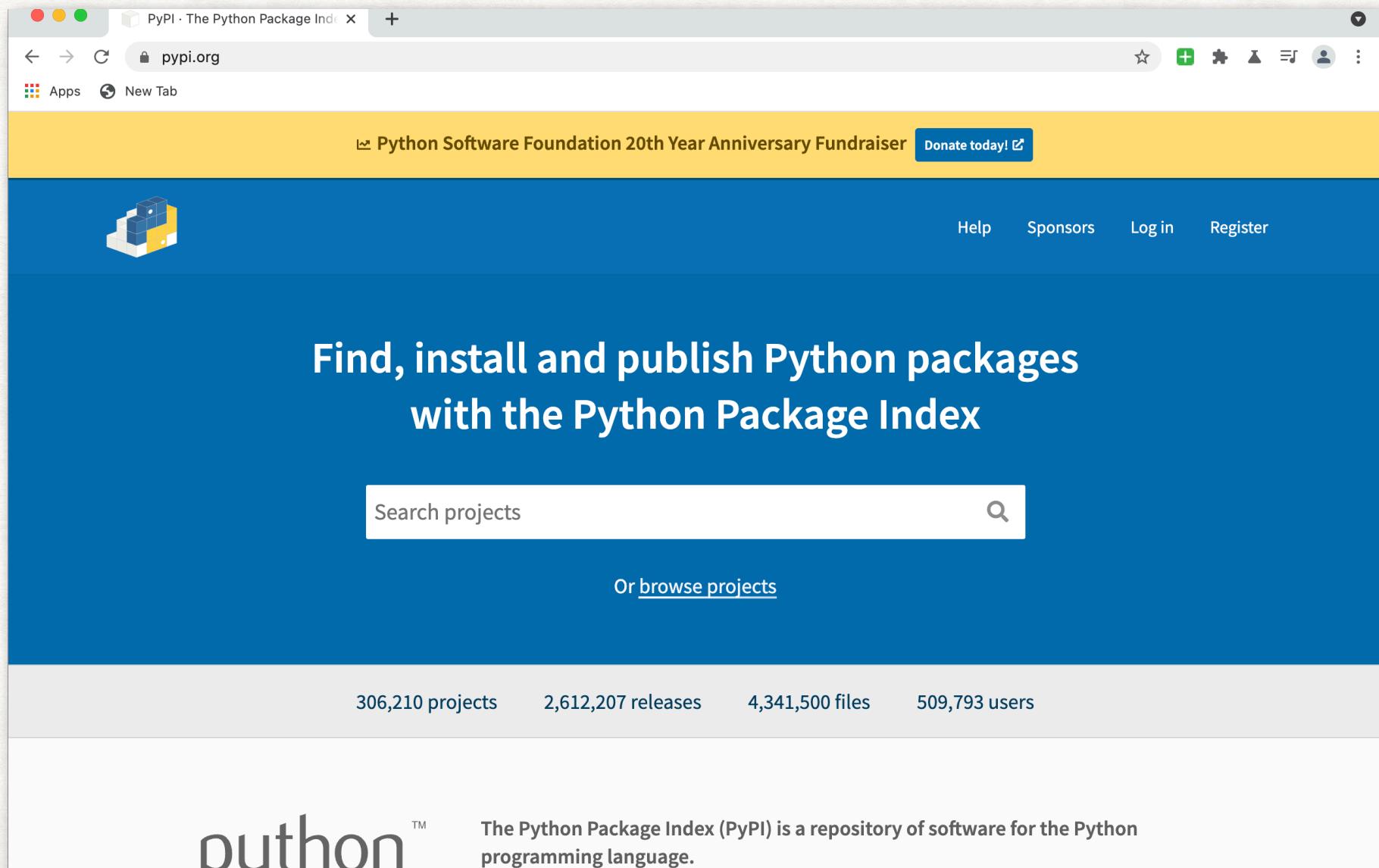
PIP



```
$ pip install --upgrade pip
Requirement already satisfied: pip in ./venv/lib/python3.9/site-packages (21.1.1)
Collecting pip
  Downloading pip-21.1.2-py3-none-any.whl (1.5 MB)
|████████████████████████████████████████████████████████████████| 1.5 MB 1.8 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.1.1
    Uninstalling pip-21.1.1:
      Successfully uninstalled pip-21.1.1
Successfully installed pip-21.1.2
```

PYPI

HTTPS://PYPI.ORG/



A screenshot of a web browser displaying the PyPI homepage. The browser window has a light gray header bar with the title "PyPI · The Python Package Index" and a URL field containing "pypi.org". Below the header is a yellow navigation bar with a "Python Software Foundation 20th Year Anniversary Fundraiser" message and a "Donate today!" button. The main content area has a blue background. It features the PyPI logo (a stylized 3D cube icon) on the left and a large white text block in the center reading "Find, install and publish Python packages with the Python Package Index". Below this is a search bar with the placeholder "Search projects" and a magnifying glass icon. Underneath the search bar is a link "Or browse projects". At the bottom of the page, a light gray footer bar displays statistics: "306,210 projects", "2,612,207 releases", "4,341,500 files", and "509,793 users". The footer also contains the Python logo and a copyright notice: "The Python Package Index (PyPI) is a repository of software for the Python programming language."

