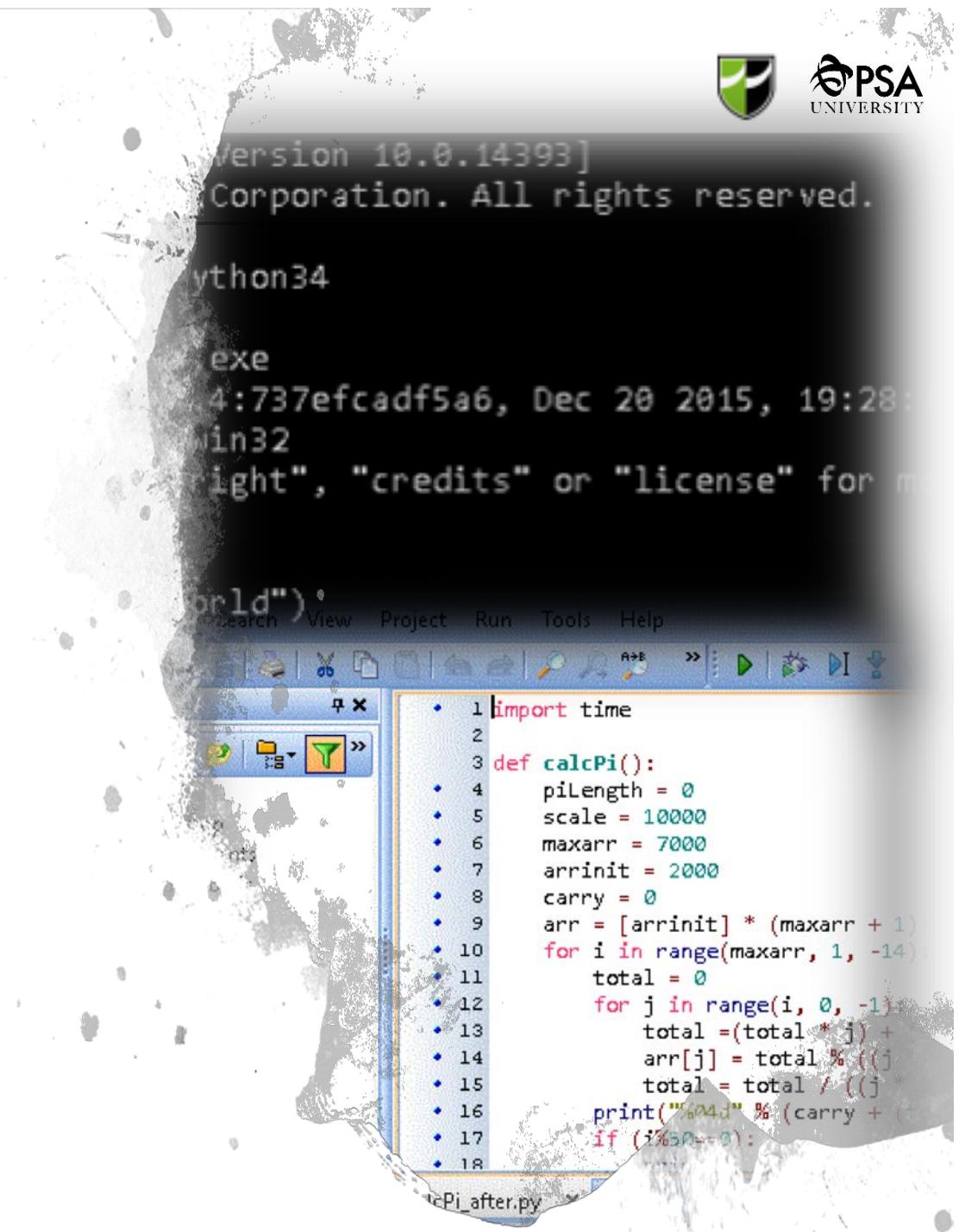


Introductory Programming in Python

Day 2

Programme Day One

- A brief history of Python
- Setting up python environment
- Learn the basics:
 - Data types
 - Conversions
 - String operations
 - Functions
 - And more!
- String functions, formatting
- Find files by name, by extension, by size, by content and calculate the total size



The screenshot shows a Windows desktop environment. In the background, a command prompt window is open with the following text:

```
version 10.0.14393]
Corporation. All rights reserved.

python34
exe
4:737efcadf5a6, Dec 20 2015, 19:28:
in32
ight", "credits" or "license" for m
```

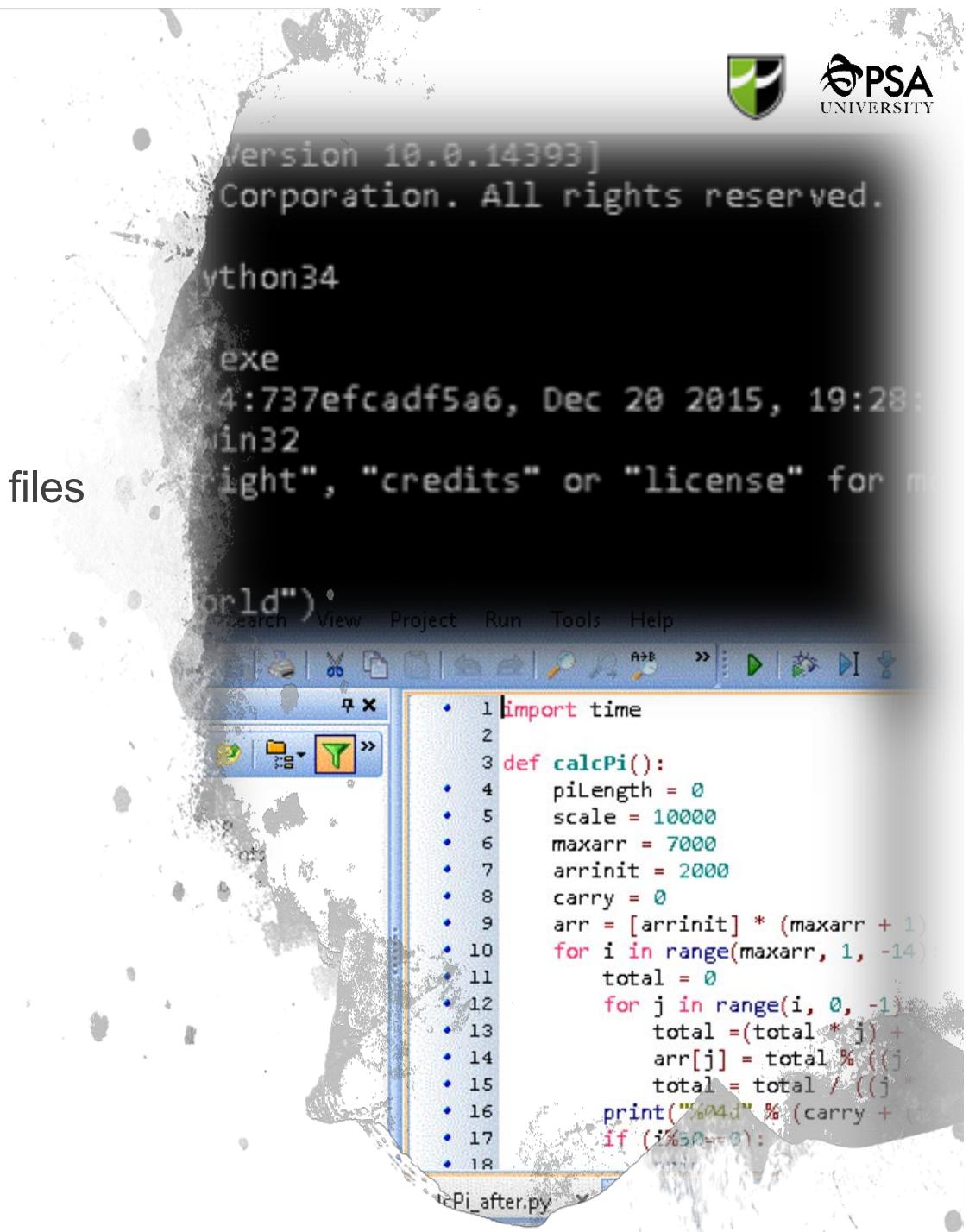
In the foreground, a Python code editor window titled "cpi_after.py" is displayed. The code is a script to calculate pi using a specific algorithm. The code is as follows:

```
import time
def calcPi():
    pilength = 0
    scale = 10000
    maxarr = 7000
    arrinit = 2000
    carry = 0
    arr = [arrinit] * (maxarr + 1)
    for i in range(maxarr, 1, -14):
        total = 0
        for j in range(i, 0, -1):
            total = (total * j) +
        arr[j] = total % ((j *
        total = total / ((j *
    print("%04d" % (carry +
if (i%50==0):
```

What is a python project you would like to work on after this training?

Programme Day Two

- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Processing CSV files
- Generating PDF
- Image processing
- Data Visualization - Charting
- Connecting to the Web
- Sending emails
- Telegram bot



The screenshot shows a Python code editor with a script titled 'calcPi_after.py'. The code implements the Bailey–Borwein–Plouffe algorithm to calculate pi. It includes imports for time, defines a function calcPi, and uses nested loops to calculate the digits of pi.

```
import time
def calcPi():
    pilength = 0
    scale = 10000
    maxarr = 7000
    arrinit = 2000
    carry = 0
    arr = [arrinit] * (maxarr + 1)
    for i in range(maxarr, 1, -14):
        total = 0
        for j in range(i, 0, -1):
            total = (total * j) +
            arr[j] = total % ((j *
            total = total / ((j *
            print("%04d" % (carry +
            if (i%10==0):
```

File Paths

- Absolute & Relative file paths
- Use os.getcwd()

Absolute file paths are notated by a **leading forward slash or drive label**.

For example,

/home/example_user/example_directory

or

C:/system32/cmd.exe

An absolute file path describes how to access a given file or directory, starting from the root of the file system. A file path is also called a *pathname*.

Relative file paths are notated by a **lack of a leading forward slash**.

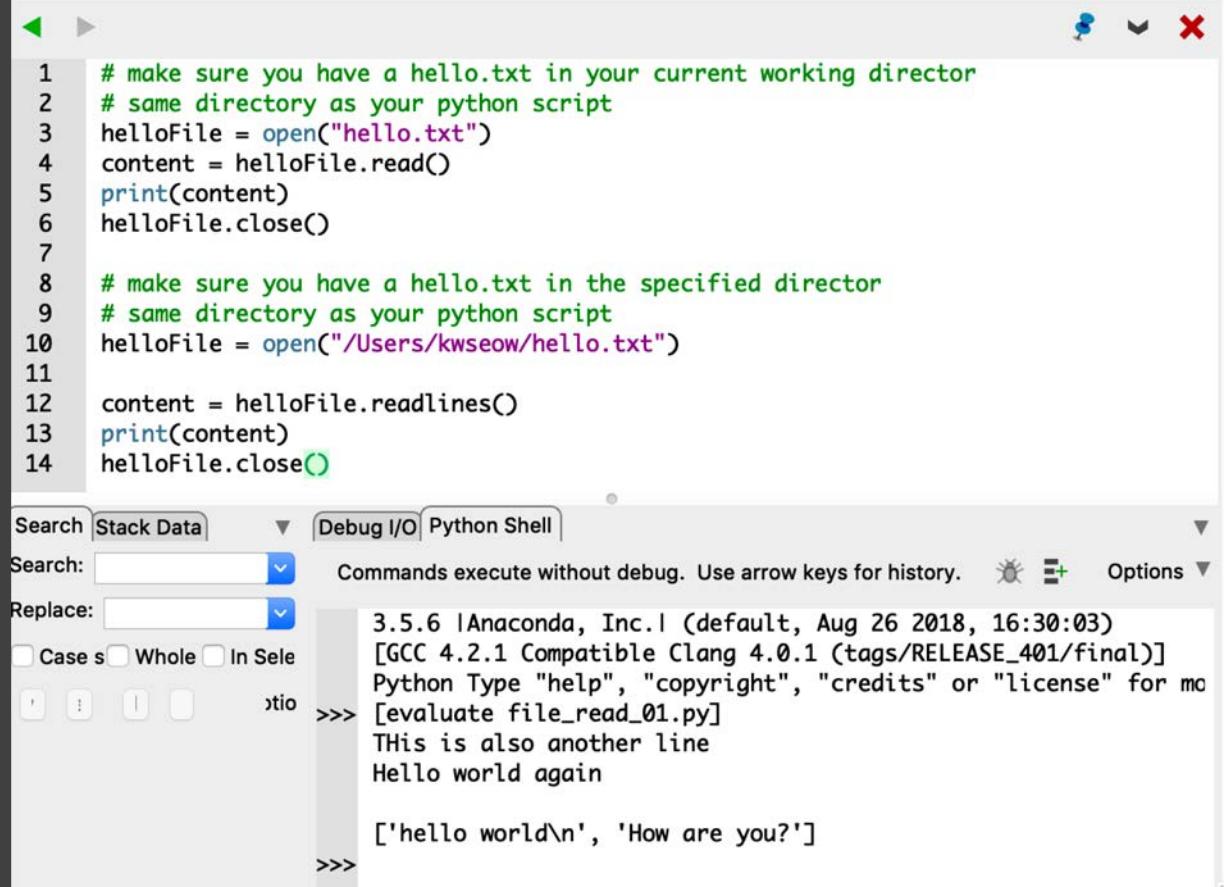
For example,

example_directory.

A relative file path is interpreted from the perspective your current working directory. If you use a relative file path from the wrong directory, then the path will refer to a different file than you intend, or it will refer to no file at all..

Read files

- `Open()` will return a file object which has reading and writing related methods
- Pass ‘r’ (or nothing) to `open()` to open the file in read mode.
- Call `read()` to read the contents of a file
- Call `readlines()` to return a list of strings of the file’s content.
- Call `close()` when you are done with the file.



```

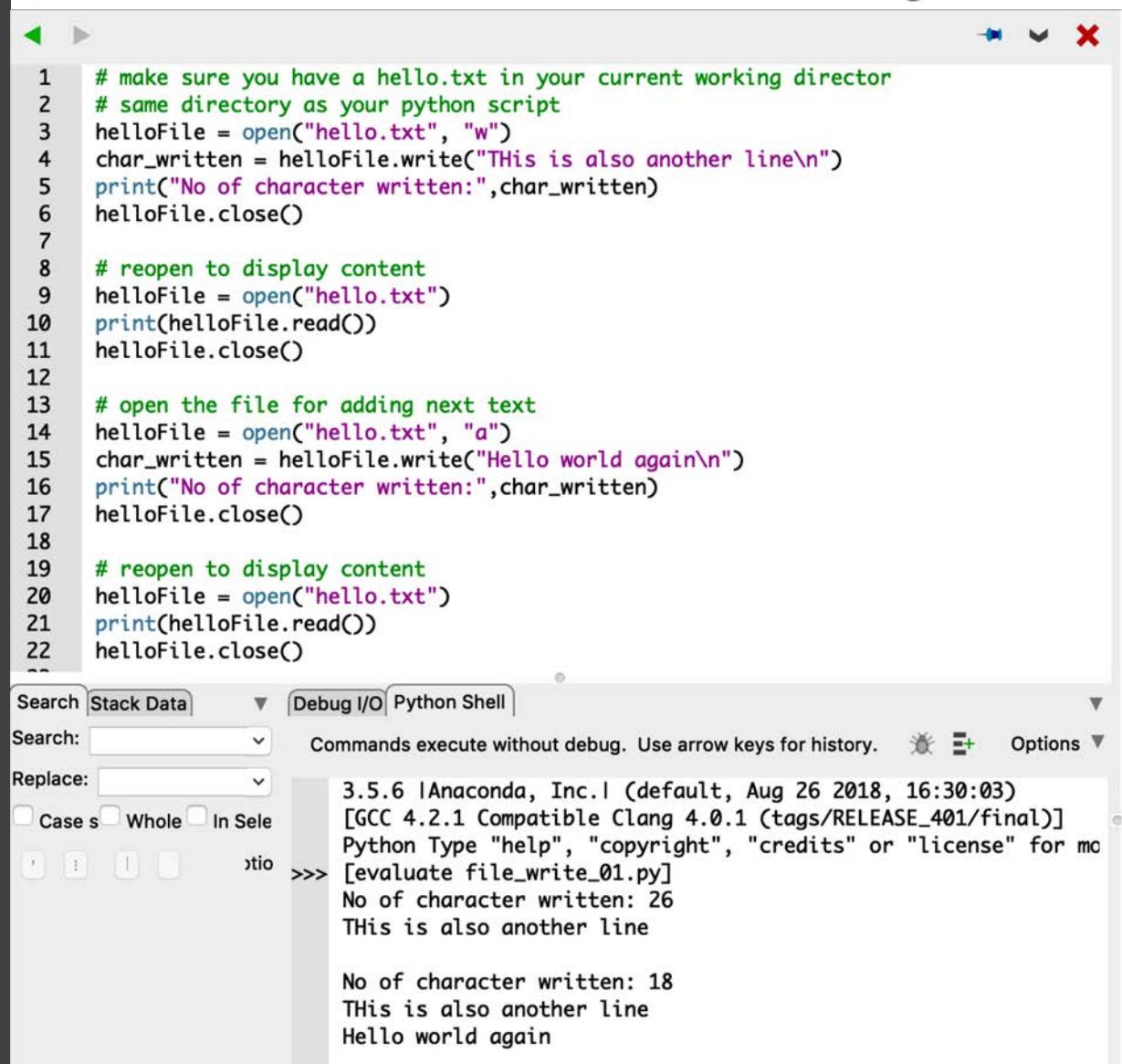
1 # make sure you have a hello.txt in your current working director
2 # same directory as your python script
3 helloFile = open("hello.txt")
4 content = helloFile.read()
5 print(content)
6 helloFile.close()
7
8 # make sure you have a hello.txt in the specified director
9 # same directory as your python script
10 helloFile = open("/Users/kwseow/hello.txt")
11
12 content = helloFile.readlines()
13 print(content)
14 helloFile.close()

```

The screenshot shows a code editor window with a Python script. The script reads a file named 'hello.txt' from the current directory or a specified path. It prints the content of the file to the console. Below the code editor is a Python Shell tab where the script is run. The shell shows the command history, the Python version (3.5.6), compiler information (GCC 4.2.1 Compatible Clang 4.0.1), and the output of the script. The output includes the text 'Hello world again' and the list ['hello world\n', 'How are you?'].

Write files

- Pass ‘w’ to open() to open the file in write mode or ‘a’ for append mode
- Opening a non-existent file in write or append mode will create that file
- Call write() to write a string to a file



```

1 # make sure you have a hello.txt in your current working director
2 # same directory as your python script
3 helloFile = open("hello.txt", "w")
4 char_written = helloFile.write("THis is also another line\n")
5 print("No of character written:",char_written)
6 helloFile.close()
7
8 # reopen to display content
9 helloFile = open("hello.txt")
10 print(helloFile.read())
11 helloFile.close()
12
13 # open the file for adding next text
14 helloFile = open("hello.txt", "a")
15 char_written = helloFile.write("Hello world again\n")
16 print("No of character written:",char_written)
17 helloFile.close()
18
19 # reopen to display content
20 helloFile = open("hello.txt")
21 print(helloFile.read())
22 helloFile.close()

```

Search Stack Data Debug I/O Python Shell

Search: Commands execute without debug. Use arrow keys for history.

Replace: Options

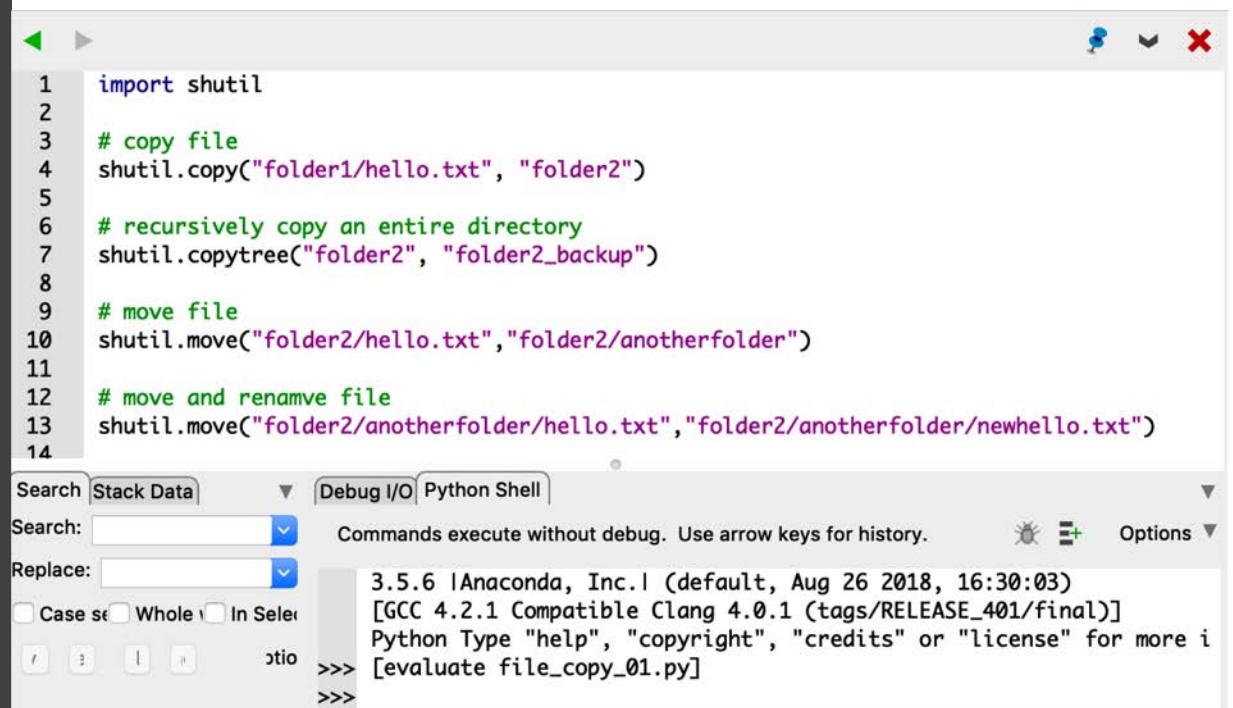
Case s Whole In Sele

>>> 3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" for more
[evaluate file_write_01.py]
No of character written: 26
THis is also another line