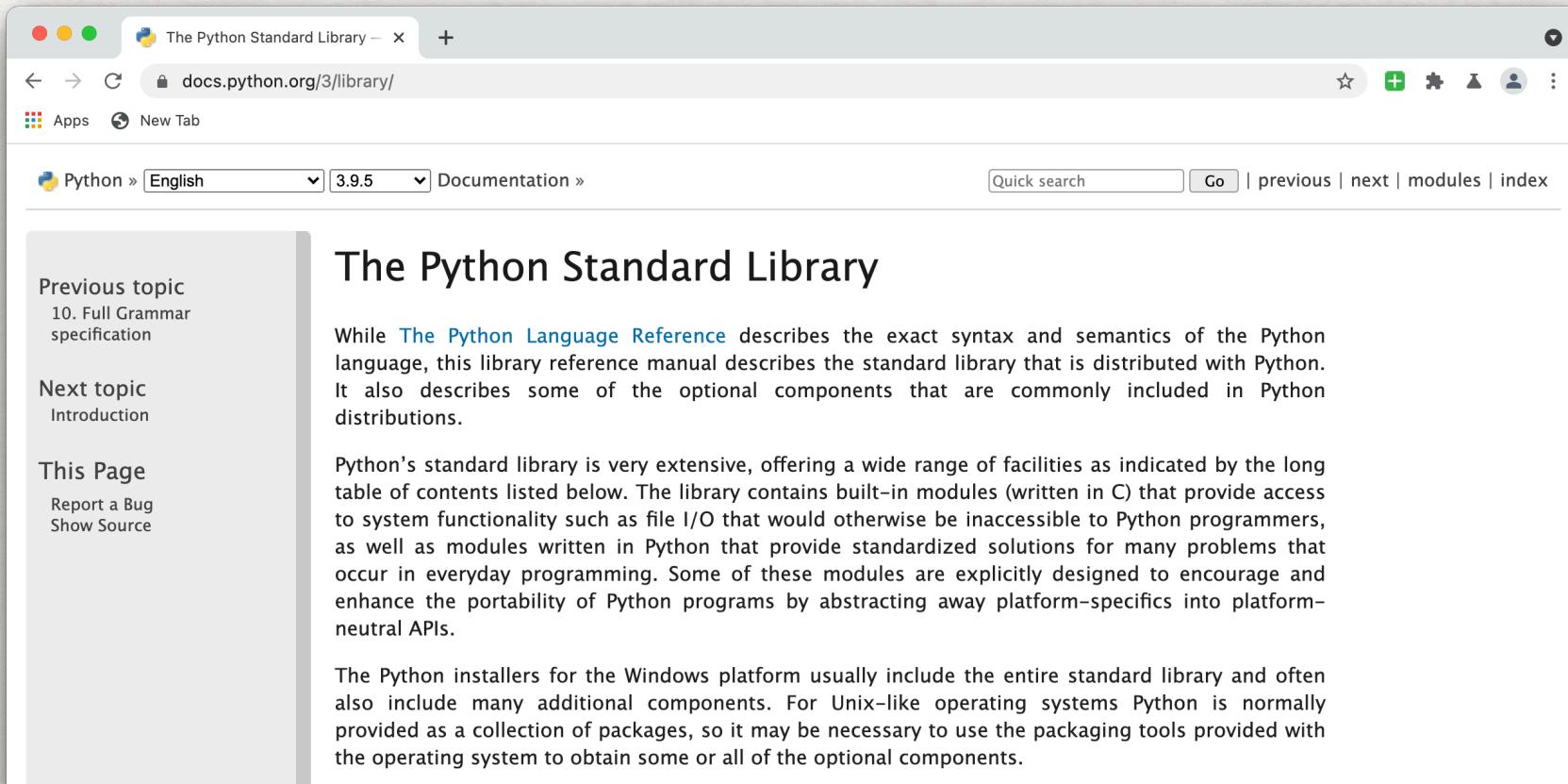
LIST OF EXTERNAL PACKAGES WE WILL USE

- click (<https://pypi.org/project/click8/>)
CLI interfaces
- sh (<https://pypi.org/project/sh/>)
Call external programs
- Pandas (<https://pypi.org/project/pandas/>) Numerical treatment
- Matplotlib (<https://pypi.org/project/matplotlib/>)
Draw graphics
- FPDF (<https://pypi.org/project/fpdf/>)
Create PDF files
- python-docx (<https://pypi.org/project/python-docx/>)
Create .docx files
- mistune (<https://pypi.org/project/mistune/>)
Render Markdown

OTHER INTERESTING PACKAGES

- Requests (to do HTTP requests)
- Django (Web framework)
- Pytest (Tests framework)
- NumPy
- SciPy
- Delorean (dealing with time and date)
- Pint (Work with physical units)

OTHER INTERESTING PACKAGES



The screenshot shows a web browser window with the title "The Python Standard Library" and the URL "docs.python.org/3/library/". The browser interface includes standard controls like back, forward, and search, along with a tab bar showing "New Tab" and "Apps". The main content area displays the Python Standard Library documentation. On the left, there's a sidebar with links for "Previous topic" (10. Full Grammar specification), "Next topic" (Introduction), and "This Page" (Report a Bug, Show Source). The main content starts with the heading "The Python Standard Library" and a brief description of what it covers. It then goes into more detail about the library's extensive facilities and portability. Below that, there's another paragraph about how Python installers handle the standard library.

Previous topic
10. Full Grammar specification

Next topic
Introduction

This Page
Report a Bug
Show Source

The Python Standard Library

While [The Python Language Reference](#) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

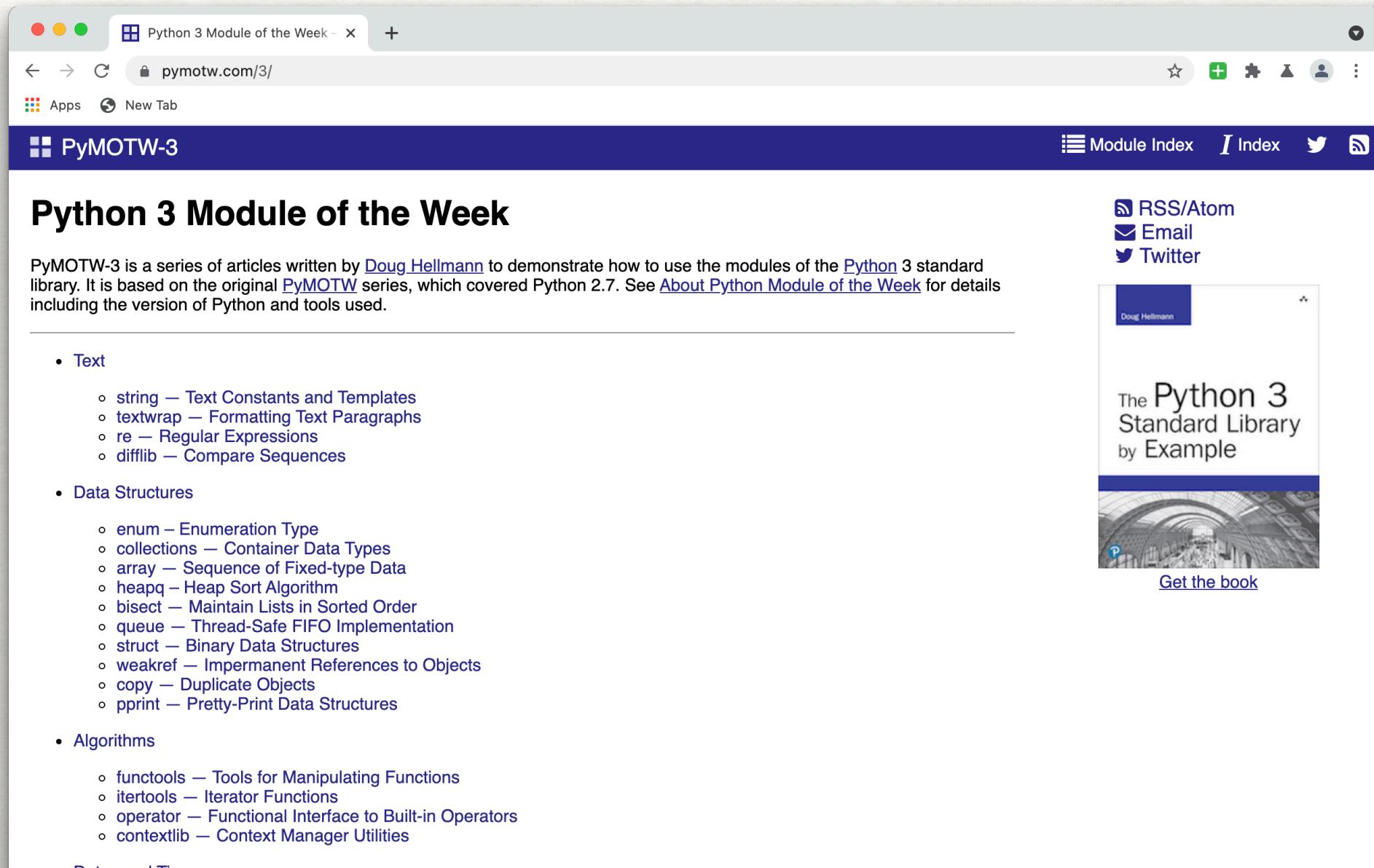
Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

Batteries included!