No of character written: 18
THis is also another line
Hello world again

Copying and Moving Files

- `shutil.copy(src, dst)` – Copy the file `src` to the file or directory `dst`
- `shutil.copytree(src, dst)` - Recursively copy an entire directory tree rooted at `src`.
- `shutil.move(src, dst)` - Recursively move a file or directory (`src`) to another location (`dst`).



```

1 import shutil
2
3 # copy file
4 shutil.copy("folder1/hello.txt", "folder2")
5
6 # recursively copy an entire directory
7 shutil.copytree("folder2", "folder2_backup")
8
9 # move file
10 shutil.move("folder2/hello.txt", "folder2/anotherfolder")
11
12 # move and rename file
13 shutil.move("folder2/anotherfolder/hello.txt", "folder2/anotherfolder/newhello.txt")
14

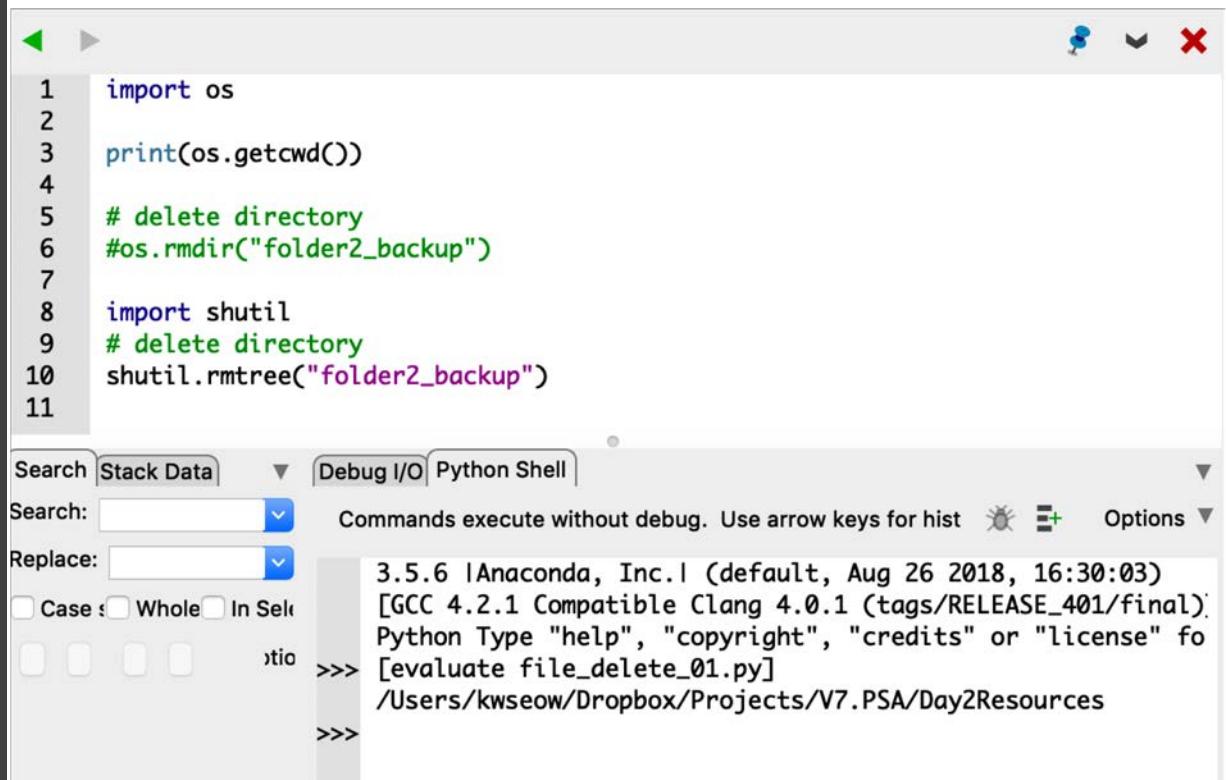
```

The screenshot shows a Python code editor window with the following details:

- Code Area:** Displays the provided Python script.
- Toolbars:** Standard Python IDLE-style toolbars for file operations.
- Search and Replace:** Search bar and replace bar.
- Python Shell:** A terminal-like interface where commands can be run. It shows the Python version (3.5.6), build date (Aug 26 2018), and compiler information (GCC 4.2.1 Compatible Clang 4.0.1). It also indicates that commands execute without debug and provides history navigation via arrow keys.
- Output Area:** Shows the command prompt (>>>) and the output of the executed code, which includes the creation of a backup directory and the movement of files.

Deleting Files

- `os.unlink()` will delete a file
- `os.rmdir()` will delete a folder (but folder must be empty)
- `shutil.rmtree()` will delete a folder and all its contents
- Deleting can be dangerous, so do a dry run first



```

1 import os
2
3 print(os.getcwd())
4
5 # delete directory
6 #os.rmdir("folder2_backup")
7
8 import shutil
9 # delete directory
10 shutil.rmtree("folder2_backup")
11

```

Search Stack Data Debug I/O Python Shell

Search: Commands execute without debug. Use arrow keys for hist Options

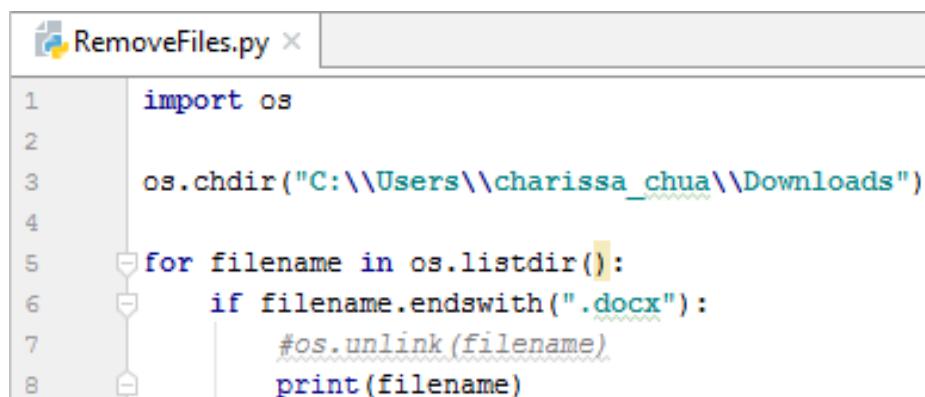
Replace:

< Case < Whole < In Selection

>>> [evaluate file_delete_01.py]

/Users/kwseow/Dropbox/Projects/V7.PSA/Day2Resources

>>>



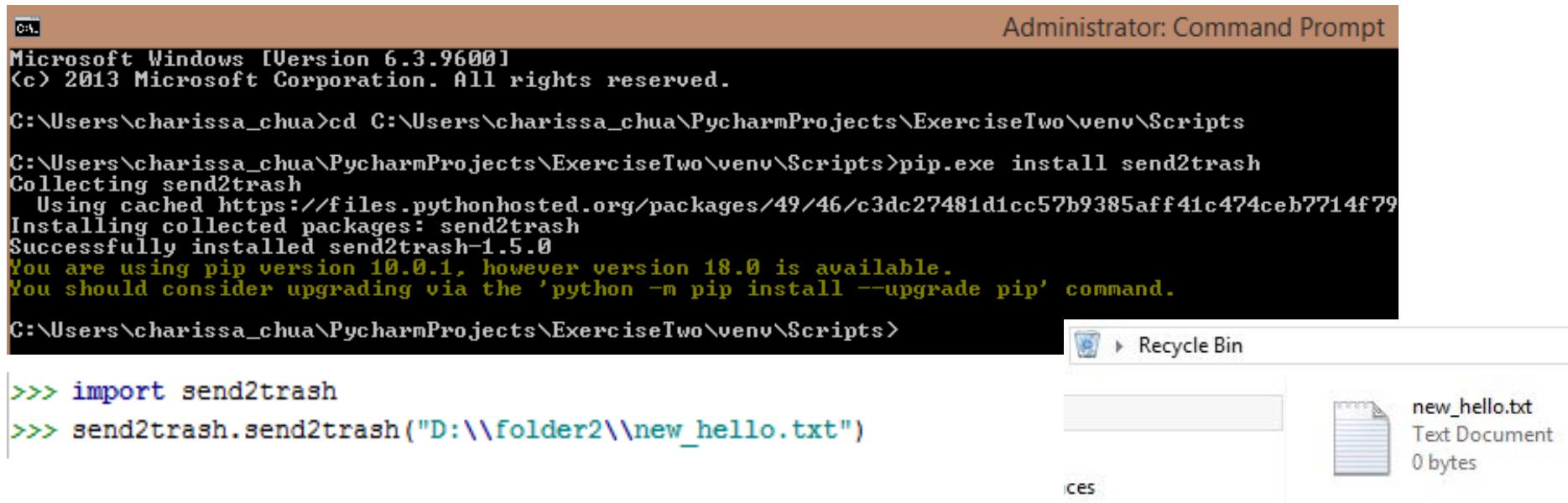
```

RemoveFiles.py x
1 import os
2
3 os.chdir("C:\\\\Users\\\\charissa_chua\\\\Downloads")
4
5 for filename in os.listdir():
6     if filename.endswith(".docx"):
7         #os.unlink(filename)
8         print(filename)

```

send2trash module

- Install send2trash module using pip.exe
- send2trash.send2trash() will send a file or folder to the recycling bin



```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\charissa_chua>cd C:\Users\charissa_chua\PycharmProjects\ExerciseTwo\venv\Scripts
C:\Users\charissa_chua\PycharmProjects\ExerciseTwo\venv\Scripts>pip.exe install send2trash
Collecting send2trash
  Using cached https://files.pythonhosted.org/packages/49/46/c3dc27481d1cc57b9385aff41c474ceb7714f79
Installing collected packages: send2trash
Successfully installed send2trash-1.5.0
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\charissa_chua\PycharmProjects\ExerciseTwo\venv\Scripts>
```

Recycle Bin

- new_hello.txt
Text Document
0 bytes

Walk a Directory

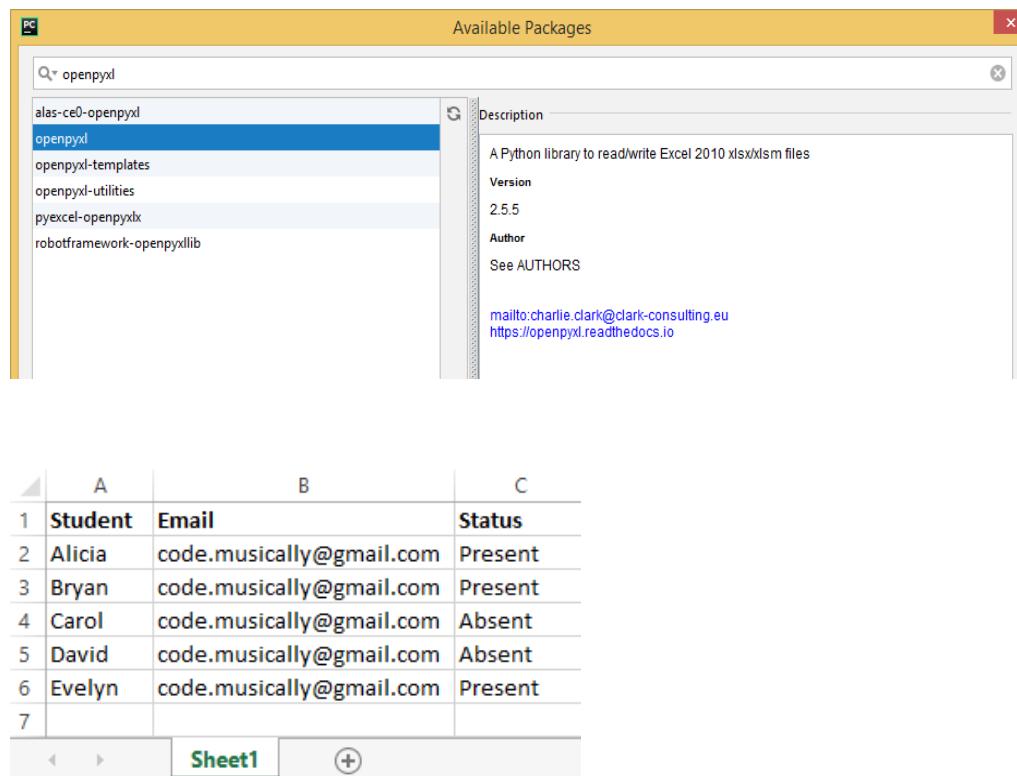
```
FileUtils.py x
1 import os
2 import shutil
3
4 for folderName, subfolders, filenames in os.walk("C:\\\\Users\\\\charissa_chua\\\\Downloads"):
5     print("The folder is " + folderName)
6     print("The subfolders in " + folderName + " are: " + str(subfolders))
7     print("The filenames in " + folderName + " are: " + str(filenames))
8     print()
9
10    for subfolder in subfolders:
11        if "c235" in subfolder:
12            rmfolder = os.path.join(folderName, subfolder)
13            print("rmtree on " + rmfolder)
14            shutil.rmtree(rmfolder)
15
16
17    for file in filenames:
18        if file.endswith(".jpeg"):
19            srcFile = os.path.join(folderName, file)
20            dstFile = os.path.join(folderName, file + ".backup")
21            print(dstFile)
22            shutil.copy(srcFile, dstFile)
```

Remove all folders with “c235” in them

Back up all jpeg files with file extension “.backup”

Working with Excel

- Install openpyxl module using PyCharm or “pip install openpyxl”
- Make sure the file is available - students_attendance.xlsx
- Full openpyxl documentation: <https://openpyxl.readthedocs.io/en/stable/index.html>



The screenshot shows the PyCharm interface with the 'Available Packages' window open. A search query 'openpyxl' has been entered, and the 'openpyxl' package is highlighted. The right pane displays the package's description as a Python library for reading and writing Excel 2010 xlsx/xlsm files, version 2.5.5, and author Charlie Clark. Below the window is a screenshot of an Excel spreadsheet titled 'Sheet1' containing the following data:

	A	B	C
1	Student	Email	Status
2	Alicia	code.musically@gmail.com	Present
3	Bryan	code.musically@gmail.com	Present
4	Carol	code.musically@gmail.com	Absent
5	David	code.musically@gmail.com	Absent
6	Evelyn	code.musically@gmail.com	Present
7			

Working with Excel

- Import openpyxl
- Load Excel content into “workbook” object by specifying the entire path
- Get the active worksheet named "Sheet1"
- Get the number of rows and columns
- Use for loop to go through every row.
- Extract the status at Column C to check for attendance

```
1 import openpyxl
2
3 workbook = openpyxl.load_workbook("students_attendance.xlsx")
4 sheet=workbook["Sheet1"]
5
6 max_row = sheet.max_row
7 max_column = sheet.max_column
8
9 #loop through every row
10 for i in range(1,max_row+1):
11
12     #read cell
13     attendance = sheet.cell(row=i, column=3).value
14
15     #check attendance
16     if attendance == "Absent":
17         name = sheet.cell(row=i,column=1).value
18         email = sheet.cell(row=i,column=2).value
19         print(name + " is absent")
```

Working with Excel - Update

- Import openpyxl
- Load file into memory & get the sheet
- Add value to cell
- Save the spreadsheet

```
1 import openpyxl
2 from openpyxl.comments import Comment
3
4 workbook = openpyxl.load_workbook("students_attendance.xlsx")
5 sheet=workbook["Sheet1"]
6
7 max_row = sheet.max_row
8 max_column = sheet.max_column
9
10 #read cell
11 for i in range(1,max_row+1):
12     attendance = sheet.cell(row=i, column=3).value
13     if attendance == "Absent":
14         name = sheet.cell(row=i,column=1).value
15         email = sheet.cell(row=i,column=2).value
16         print(name + " is absent")
17
18 #add value
19 sheet['A7'].value='Felicia'
20 sheet['B7'].value='Felicia@gmail.com'
21 sheet['C7'].value='Present'
22
23 #add comment
24 sheet['A7'].comment= Comment('Change text automatically','User')
25
26 #add a new element that count the number of non empty cell
27 #sheet['D7'] = '=COUNTA(A2:A50)'
28
29 #save the file
30 workbook.save("students_attendance_comment.xlsx")
```

Working with Excel - Create

- Import openpyxl
- Create new workbook
- Get default sheet or create new sheet
- Insert value into cells
- Save spreadsheet

```
1 import openpyxl
2
3 workbook = openpyxl.Workbook()
4
5 #get the default sheet
6 sheet=workbook["Sheet"]
7
8 #create a list of tuples as data source
9 data = [
10     (225.7, 'Gone with the Wind', 'Victor Fleming'),
11     (194.4, 'Star Wars', 'George Lucas'),
12     (161.0, 'ET: The Extraterrestrial', 'Steven Spielberg')
13 ]
14
15 #update value into cell
16 for row, (admissions,name, director) in enumerate(data,1):
17     sheet['A{}'.format(row)].value = admissions
18     sheet['B{}'.format(row)].value = name
19
20 #create a new sheet
21 sheet = workbook.create_sheet("Directors")
22
23 #print out added sheet name
24 print(workbook.sheetnames)
25
26 #update value into cell
27 for row, (admissions,name, director) in enumerate(data,1):
28     sheet['A{}'.format(row)].value = director
29     sheet['B{}'.format(row)].value = name
30
31 #save the spreadsheet
32 workbook.save("movies1.xlsx")
```

Working with Excel - Format

- Import openpyxl & styles
- Set up colors and styles
- Loop through cell and set properties
- Save spreadsheet

```
1 import openpyxl
2 from openpyxl.styles import Font, PatternFill, Border, Side
3
4 workbook = openpyxl.Workbook()
5
6 # create a list of tuples as data source
7 data = [
8     ('Name', 'Admission'),
9     ('Gone with the Wind', 225.7),
10    ('Star Wars', 161.0),
11    ('ET: The Extraterrestrial', 161.0)
12 ]
13
14 sheet = workbook['Sheet']
15 for row in data:
16     sheet.append(row)
17
18 #define the colors to use for styling
19 BLUE = "#0033CC"
20 LIGHT_BLUE = "#E6ECFF"
21 WHITE = "#FFFFFF"
22
23 #define styling
24 header_font = Font(name="Tahoma", size=14, color=WHITE)
25 header_fill = PatternFill("solid", fgColor=BLUE)
26
27 # format header
28 for row in sheet["A1:B1"]:
29     for cell in row:
30         cell.font = header_font
31         cell.fill = header_fill
32
33 #define styling
34 white_side = Side(border_style="thin", color=WHITE)
35 blue_side = Side(border_style="thin", color=BLUE)
36 alternate_fill = PatternFill("solid", fgColor=LIGHT_BLUE)
37 border = Border(bottom=blue_side, left=white_side, right=white_side)
38
39 # format rows
40 for row_index, row in enumerate(sheet["A2:B5"]):
41     for cell in row:
42         cell.border = border
43         if row_index % 2 :
44             cell.fill = alternate_fill
45
46 workbook.save("movie_format.xlsx")
```

Working with CSV File

- **CSV** stands for Comma-Separated Values (sometimes also called Comma Delimited File).
- It is commonly used for storing data in a table structured format.
- Each line/row in the file is a data record.
- Each field in the row is separated using a comma. The comma serves as a column boundary (aka delimiter) that separates the values into different cells of a table. (see next slide)

What is CSV format?

- The same data when viewed with Excel ...

A	B	C	D	E	F
1 AARON	D	X	X	X	X
2 BERT	A	X	X	X	X
3 BRADLEY	C	A	X	X	B
4 JEFFREY	B	C	C	X	C
5 ELLIOT	B	B	B	X	A
6 CLAY	F	F	X	X	X
7 JESSE	A	A	A	A	A
8 FELIX	C	C	C	X	X
9 ERIN	B	B	B	X	B
10 TORY	B	A	B	X	C
11 HECTOR	B	C	A	X	A
12 ZACK	X	X	X	C	D

← Data is automatically tabulated in Excel into rows and columns (each value is in a cell)

- ... and when viewed in plain text (e.g. in notepad) ...

File	Edit	Format	View	Help
AARON,D,X,X,X,X				
BERT,A,X,X,X,X				
BRADLEY,C,A,X,X,B				
JEFFREY,B,C,C,X,C				
ELLIOT,B,B,B,X,A				
CLAY,F,F,X,X,X				
JESSE,A,A,A,A,A				
FELIX,C,C,C,X,X				
ERIN,B,B,B,X,B				
TORY,B,A,B,X,C				
HECTOR,B,C,A,X,A				
ZACK,X,X,X,C,D				

← This is the RAW FORMAT of the file seen by computer programs:

- Each row is a record
- Values in a row are separated / delimited by comma ','

Reading from a CSV File

```

1) Load CSV library
2) Open the file named 'W65Z.csv' for reading (indicated by 'r'
argument). readerFileHandle linked the CSV file to the program.

1 import csv
2
3 with open("W65Z.csv", "r", newline='') as readerFileHandle:
4     reader1 = csv.reader(readerFileHandle)
5     #using for-loop to retrieve from the CSV file line by line
6     for row in reader1:
7         print (row)
8
9 readerFileHandle.close()

5) Close the file (remove the link)

3) Read the file
content as CSV
format and store it in
reader1 as a list of
values

4) For-loop retrieve each item in reader1 (a list) into
the loop variable row and display it. (Note: Each line
in the file becomes a list of values)

['AARON', 'D', 'X', 'X', 'X', 'X']
['BERT', 'A', 'X', 'X', 'X', 'X']
['BRADLEY', 'C', 'A', 'X', 'X', 'B']
['JEFFREY', 'B', 'C', 'C', 'X', 'C']
['ELLIOT', 'B', 'B', 'B', 'X', 'A']
['CLAY', 'E', 'F', 'X', 'X', 'X']

```

Note: There are other libraries/modules that have functions that read or write data in spreadsheets.

Writing to a CSV File

1) Load CSV library

2) Create (if new) & Open the file named “W65z_new.csv” for writing (indicated by ‘w’ argument). `writerFileHandle` links the file to the program.

```

1 import csv
2 with open("W65z_new.csv", "w", newline='') as writerFileHandle:
3     writer1 = csv.writer(writerFileHandle)
4
5     row1 = ['AARON', 'D', 'X', 'X', 'X', 'X']
6     row2 = ['BERT', 'A', 'X', 'X', 'X', 'X']
7     row3 = ['BRADLEY', 'C', 'A', 'X', 'X', 'B']
8     rowlist = [row1, row2, row3]
9
10    for row in rowlist:
11        writer1.writerow(row)
12
13    writerFileHandle.close()

```

3) `writer1` stores content to be written to the file in CSV format

4) `rowlist` stores the content to be written to the CSV file.
(`rowlist` is a list containing lists as items)

5) For-loop retrieves each item from `rowlist` into loop variable
`row` → `row` (a list) is written as 1 csv formatted line into the file.