PYTHON MODULE OF THE WEEK

PYMOTW.COM



The screenshot shows a web browser window with the title "Python 3 Module of the Week" and the URL "pymotw.com/3/" in the address bar. The browser interface includes standard controls like back, forward, and search, along with a toolbar for apps and new tabs.

The main content area has a dark blue header bar with the text "PyMOTW-3" on the left and navigation links for "Module Index", "Index", and social media icons for Twitter and RSS/Atom on the right.

Python 3 Module of the Week

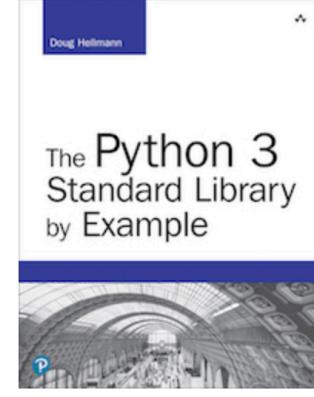
PyMOTW-3 is a series of articles written by [Doug Hellmann](#) to demonstrate how to use the modules of the [Python](#) 3 standard library. It is based on the original [PyMOTW](#) series, which covered Python 2.7. See [About Python Module of the Week](#) for details including the version of Python and tools used.

- [Text](#)
 - [string — Text Constants and Templates](#)
 - [textwrap — Formatting Text Paragraphs](#)
 - [re — Regular Expressions](#)
 - [difflib — Compare Sequences](#)
- [Data Structures](#)
 - [enum — Enumeration Type](#)
 - [collections — Container Data Types](#)
 - [array — Sequence of Fixed-type Data](#)
 - [heapq — Heap Sort Algorithm](#)
 - [bisect — Maintain Lists in Sorted Order](#)
 - [queue — Thread-Safe FIFO Implementation](#)
 - [struct — Binary Data Structures](#)
 - [weakref — Impermanent References to Objects](#)
 - [copy — Duplicate Objects](#)
 - [pprint — Pretty-Print Data Structures](#)
- [Algorithms](#)
 - [functools — Tools for Manipulating Functions](#)
 - [itertools — Iterator Functions](#)
 - [operator — Functional Interface to Built-in Operators](#)
 - [contextlib — Context Manager Utilities](#)

[Dates and Times](#)

[Doug Hellmann](#)

The Python 3 Standard Library by Example



[Get the book](#)

INSTALLING MULTIPLE FILES

```
requirements.txt
---
click
sh
pandas
matplotlib
pint
delorean
```

```
$ pip install -r 01_basic_python/requirements.txt
Collecting sh
    Downloading sh-1.14.2-py2.py3-none-any.whl (40 kB)
        |████████████████████████████████| 40 kB 1.8 MB/s
Collecting pandas
...
Installing collected packages
$
```

CLI INTERFACES

CLI_BASICS.PY



```
import click
import time

@click.command()
@click.argument('number', type=int)
@click.option('--text', default='EXAMPLE')
def main(number, text):
    for _ in range(number):
        click.echo(click.style('echo ', fg='green') + text)
        time.sleep(.2)

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

EXERCISE 01

CREATE A CLI SCRIPT THAT

- Input two parameters:
 - Trigonometric function (sin, cos, tan)
 - Number between 0 and 360
- Returns the value



Tip: Check Python library math

```
$ python3 exercise_01.py sin 0
SIN(0) = 0.0
$ python3 exercise_01.py sin 90
SIN(90) = 1.0
$ python3 exercise_01.py tan 0
TAN(0) = 0.0
$ python3 exercise_01.py cos 0
COS(0) = 1.0
```

BASIC TYPES



```
int = 1
float = 0.3
bool = True | False
string = "some text"
None
```

```
# Groups
dict = {'key': value, 'key2': 'value2' }
list = [value, value2, value]
tuple = (value, value2, value)
set = {value, value2}
```

ADDRESSING ELEMENTS



```
list[2] = value  
tuple[1] == value  
value, value2 = tuple
```

```
'key' in dict  
dict['key']  
dict['key'] = value  
del dict['key']
```

```
value in set  
set.add(value)  
set.remove(value)
```

ITERATING THROUGH GROUP TYPES



```
for i in list:  
    print(i)
```

```
for i in tuple:  
    print(i)
```

```
for key in dict:  
    print(key)
```

```
for key, value in dict.items():  
    print(key, value)
```

```
for i in set:  
    print(i)
```

CALL OTHER PROCESSES

SH



```
import sh

# awk -F "\*,\*'{print $1}' simpsons_data.csv | uniq -c

result = sh.uniq(sh.awk('-F', '\*,\*', '{print $1}', 'simpsons_data.csv'),
                 '-c')
seasons = {}

for line in result:
    if 'Season' in line:
        continue

    number, season = line.split()
    seasons[season] = int(number)

print(seasons)
print(sum(seasons.values()))
```

EXERCISE 2

USE SH TO UNCOMPRESS A FILE

```
$ python3 exercise_02.py
```

Extract zipped_file.zip

DEALING WITH FILES

OPEN FILES



```
with open(filename) as fp:  
    for line in fp:  
        print(line)
```

```
with open(binary_filename, 'rb') as fp:  
    content = fp.read()
```

WRITE FILES



```
with open(filename, 'w') as fp:  
    for line in data:  
        fp.write(line + '\n')
```

```
with open(binary_filename, 'wb') as fp:  
    fp.write(data)
```

ENCODINGS



```
>>> with open('text_iso.txt') as fp:  
...     for l in fp:  
...         print(l)  
...  
...  
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa3 in  
position 2: invalid start byte  
  