6) Close the file (Remove the link)

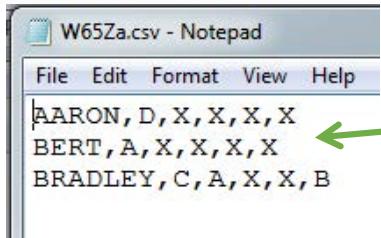


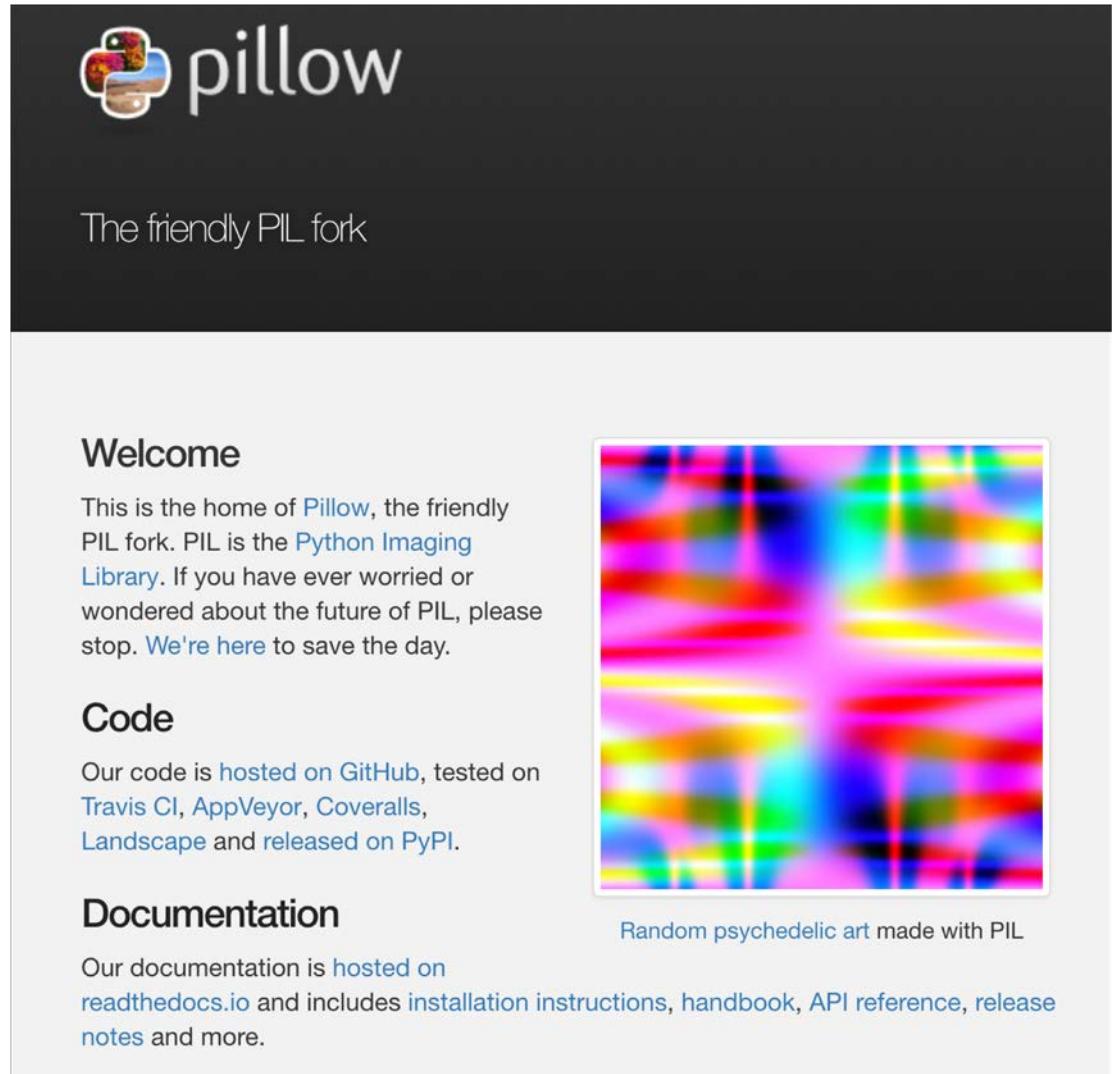
Image processing

For the next chapter we are going to use the new Python Image Library, or in short Pillow.

If it is not installed, you can install this package directly in PyCharm.

Alternatively, run the command:
`pip install Pillow`

The documentation url is at:
<http://pillow.readthedocs.io/en/5.1.x/handbook/index.html>



The screenshot shows the official website for Pillow. At the top, there's a dark header with the Pillow logo (a stylized icon made of colored pixels) and the word "pillow" in white lowercase letters. Below the header, the text "The friendly PIL fork" is displayed. The main content area has a light gray background. It features several sections: "Welcome" (with a paragraph about the project), "Code" (mentioning GitHub, Travis CI, AppVeyor, Coveralls, Landscape, and PyPI), "Documentation" (linking to readthedocs.io), and a sidebar with a colorful, abstract image titled "Random psychedelic art made with PIL".

Welcome

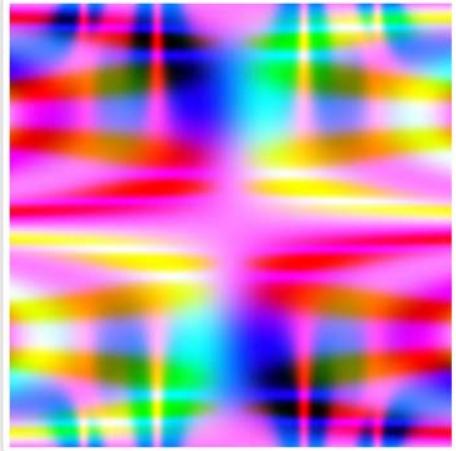
This is the home of [Pillow](#), the friendly PIL fork. PIL is the [Python Imaging Library](#). If you have ever worried or wondered about the future of PIL, please stop. [We're here](#) to save the day.

Code

Our code is [hosted on GitHub](#), tested on [Travis CI](#), [AppVeyor](#), [Coveralls](#), [Landscape](#) and released on [PyPI](#).

Documentation

Our documentation is [hosted on readthedocs.io](#) and includes [installation instructions](#), [handbook](#), [API reference](#), [release notes](#) and more.



Random psychedelic art made with PIL

Image processing

Before we start processing images, we need to be able to loop through all the images.

Luckily we just learned how to do that, so let's start by copying last exercise in a new file.

You can retain the function `searchByExtension`, because that is the most useful one in this case.

```
1 import os
2
3 where = ".\\Day2Resource\\img\\"
4
5 def searchByExtension(ext):
6     c = 1
7     for root, dirs, files in os.walk(where):
8         for file in files:
9             fullname = os.path.join(root, file)
10            if file.endswith(ext):
11                print("%2d %s" % (c, fullname))
12                c += 1
13
14 searchByExtension("jpg")
15
```

Also we would like to keep track of the number of images, so we add a variable `c` (for count), set it to 1 and increase it by one every time.

In the end you should have something like the image above.



Image processing

When you run it, it should list all the .jpg in all folders.

However, not all images are in the list. There is one SVG image, one PNG image and 1 JPG image with extension in UPPER CASE. These don't match `.endswith(".jpg")`.

Lets fix these and name the function `processAllImages()`

You can convert fileName to lower with
`fileName.lower().endswith(...)`

Name _____

-  image1.jpg
 -  image2.jpg
 -  image3.JPG
 -  image4.svg
 -  image5.png

```
C:\Users\denise_quek\AppData\Local\P  
1 .\Day2Resource\img\image1.jpg  
2 .\Day2Resource\img\image2.jpg
```

Process finished with exit code 0

Image processing

This will print all the images, regardless of upper case.

Of course there are more types of images than JPEG, PNG and SVG.

You should now see a list of 5 images.

But if we want to experiment with the Image library, we don't want to apply it on all images, so let's add another parameter to the function called onlyFirst and abort the function after the first.

```
1 import os
2
3 where = ".\\Day2Resource\\img\\"
4
5 def processAllImages():
6     c = 1
7     for root, dirs, files in os.walk(where):
8         for file in files:
9             fullname = os.path.join(root, file)
10            if file.lower().endswith("jpg") or \
11                file.lower().endswith("bmp") or \
12                file.lower().endswith("png") or \
13                file.lower().endswith("svg"):
14                print("%2d %s" % (c, fullname))
15                c += 1
16
17 processAllImages()
```

Image processing

Now our function will only process one if the parameter onlyFirst is set to True.

Not really a good name for this function, but let's go ahead with this.

Now we are ready to explore the Image library

```
1 import os
2
3 where = ".\\Day2Resource\\img\\"
4
5 def processAllImages(onlyFirst):
6     c = 1
7     for root, dirs, files in os.walk(where):
8         for file in files:
9             fullname = os.path.join(root, file)
10            if file.lower().endswith("jpg") or \
11                file.lower().endswith("bmp") or \
12                file.lower().endswith("png") or \
13                file.lower().endswith("svg"):
14                print("%2d %s" % (c, fullname))
15                c += 1
16                if (onlyFirst):
17                    return
18
19 processAllImages(True)
```

```
C:\Users\denise_quek\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/denise_quek/Desktop/Day2Resource/day2resource.py
1 .\Day2Resource\img\image1.jpg

Process finished with exit code 0
```

Image processing

Let's explore what Pillow can do.

As a start we need to import it:

import Image

We can open images with
im = Image.open(fullname)

Then we can get the size of the
image using im.size

```
1  import os
2  from PIL import Image
3
4  where = ".\\Day2Resource\\img\\"
5
6  def processAllImages(onlyFirst):
7      c = 1
8      for root, dirs, files in os.walk(where):
9          for file in files:
10             fullname = os.path.join(root, file)
11             if file.lower().endswith("jpg") or \
12                 file.lower().endswith("bmp") or \
13                 file.lower().endswith("png") or \
14                 file.lower().endswith("svg"):
15                 im = Image.open(fullname)
16                 print("%2d %s %s" % (c, fullname, im.size))
17                 im.show()
18                 c += 1
19             if (onlyFirst):
20                 return
21
22 processAllImages(True)
```

Image processing

Let's print more info :
im.size, im.mode etc.

You can see the image with
im.show()

Note:
If your code does not fit on one
line, you can use \ (backslash)
and continue on the next.

```
1 import os
2 from PIL import Image
3
4 where = ".\\Day2Resource\\img\\"
5
6 def processAllImages(onlyFirst):
7     c = 1
8     for root, dirs, files in os.walk(where):
9         for file in files:
10             fullname = os.path.join(root, file)
11             if file.lower().endswith("jpg") or \
12                 file.lower().endswith("bmp") or \
13                 file.lower().endswith("png") or \
14                 file.lower().endswith("svg"):
15                 im = Image.open(fullname)
16                 print("%2d %s %s (%s)" % (c, fullname, im.size, im.mode))
17                 im.show()
18                 c += 1
19             if (onlyFirst):
20                 return
21
22 processAllImages(True)
```

```
C:\Users\denise_quek\AppData\Local\Programs\Python\Python
1 .\Day2Resource\img\image1.jpg (1599, 1066) (RGB)
```

```
Process finished with exit code 0
```

Image processing - filtering

Now that we can load and understand the image, it is time to try and modify it.