>>> with open('text_iso.txt', encoding='iso-8859-1') as fp:  
...     for l in fp:  
...         print(l)  
...  
20£
```

SIMPSONS_DATA.CSV

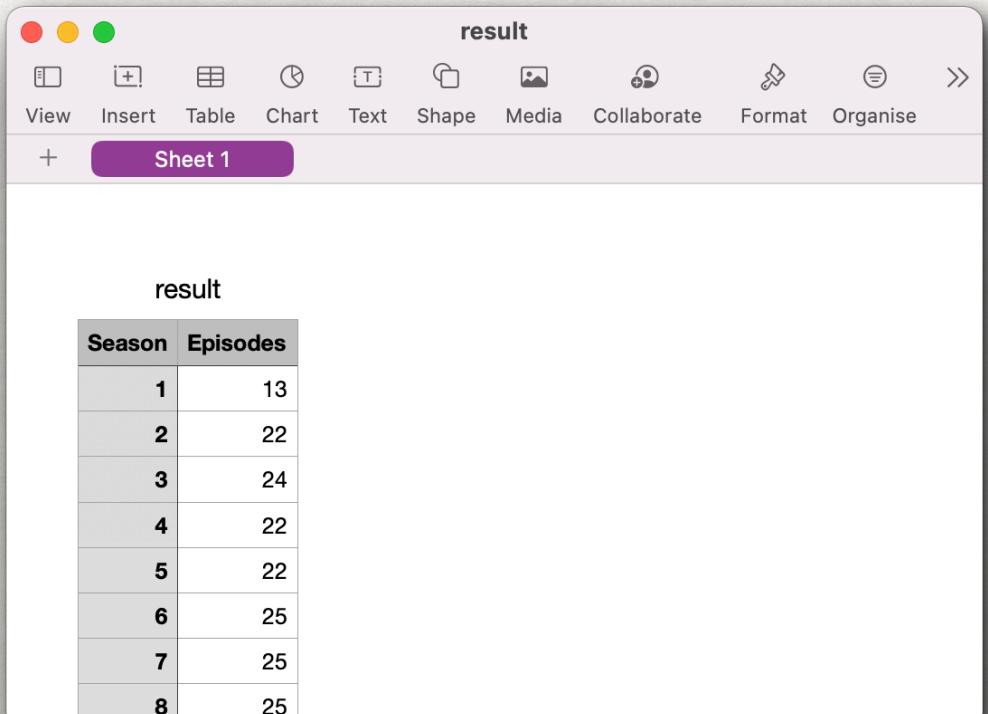
The screenshot shows a spreadsheet application window titled "simpsons_data". The interface includes standard Mac OS X window controls (red, yellow, green buttons) and a toolbar with View, Zoom, Add Category, Insert, Table, Chart, Text, Shape, Media, and Comment buttons. A zoom level of 125% is selected. The main area displays a table with the following data:

Season	Episode	Title	Airdate	Description
1	1	Simpsons Roasting on an Open Fire	17 Dec. 1989	The family is forced to spend all of their time at home.
1	2	Bart the Genius	14 Jan. 1990	Bart ends up at a school for gifted children.
1	3	Homer's Odyssey	21 Jan. 1990	After losing his job, Homer commutes to work by boat.
1	4	There's No Disgrace Like Home	28 Jan. 1990	After being embarrassed by the rest of the town, Homer commutes to work by boat.
1	5	Bart the General	4 Feb. 1990	After being beaten up by Nelson Muntz, Bart commutes to work by boat.
1	6	Moaning Lisa	11 Feb. 1990	A depressed Lisa's spirit is lifted when she commutes to work by boat.
1	7	The Call of the Simpsons	18 Feb. 1990	Homer takes the family camping.
1	8	The Telltale Head	25 Feb. 1990	Bart gets more than he bargained for when he commutes to work by boat.
1	9	Life on the Fast Lane	18 Mar. 1990	Marge contemplates an affair with a haughty man.
1	10	Homer's Night Out	25 Mar. 1990	After a photograph of Homer canoodling with Marge, he commutes to work by boat.
1	11	The Crepes of Wrath	15 Apr. 1990	Bart is sent to France on a student exchange.
1	12	Krusty Gets Busted	29 Apr. 1990	Homer witnesses a robbery at the Kwik-E-Mart.
1	13	Some Enchanted Evening	13 May 1990	Homer and Marge enjoy a night on the town.
2	1	Bart Gets an F	11 Oct. 1990	Mrs. Krabappel is fed up with Bart's poor grades.

EXERCISE 3

Using the file `simpsons_data.csv` in the repo, create a cli script that:

- reads the file
- Determine how many episodes are per season
- Write a file with the info, in CSV format



The screenshot shows a spreadsheet application window titled "result". The toolbar includes standard options like View, Insert, Table, Chart, Text, Shape, Media, Collaborate, Format, and Organise. A purple tab bar at the bottom indicates the active sheet is "Sheet 1". The main area displays a table with two columns: "Season" and "Episodes". The data is as follows:

Season	Episodes
1	13
2	22
3	24
4	22
5	22
6	25
7	25
8	25

CREATE A GRAPHIC

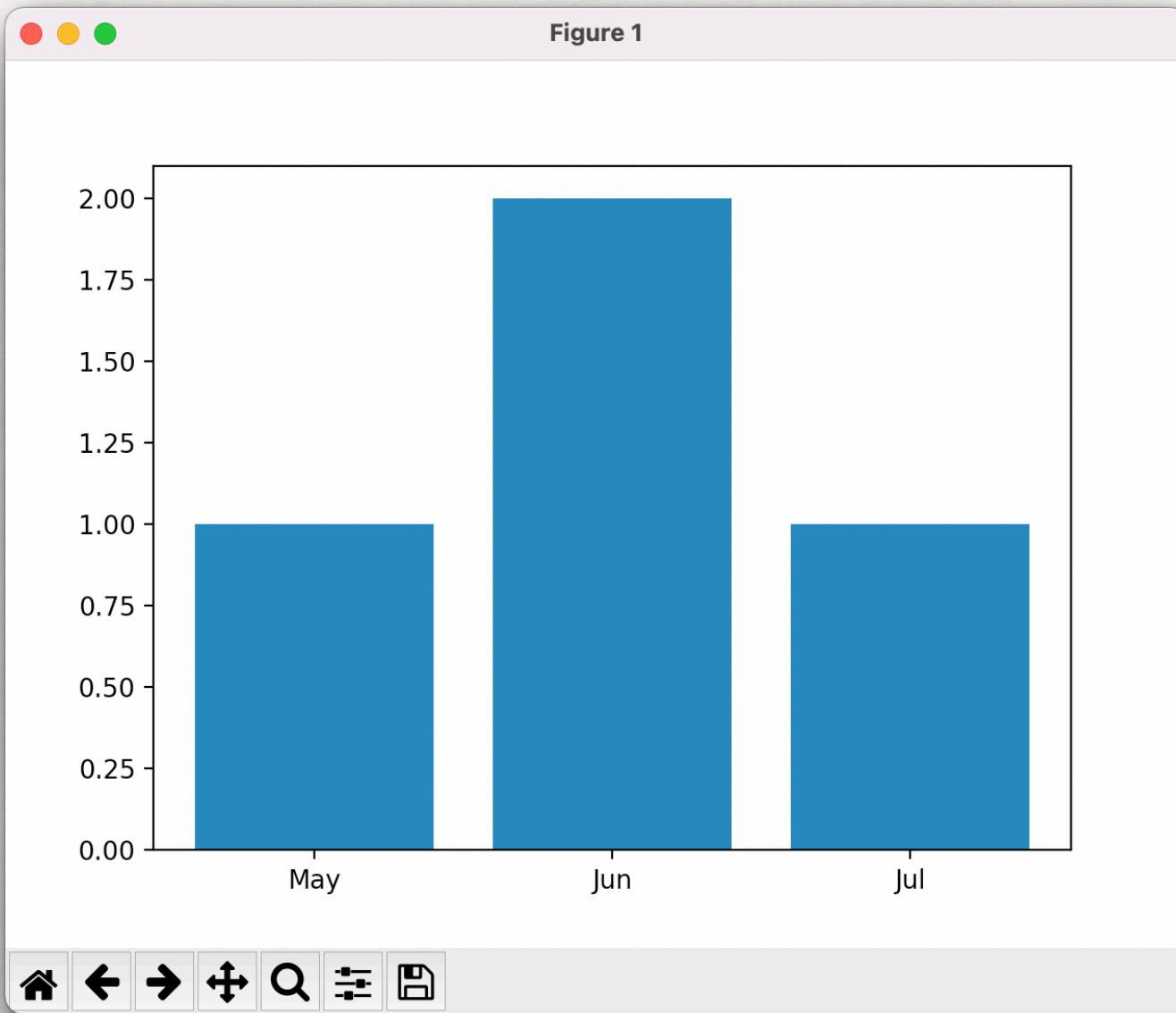
MATPLOTLIB

```
>>> import matplotlib.pyplot as plt

>>> pos = [0, 1, 2]
>>> values = [1, 2, 1]
>>> labels = ['May', 'Jun', 'Jul']

>>> plt.bar(pos, values)
>>> plt.xticks(pos, labels)
...
>>> plt.show()
```

MATPLOTLIB



MATPLOTLIB

TO SAVE

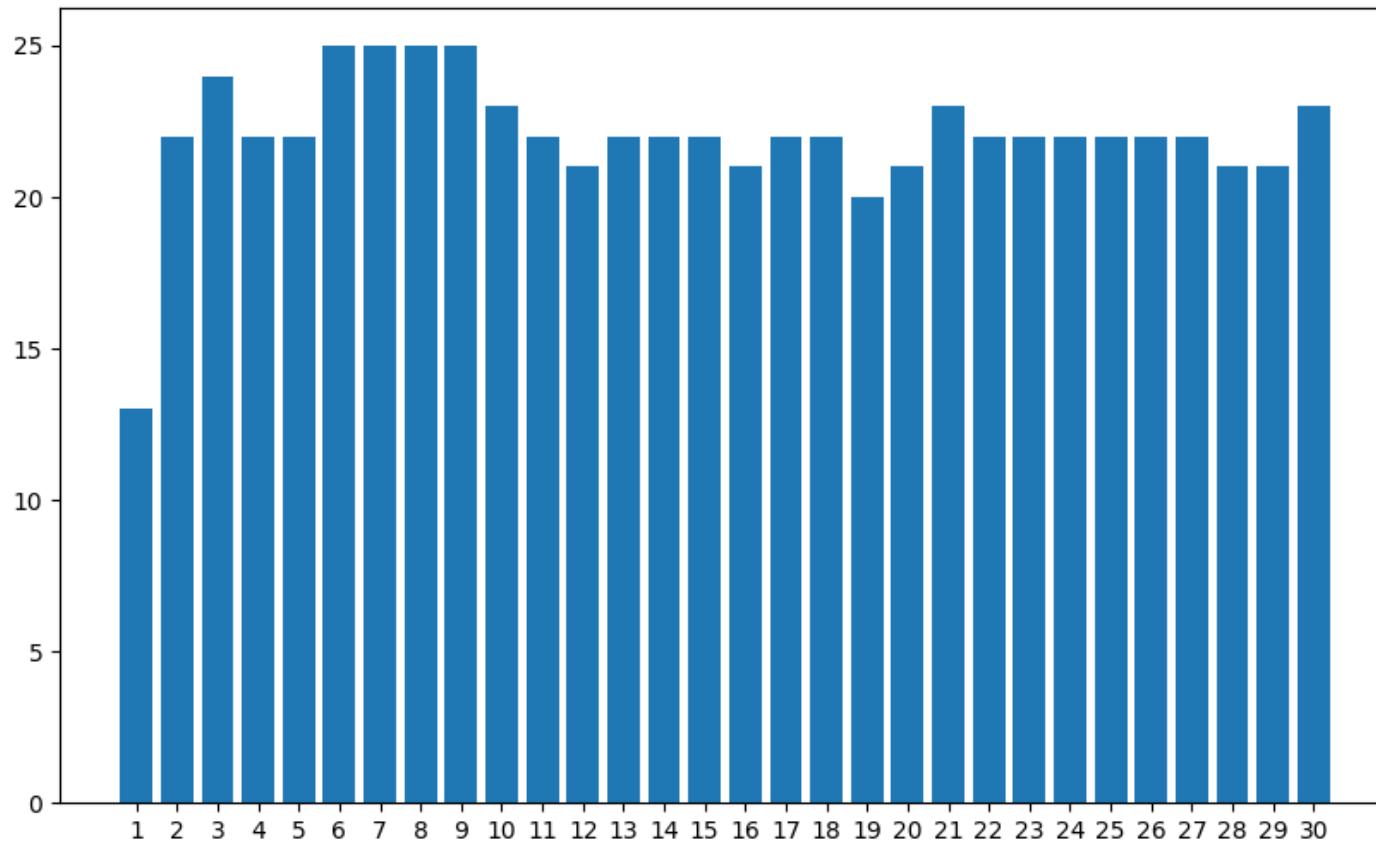


```
# Adjust size
>>> figure = plt.gcf()
>>> figure.set_size_inches(5, 5)
# save
>>> plt.savefig('result.png')
```

EXERCISE 4

WRITE A PNG FILE WITH SEASON DATA

```
$ python3 exercise_04.py simpsons_data.csv result.png
```



WRITE A PDF

FPDF



```
document = fpdf.FPDF()
document.set_font('Times', '', 11)
document.add_page()