Pillow has many conversion and filters, we will use some of them.
 But if you need more, go ahead :
<http://pillow.readthedocs.io/en/5.1.x/handbook/index.html>

To use filters we need to extend our import:

from PIL import Image, ImageFilter,
 ImageOps

The way you can apply filters is :
 out = im.filter(ImageFilter.BLUR)

```

1  import os
2  from PIL import Image, ImageFilter
3
4  where = ".\\Day2Resource\\img\\"
5
6  def processAllImages(onlyFirst):
7      c = 1
8      for root, dirs, files in os.walk(where):
9          for file in files:
10             fullname = os.path.join(root, file)
11             if (file.lower().endswith("jpg") or \
12                 file.lower().endswith("bmp") or \
13                 file.lower().endswith("png") or \
14                 file.lower().endswith("svg")):
15                 im = Image.open(fullname)
16                 print("%2d %s size:%s (%s) " \
17                     % (c, fullname, im.size, im.mode))
18
19                 out = im.filter(ImageFilter.BLUR)
20                 im.show()
21                 out.show()
22                 c += 1
23             if (onlyFirst):
24                 return
25
26 processAllImages(True)

```

Image processing - filtering



```
image = image.filter(ImageFilter.FIND_EDGES)
```



```
image = ImageOps.solarize(image)
```

```
image = ImageOps.grayscale(image)
```



```
image = image.filter(ImageFilter.CONTOUR)
```



Image processing - rotating

We can do a lot with images.
Let's look at rotation and flipping

Try to rotate and flip your images.

Another cool effect is to make it brighter by changing the contrast

Flipping the image horizontally or vertically
out = im.transpose(Image.FLIP_LEFT_RIGHT)
out = im.transpose(Image.FLIP_TOP_BOTTOM)

Rotating the image

```
out = im.transpose(Image.ROTATE_90)
out = im.transpose(Image.ROTATE_180)
out = im.transpose(Image.ROTATE_270)
```

Contrast

First add ImageEnhance to our imports:
from PIL import Image, ImageFilter, ImageEnhance
Then:
enh = ImageEnhance.Contrast(im)
out = enh.enhance(1.3)

Image processing - writing

You can see the image, but it's not being saved !

Let's agree we store the output images in \Day2Resource\img\out and store this string in a variable "outFolder" at line 5.

All you need to do to save the images in the "out" folder is:
`out.save(the name of the output file)`

You know how we create the fullname, can you copy that line and use it to create outFilename?
Then we use `out.save(outFilename)` after line 18.

```
1 import os
2 from PIL import Image, ImageFilter
3
4 where = ".\\Day2Resource\\img\\"
5
6 def processAllImages(onlyFirst):
7     c = 1
8     for root, dirs, files in os.walk(where):
9         for file in files:
10             fullname = os.path.join(root, file)
11             if (file.lower().endswith("jpg") or \
12                 file.lower().endswith("bmp") or \
13                 file.lower().endswith("png") or \
14                 file.lower().endswith("svg")):
15                 im = Image.open(fullname)
16                 print("%d %s size:%s (%s) " \
17                     % (c, fullname, im.size, im.mode))
18
19                 out = im.filter(ImageFilter.BLUR)
20                 im.show()
21                 out.show()
22                 c += 1
23             if (onlyFirst):
24                 return
25
26 processAllImages(True)
```

Image processing - writing

I hope you can see your converted image in the out folder now.

If you are ready, you can set the safety off (False) and convert all the images

We are going to convert our images again, so it would be good if we can clean up the out folder before we run our processAllImages.

Let's create a small function **cleanOutput** to delete all files in the output folder. You can use **os.remove(fullName)** to delete.

```

1 import os
2 from PIL import Image, ImageFilter
3
4 where = ".\\Day2Resource\\img\\"
5 outFolder = ".\\Day2Resource\\img\\out\\"
6
7 def processAllImages(onlyFirst):
8     c = 1
9     for root, dirs, files in os.walk(where):
10         for file in files:
11             fullname = os.path.join(root, file)
12             if (file.lower().endswith("jpg") or \
13                 file.lower().endswith("bmp") or \
14                 file.lower().endswith("png") or \
15                 file.lower().endswith("svg")):
16                 im = Image.open(fullname)
17                 outFilename = os.path.join(outFolder, file)
18                 print("%2d %s size:%s (%s)" \
19                     % (c, fullname, im.size, im.mode))
20
21                 out = im.filter(ImageFilter.BLUR)
22                 im.show()
23                 out.show()
24                 out.save(outFilename)
25                 c += 1
26             if (onlyFirst):
27                 return
28
29 processAllImages(True)

```

Image processing - writing

Be careful, you don't want to delete your holiday photos !

You could have used the same code to walk through the files but use the outFolder instead !

Then for each file, you call the os.remove(fullName)

Calling it right before our processAllImages should make sure we have a clean output folder.

```
7 def cleanOutput():
8     print("Removing old out files")
9     for root, dirs, files in os.walk(outFolder):
10         for file in files:
11             fullName = os.path.join(root, file)
12             os.remove(fullName)
13             print(".")
14     print("done")
```

```
cleanOutput()
processAllImages(True)
```

Image processing - watermark

Create the mark image
You can reduce the size to 100,100

Create a new function called
`def watermark(im, mark, position):`
....

It takes the original image, the watermark image and the desired position
that we want the watermark to appear.
The function will return the result.

We can use this function
`watermark(im, mark, (0, 50)).show()`
or
`imOut = watermark(im, mark, (0,50))
imOut.save(fileOut)`

`mark = Image.open(".\\Day2Resources\\watermark.png")
mark = mark.resize((100,100))`

Maybe you want to leave a
small footprint on your
images, called watermark.

In this case we can use the
\\Day2Resource\\watermark.p
ng
and place it in each image on
the bottom right.



Image processing - watermark

First we need to create a new layer with the size of the original image.

Then we paste the watermark image at the desired position and we return the composite.

Finally we merge the image and the layer together and return the result.

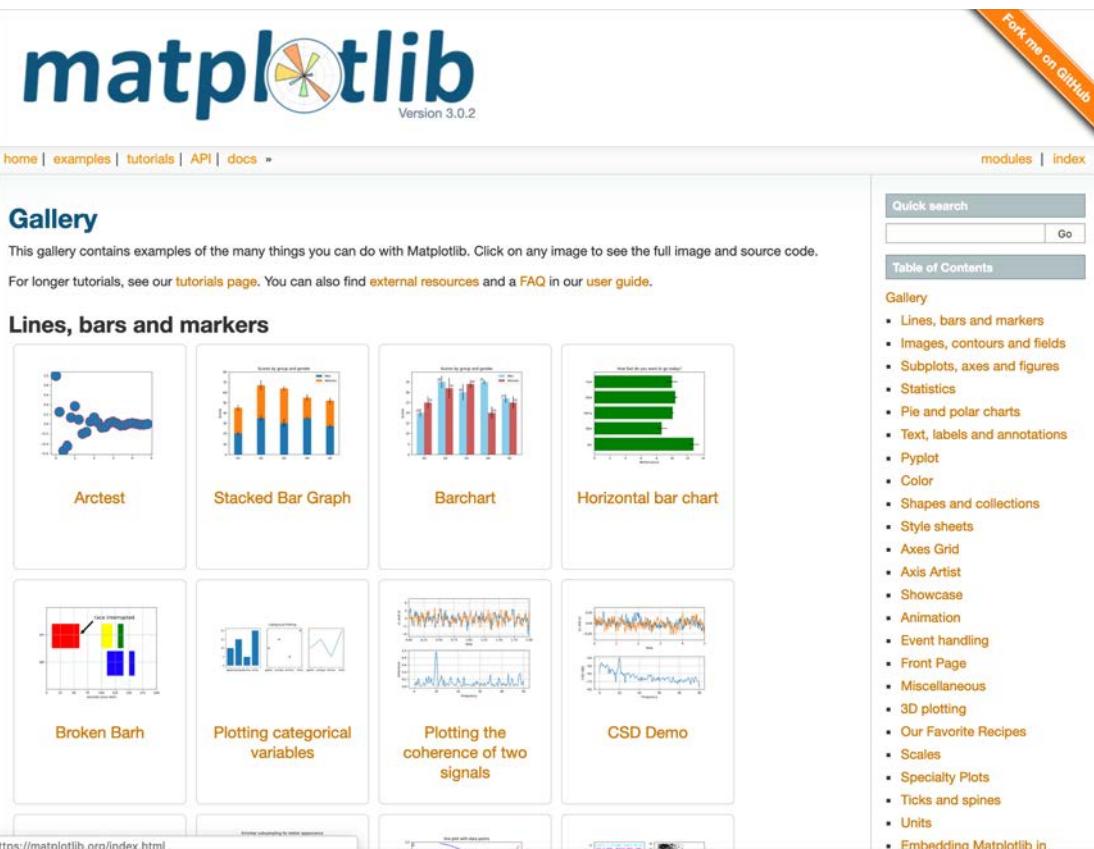
Then you can use it like this.

```
1  from PIL import Image
2
3  def watermark(im, mark, position):
4      layer = Image.new("RGBA", im.size, (0,0,0,0))
5      layer.paste(mark, position)
6      return Image.composite(layer, im, layer)
7
8
9  im = Image.open("Day2Resource\\img\\clungup.jpg")
10 mark = Image.open("Day2Resource\\watermark.png")
11 mark = mark.resize((100,100))
12
13 out = watermark(im, mark, (0,50))
14 out.show()
```

Data Visualization – charting with matplotlib

- Install matplotlib
- Full documentation:
<https://matplotlib.org/>

<https://matplotlib.org/gallery/index.html>



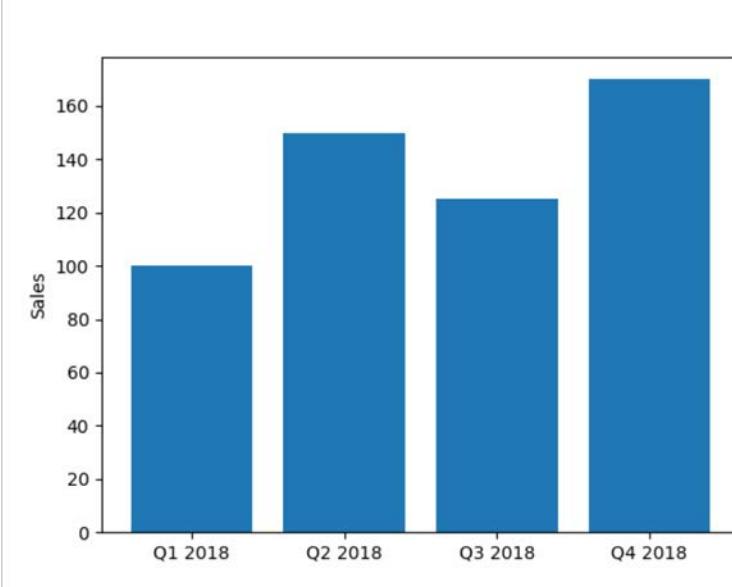
The screenshot shows the Matplotlib Gallery page. At the top, there's a large "matplotlib" logo with "Version 3.0.2" below it. To the right is an orange "Fork me on GitHub" button. Below the logo, there's a navigation bar with links for "home", "examples", "tutorials", "API", "docs", "modules", and "index". On the left, there's a "Quick search" input field and a "Table of Contents" section. The main content area is titled "Gallery" and contains a grid of 12 chart examples. Each example has a small thumbnail and a title: "Arctest", "Stacked Bar Graph", "Barchart", "Horizontal bar chart", "Broken Barth", "Plotting categorical variables", "Plotting the coherence of two signals", and "CSD Demo". A footer at the bottom of the page includes a link to "https://matplotlib.org/index.html".