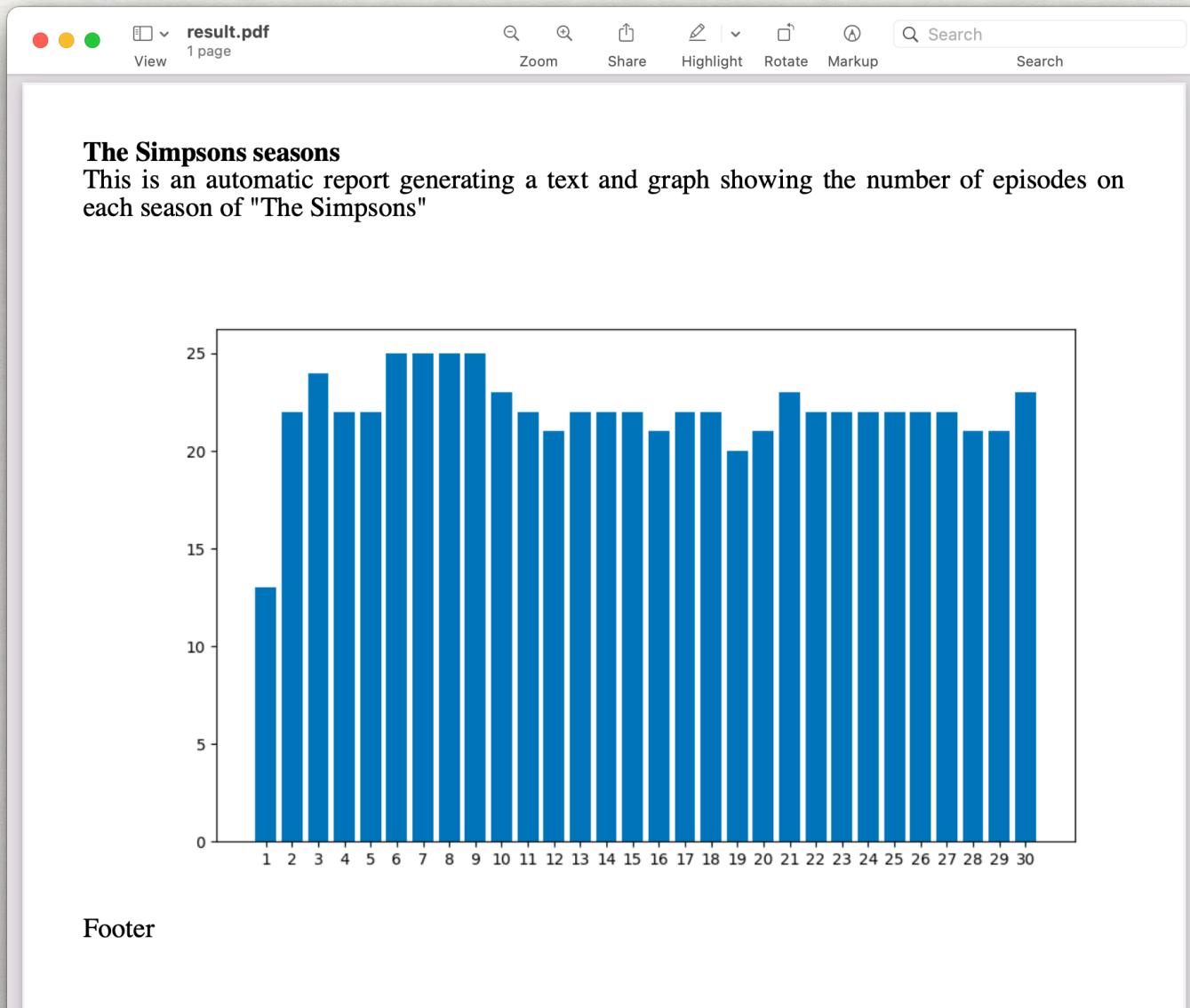
document.cell(0, 5, 'Example text')
document.ln()

document.output(file_out)
```

EXERCISE 5

WRITE A PDF WITH SOME TEXT AND THE GRAPH

```
$ python3 exercise_05.py simpsons_data.csv result.pdf
```



CREATE HTML TEXT THROUGH MARKDOWN AND MISTUNE



Template in Markdown

Create a template and keep the {key} values in ` `.format() ` way,
to transform them

```

```
>> TEMPLATE = 'This is text with {element}'
>> result = TEMPLATE.format(element='replaced')
>> result
'This is text with replaced'
```
```

MISTUNE



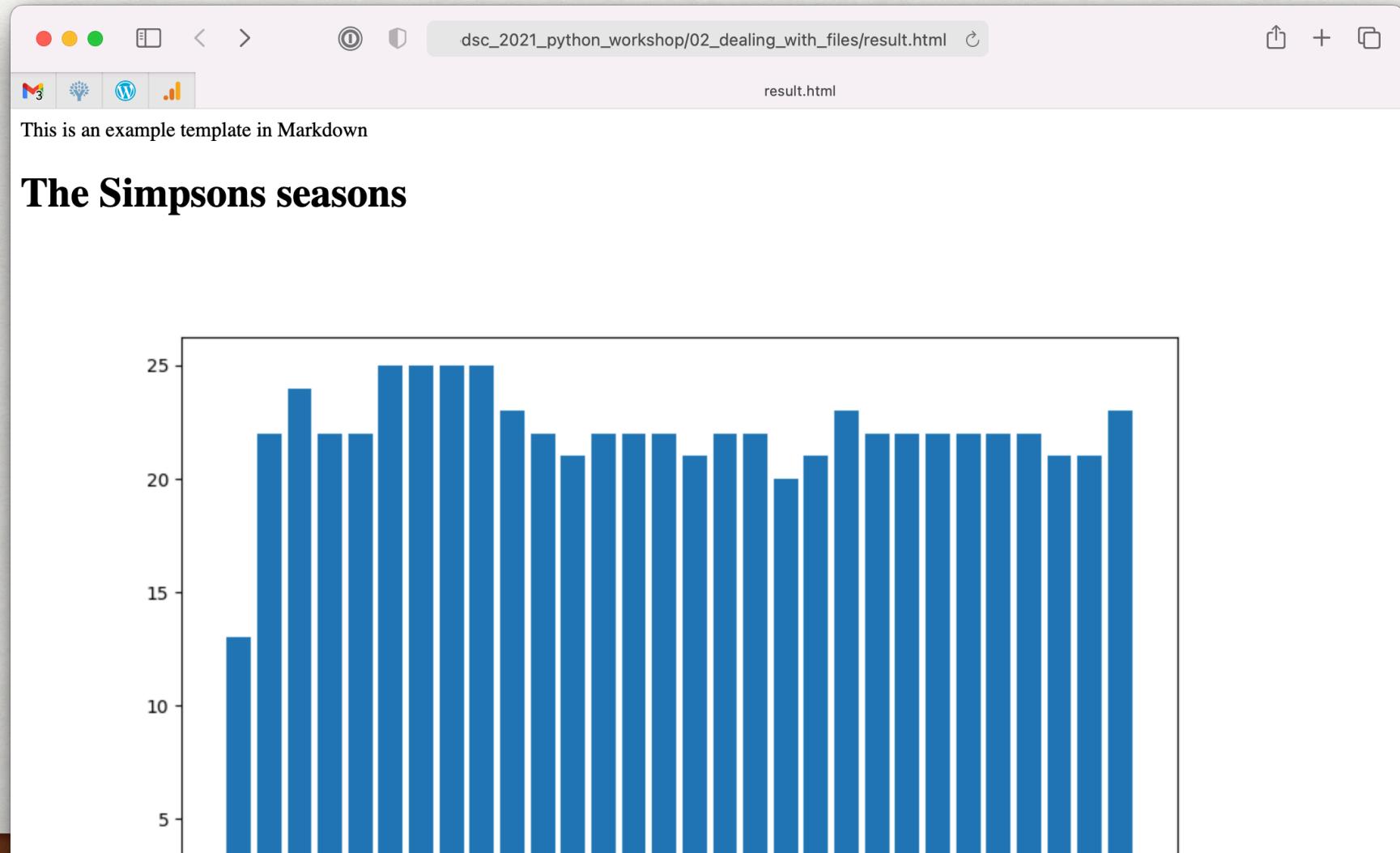
```
import mistune
```

```
html = mistune.markdown(markdown)
```

EXERCISE 6

CREATE AN HTML REPORT

```
$ python3 exercise_06.py simpsons_data.csv template.md result.html
```



CREATE WORD TEXT

PYTHON-DOCX



```
import docx

document = docx.Document( )

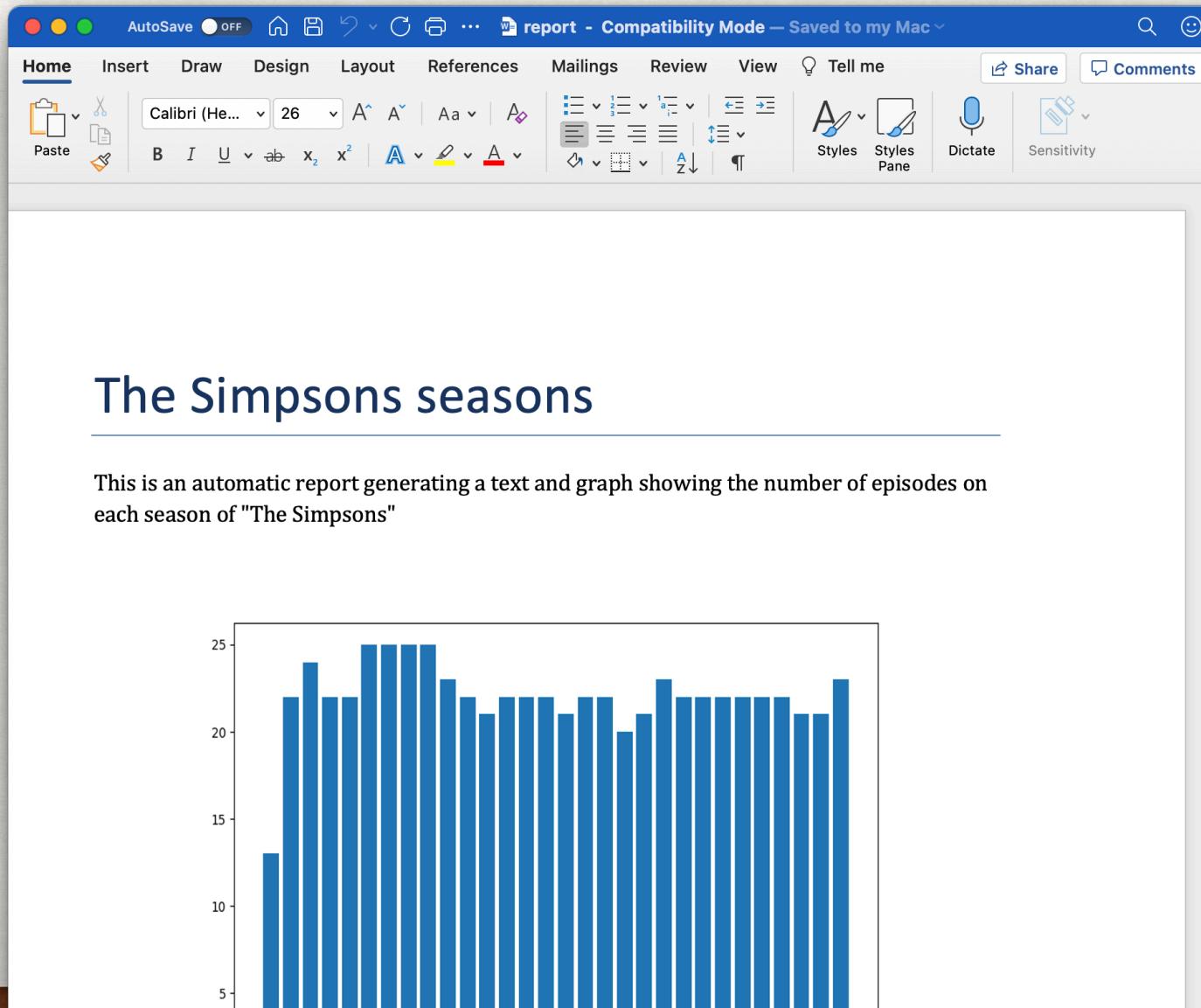
paragraph = document.add_paragraph()
paragraph.add_run('Each run is a small piece of text. ')
r = paragraph.add_run('Multiple can be added and styled indepently')
r.font.name = 'Arial'

document.save('result.docx')
```

EXERCISE 7

CREATE WORD REPORT

```
$ python3 exercise_07.py simpsons_data.csv result.docx
```



EFFICIENT DATA TREATMENT

SPECIALISED DATA TYPES

- Tuples
- Array
- Deques
- Data classes

ARRAY

NUMERIC ARRAYS



```
>>> from array import array
# 'I' is for unsigned int
>>> myarray = array('I')
>>> myarray.extend([1,2,3])
>>> myarray.append('a')
...
TypeError: an integer is required (got type str)
>>> myarray.append(-1)
...
OverflowError: can't convert negative value to unsigned int
```

MUCH MORE MEMORY EFFICIENT



```
myarray = list(range(100_000_000))

myarray = array('I', range(100_000_000))
```



```
$ python3 array_memory_test.py
Current memory usage is 410.177592MB; Peak was 410.177608MB
$ python3 list_memory_test.py
Current memory usage is 3599.992996MB; Peak was 3599.992996MB
```

DEQUES

STRUCTURES OPTIMISED FOR ADDING AND REMOVING ELEMENTS ON EACH END

```
● ● ●  
from collections import deque  
  
myqueue = deque(range(100_000_000))  
myqueue = list(range(100_000_000))  
  
for _ in range(10):  
    myqueue.append(1)  
    myqueue.pop(0)      myqueue.popleft()  
    len(myqueue)
```



```
$ python3 deque_appending.py  
Time is 7 us  
$ python3 list_appending.py  
Time is 1243022 us
```

EXERCISE 8

- Moving averages using a deque, using `average_values.txt`



```
$ cat average_values.txt  
67  
45  
39  
...
```



```
$ python3 moving_average.py  
67, 45, 39, 87, 36, 71, 61, 19, 71, 62  
, , 50, 57, 54, 64, 56, 50, 50, 50
```

NAMEDTUPLE

HELPS ORGANISING DATA INTO LIGHT OBJECTS



```
from collections import namedtuple

City = namedtuple('City', 'name, population, area_km2')

dublin = City(name='Dublin', population=554_554, area_km2=117.8)
toledo = City(name='Toledo', population=84_282, area_km2=232.1)
```

DATA CLASSES

USEFUL FOR OOP



```
from dataclasses import dataclass

@dataclass
class City:
    name: str
    population: int
    area_km2: float
    capital: bool = False

dublin = City(name='Dublin', population=554_554, area_km2=117.8, capital=True)
toledo = City(name='Toledo', population=84_282, area_km2=232.1)
bad = City(name='Toledo', population='84_282', area_km2=232.1)
```

PANDAS INTRO

PANDAS_EXAMPLE.PY