<https://matplotlib.org/gallery/index.html>

Data Visualization – charting with matplotlib

- Install matplotlib
- Prepare data
- Create bar graph
- Display the chart

```
1 import matplotlib.pyplot as plt
2
3 #set up values
4 VALUES = [100,150,125,170]
5 POS = [0,1,2,3]
6 LABELS = ['Q1 2018','Q2 2018','Q3 2018','Q4 2018']
7
8 #set up the chart
9 plt.bar(POS,VALUES)
10 plt.xticks(POS, LABELS)
11 plt.ylabel('Sales')
12
13 #to display the chart
14 plt.show()
```



https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html

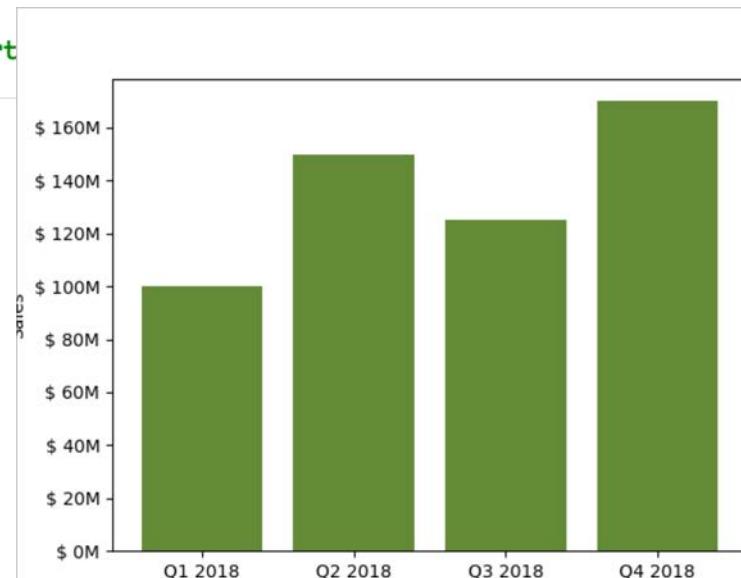
Data Visualization – charting with matplotlib

- Install matplotlib
- Prepare data
- Customise graph options
- Create bar graph
- Display the chart

```

1 import matplotlib.pyplot as plt
2 from matplotlib.ticker import FuncFormatter
3
4 def value_format(value, position):
5     return '$ {:.0f}M'.format(int(value))
6
7 # set up values
8 VALUES = [100, 150, 125, 170]
9 POS = [0, 1, 2, 3]
10 LABELS = ['Q1 2018', 'Q2 2018', 'Q3 2018', 'Q4 2018']
11
12 # set up the chart
13 # Colors can be specified in multiple formats, as
14 # described in https://matplotlib.org/api/colors_api.html
15 # https://xkcd.com/color/rgb/
16 plt.bar(POS,VALUES, color='xkcd:moss green')
17 plt.xticks(POS, LABELS)
18 plt.ylabel('Sales')
19
20 # retrieve the current axes and apply formatter |
21 axes = plt.gca()
22 axes.yaxis.set_major_formatter(FuncFormatter(value_format))
23
24 # to display the chart
25 plt.show()

```



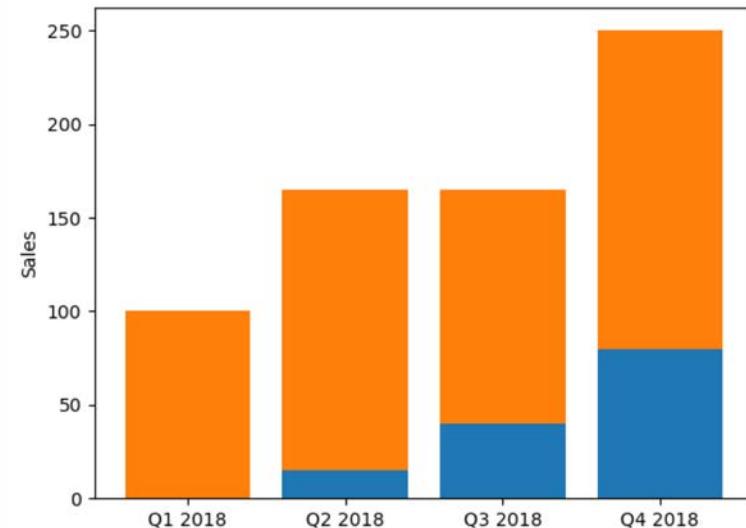
Data Visualization – charting with matplotlib

- Stacked bars

```

1 import matplotlib.pyplot as plt
2
3 #set up values
4 VALUESA = [100,150,125,170]
5 VALUESB = [0,15,40,80]
6 POS = [0,1,2,3]
7 LABELS = ['Q1 2018','Q2 2018','Q3 2018','Q4 2018']
8
9 #set up the first bar
10 plt.bar(POS,VALUESB)
11 #set up the stacked bars
12 plt.bar(POS,VALUESA, bottom=VALUESB)
13
14 plt.ylabel('Sales')
15 plt.xticks(POS, LABELS)
16
17 #to display the chart
18 plt.show()
19

```



Data Visualization – charting with matplotlib

- Stacked bars

```
1 import matplotlib.pyplot as plt
2
3 #set up values
4 VALUESA = [100,150,125,170]
5 VALUESB = [0,15,40,80]
6 VALUESC = [3,5,7,3]
7 POS = [0,1,2,3]
8 LABELS = ['Q1 2018','Q2 2018','Q3 2018','Q4 2018']
9
10 # Create the first plot
11 plt.subplot(2,1,1)
12
13 #creata a bar graph with informaton about VALUESA
14 plt.bar(POS,VALUESA)
15 plt.ylabel('Sales')
16
17 #create a different Y axis, and add information
18 #about VALUESB as a line plot
19 plt.twinx()
20 plt.plot(POS,VALUESB,'o-',color='red')
21 plt.xticks(POS, LABELS)
22 plt.ylabel('Sales B')
23 plt.xticks(POS, LABELS)
24
25 #create another subplot and fill it iwth VALUESC
26 plt.subplot(2,1,2)
27 plt.plot(POS, VALUESC)
28 plt.gca().set_ylim(bottom=0)
29 plt.xticks(POS,LABELS)
30
31 plt.show()
```



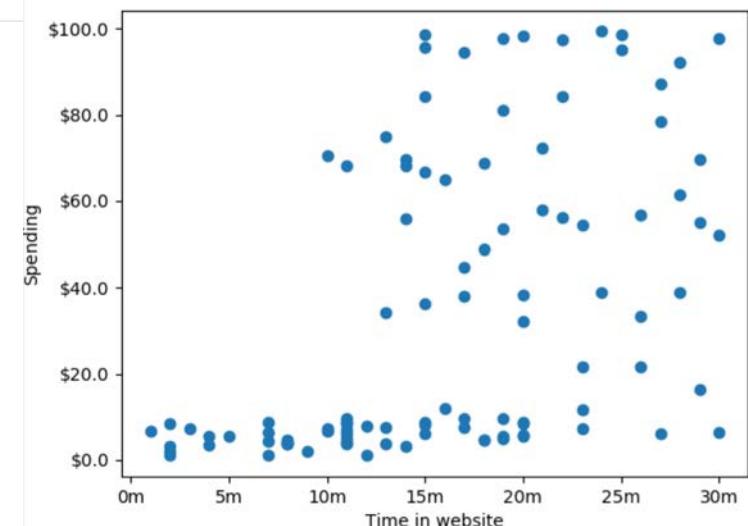
Data Visualization – scatter plot

- To save a plot:
`plt.savefig(filename)`
- Save the plot before you display.

```

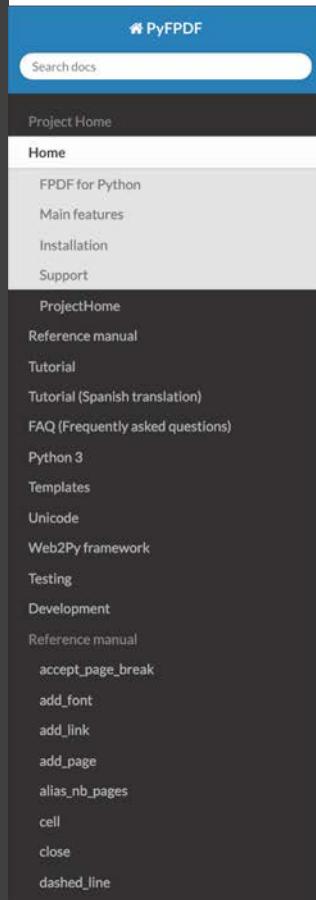
1 import csv
2 import matplotlib.pyplot as plt
3 from matplotlib.ticker import FuncFormatter
4
5 def format_minutes(value, pos):
6     return '{}m'.format(int(value))
7
8 def format_dollars(value, pos):
9     return '${}'.format(value)
10
11 # read data from csv
12 with open('scatter.csv') as fp:
13     reader = csv.reader(fp)
14     data = list(reader)
15
16 data_x = [float(x) for x, y, in data]
17 data_y = [float(y) for x, y in data]
18 plt.scatter(data_x, data_y)
19
20 plt.gca().xaxis.set_major_formatter(FuncFormatter(format_minutes))
21 plt.xlabel('Time in website')
22 plt.gca().yaxis.set_major_formatter(FuncFormatter(format_dollars))
23 plt.ylabel('Spending')
24
25 plt.show()

```



PDF

- Install fpdf
 - pip install fpdf



The screenshot shows the left sidebar of the PyFPDF documentation. The sidebar has a dark background and a light blue header bar with the text "PyFPDF". Below the header, there is a search bar labeled "Search docs". The sidebar contains a list of links:

- Project Home
- Home
 - FPDF for Python
 - Main features
 - Installation
 - Support
 - ProjectHome
 - Reference manual
 - Tutorial
 - Tutorial (Spanish translation)
 - FAQ (Frequently asked questions)
 - Python 3
 - Templates
 - Unicode
 - Web2Py framework
 - Testing
 - Development
 - Reference manual
 - accept_page_break
 - add_font
 - add_link
 - add_page
 - alias_nb_pages
 - cell
 - close
 - dashed_line

Docs » Project Home » Home

[Edit on GitHub](#)

FPDF for Python

PyFPDF is a library for PDF document generation under Python, ported from PHP (see [FPDF](#): "Free"-PDF, a well-known PDFlib-extension replacement with many examples, scripts and derivatives).