```
import pandas as pd
from dataclasses import dataclass
import matplotlib.pyplot as plt

@dataclass
class City:
    name: str
    population: int
    area_km2: float

    def pandas(self):
        data = {
            'name': self.name,
            'population': pd.to_numeric(self.population),
            'area_km2': pd.to_numeric(self.area_km2),
        }
        return data

cities = [
    City(name='Dublin', population=554_554, area_km2=117.8),
    City(name='Toledo', population=84_282, area_km2=232.1),
]
```

PANDAS INTRO



```
data = pd.DataFrame(city.pandas() for city in cities)
print(data)
data = data.set_index('name')
print(data)

# Create a figure to store the graph
fig, axs = plt.subplots(figsize=(12, 8))

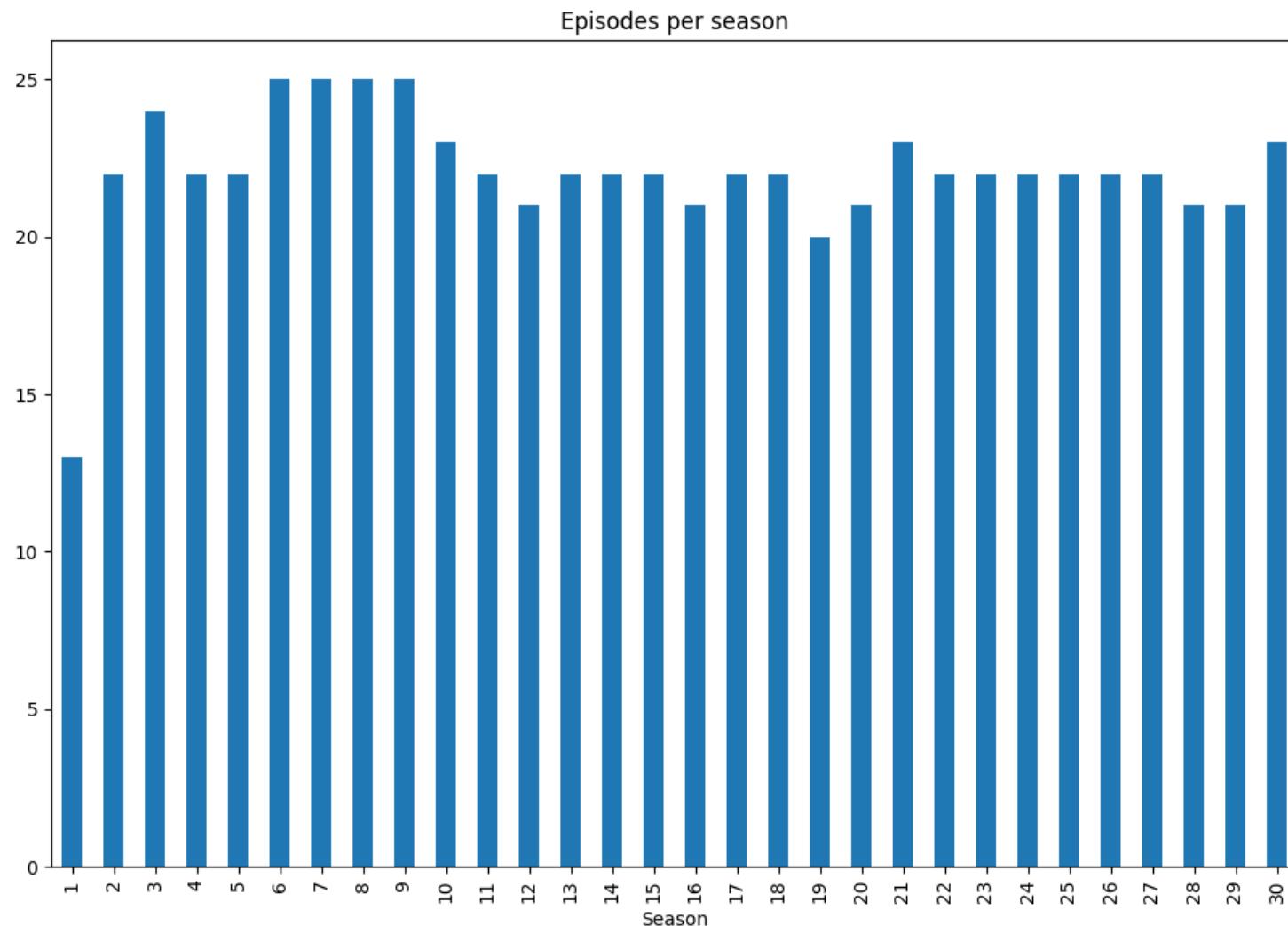
# Create a bar graph in that area
data['population'].plot.bar(ax=axs)
plt.title('Population')

# Save the result in a png
fig.savefig('cities_population.png')

# Aggregate results
result = data.agg(['sum', 'count'])
print(result)
```

EXERCISE 9

```
$ python3 seasons.py simpsons_data.csv simpsons.png
```



SPEEDING UP PYTHON WITH CYTHON

- Cython uses pyx files with pseudo Python code that can be compiled into equivalent C code.
- This speeds up significantly their execution

EXAMPLE

PRIMES.PY



```
NUM_PRIMES = 50000
```

```
def check_if_prime(number):  
  
    for i in range(2, number):  
        if number % i == 0:  
            return False  
  
    return True  
  
if __name__ == '__main__':  
    primes = [number for number in range(2, NUM_PRIMES)  
              if check_if_prime(number)]  
    print(primes)
```

EXAMPLE WITH CYTHON



```
# Will compile the code the first time is imported
import pyximport
pyximport.install()

from cython_primes import check_if_prime

NUM_PRIMES = 50000

if __name__ == '__main__':
    primes = [number for number in range(2, NUM_PRIMES)
              if check_if_prime(number)]
    print(primes)
```

CYTHON CODE

cython_primes.pyx



```
def check_if_prime(int number):
    cdef int i = 2

    while i < number:
        if number % i == 0:
            return False
        i += 1

    return True
```

regular code



```
def check_if_prime(number):
    for i in range(2, number):
        if number % i == 0:
            return False

    return True
```

TIME COMPARISON

REMEMBER TO RUN PRIMES_C.PY TWICE!



```
$ time python3 primes.py
real    0m6.846s
user    0m6.720s
sys     0m0.033s
```

```
$ time python3 primes_c.py
...
real    0m0.570s
user    0m0.405s
sys     0m0.049s
```

SUMMARY

1. Basic Python

- Using venv
- Third-party packages
- Command Line Interface
- Run external commands

2. Dealing with files

- Open and close files
- Deal with CSV files
- Create graphics
- Write PDFs, Word files and HTML

3. Efficient data treatment

- Data types
- Intro to Pandas
- Cython

THANK YOU

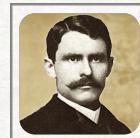
THANK YOU



buelta.com



@jaimebuelta



Podcast wrongsideoflife.com