Latest Released Version: 1.7 (August 15th, 2012) - Current Development
Version: 1.7.1

Fork me on GitHub

Main features

- Easy to use (and easy to extend)
- Many simple examples and scripts available in many languages
- No external dependencies or extensions (optionally PIL for GIF support)
- No installation, no compilation or other libraries (DLLs) required
- Small and compact code, useful for testing new features and teaching

This repository is a fork of the library's original port by [Max Pat](#), with the following enhancements:

- Python 2.5 to 3.4+ support (see [Python3 support](#))
- [Unicode](#) (UTF-8) TrueType font subset embedding (Central European, Cyrillic, Greek, Baltic, Thai, Chinese, Japanese, Korean, Hindi and almost any other language in the world) New! based on [sFPDF](#) LGPL3 PHP version from [Ian Back](#)
- Improved installers (setup.py, py2exe, PyPI) support
- Barcode I2of5 and code39, QR code coming soon ...
- PNG, GIF and JPG support (including transparency and alpha channel) New!
- Exceptions support, other minor fixes, improvements and PEP8 code cleanups
- Port of the [Tutorial](#) and [ReferenceManual](#) (Spanish translation available)

FPDF original features:



PDF

- Import fpdf
 - Create a new pdf document
 - Add page
 - Add text, logo
 - Save file

```
1 import fpdf
2
3 #create a new pdf
4 document = fpdf.FPDF()
5
6 #define font and color for title and add the first page
7 document.set_font("Times", "B", 14)
8 document.set_text_color(19,83,173)
9 document.add_page()
10
11 #write the title of the document
12 document.cell(0,5,"PDF Test Document")
13 document.ln()
14
15 #write a long paragraph
16 document.set_font("Times", "", 11)
17 document.set_text_color(0)
18 document.multi_cell(0,5, "This is an example of a long paragraph. " * 10)
19 document.ln()
20
21 #write another long paragraph
22 document.multi_cell(0,5, "Another long paragraph. \
23 Lorem ipsum dolor sit amet, consectetur adipiscing elit." * 40)
24
25 #save the document
26 document.output("pdf_report.pdf")
```

PDF – adding images

- Import fpdf
 - Create a new pdf document
 - Add page
 - Add text, logo
 - Save file

```
1 import fpdf
2
3 #create a new pdf
4 document = fpdf.FPDF()
5
6 #define font and color for title and add the first page
7 document.set_font("Times", "B", 14)
8 document.set_text_color(19,83,173)
9 document.add_page()
10
11 #add a image
12 document.image("rp_logo.png", x=10, y=8, w=23)
13 document.set_y(30);
14
15 #write the title of the document
16 document.cell(0,5,"PDF Test Document")
17 document.ln()
18
19 #write a long paragraph
20 document.set_font("Times", "", 11)
21 document.set_text_color(0)
22 document.multi_cell(0,5, "This is an example of a long paragraph. " * 10)
23 document.ln()
24
25 #write another long paragraph
26 document.multi_cell(0,5, "Another long paragraph. \
27 Lorem ipsum dolor sit amet, consectetur adipiscing elit." * 40)
28
29 #add another image
30 document.image("rp_logo.png", w=23)
31
32 #save the document
33 document.output("pdf_report.pdf")
```



PDF Test Document



PDF – adding password

- pip install PyPDF2

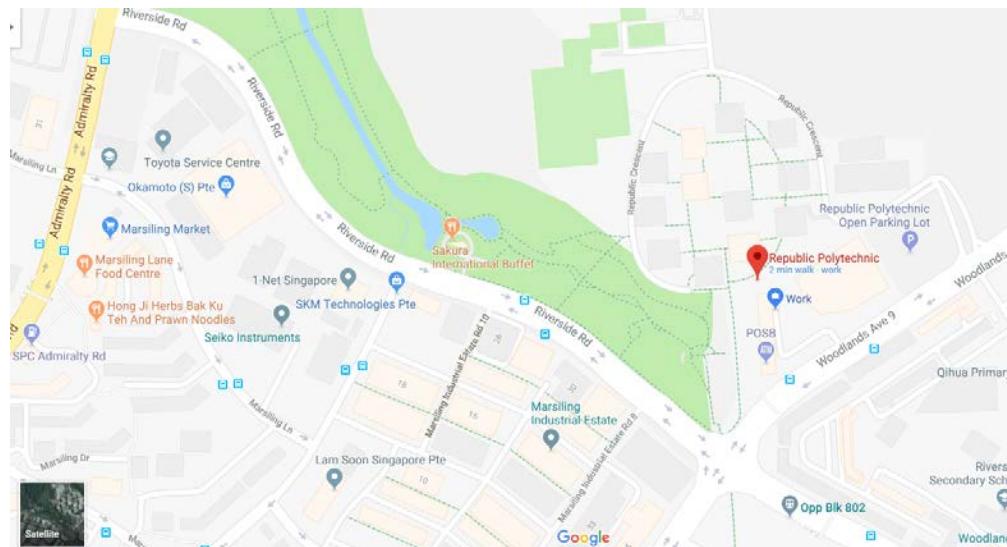
```
1 import fpdf
2 import PyPDF2
3
4 #create a new pdf
5 document = fpdf.FPDF()
6
7 #define font and color for title and add the first page
8 document.set_font("Times", "B", 14)
9 document.set_text_color(19,83,173)
10 document.add_page()
11
12 #add a image
13 document.image("rp_logo.png", x=10, y=8, w=23)
14 document.set_y(30);
15
16 #write the title of the document
17 document.cell(0,5,"PDF Test Document")
18 document.ln()
19
20 #write a long paragraph
21 document.set_font("Times", "", 11)
22 document.set_text_color(0)
23 document.multi_cell(0,5, "This is an example of a long paragraph. " * 10)
24 document.ln()
25
26 #save the document
27 document.output("pdf_report_before_pw.pdf")
28
29 #save the document into a new password protected/encrypted pdf
30 pdffile = open(r"pdf_report_before_pw.pdf", "rb")
31 pdfReader = PyPDF2.PdfFileReader(pdffile)
32 pdfWriter = PyPDF2.PdfFileWriter()
33 for pageNum in range(pdfReader.numPages):
34     pdfWriter.addPage(pdfReader.getPage(pageNum))
35
36 pdfWriter.encrypt('123')
37 resultPDF = open(r"pdf_report_after_pw.pdf", "wb")
38 pdfWriter.write(resultPDF)
39 resultPDF.close()
40 pdffile.close()
```

Connecting to the Web

- `webbrowser` – opens a browser to a specific page
- `requests` – download files and web pages from the Web
- Beautiful Soup – a third party module that parses HTML

`webbrowser.open()` opens a browser to a specific page

```
1  #! python3
2
3  import webbrowser
4
5  url = "https://www.google.com.sg/maps/"
6  webbrowser.open(url)
7
```



Connecting to the Web

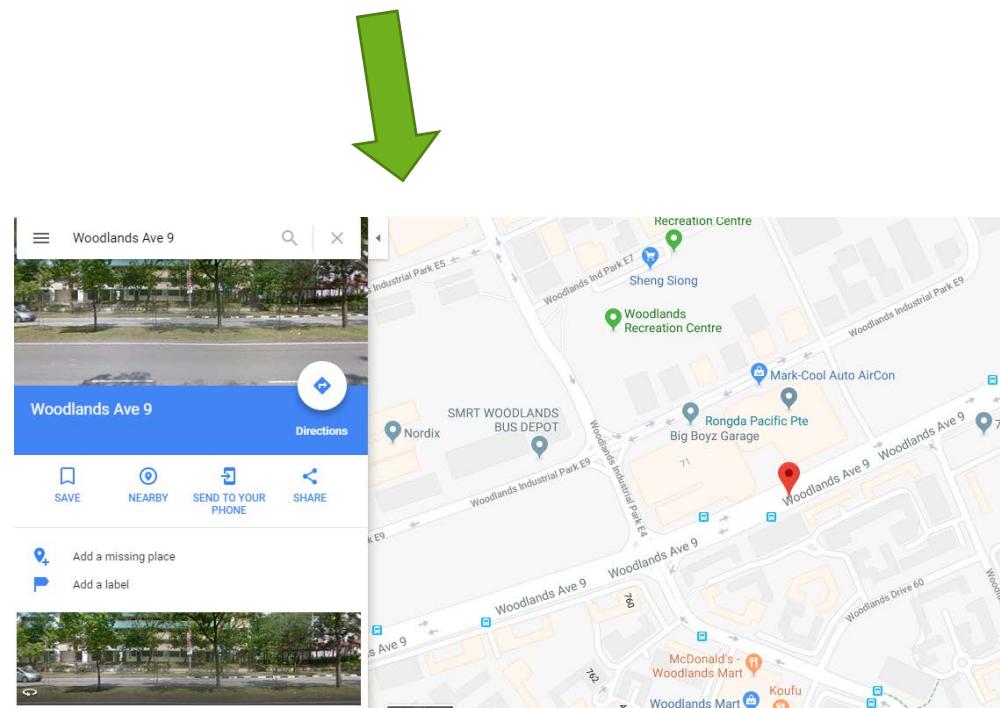
```

1 #! python3
2
3 import webbrowser
4
5 url = "https://www.google.com.sg/maps/"
6 address = input('Enter Address: ')
7 webbrowser.open(url + "place/" + address)

```

`webbrowser.open()` opens a browser to a specific page

C:\Users\denise_quek\AppData\Local\Temp
Enter Address: Woodlands Ave 9



Connecting to the Web

- requests – download files and web pages from the Web
- **Install requests module**

```
1 import requests
2
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
4 req = requests.get(url)
5 print(req.text)
```

Get the required information from
the given URL



Economy Education Environment Finance Health Infrastructure Society Technology Transport

Connecting to the Web

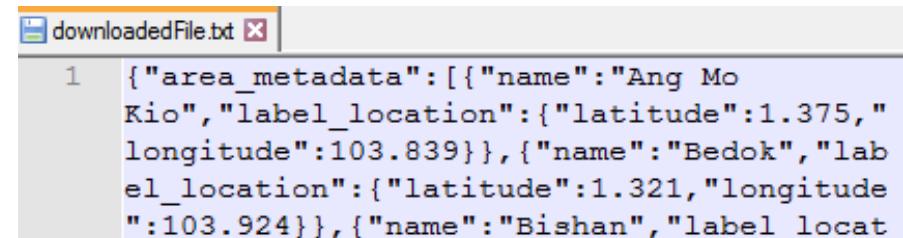
- Use `requests.get()` to get web content from specified URL
- Use `raise_for_status()` to ensure that download is successful before we continue
- Call `open()` with "wb" to create a new file in write binary mode
- Loop over the Response object using `iter_content()`
- Call `write()` on each iteration to write the content to the file
- Remember to close the file

```
1 import requests
2
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
4 req = requests.get(url)
5
6 try:
7     req.raise_for_status()
8
9     playFile = open("downloadedFile.txt", 'wb')
10    for chunk in req.iter_content(100000):
11        print(chunk)
12        playFile.write(chunk)
13    playFile.close()
14
15 except Exception as e:
16     print("There was a problem: %s" % (e))
17
```

Connecting to the Web

- File will be saved in "downloadedFile.txt" (in the same folder as your program)

```
1 import requests
2
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
4 req = requests.get(url)
5
6 try:
7     req.raise_for_status()
8
9     playFile = open("downloadedFile.txt", 'wb')
10    for chunk in req.iter_content(100000):
11        print(chunk)
12        playFile.write(chunk)
13    playFile.close()
14
15 except Exception as e:
16     print("There was a problem: %s" % (e))
17
```



The screenshot shows a file viewer window titled "downloadedFile.txt". The file contains JSON data with one object:

```
{"area_metadata": [{"name": "Ang Mo Kio", "label_location": {"latitude": 1.375, "longitude": 103.839}}, {"name": "Bedok", "label_location": {"latitude": 1.321, "longitude": 103.924}}, {"name": "Bishan", "label_locat
```

Connecting to the Web

- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

```
{"area_metadata": [{"name": "Ang Mo Kio",  
"label_location": {"latitude": 1.375, "longitude":  
103.839}}, {"name": "Bedok", "label_location": {  
"latitude": 1.321, "longitude": 103.924}}, {"name":  
"Bishan", "label_location": {"latitude": 1.350772,  
"longitude": 103.839}}, {"name": "Boon Lay",  
"label_location": {"latitude": 1.304, "longitude":  
103.701}}, {"name": "Bukit Batok",  
"label_location": {"latitude": 1.350772, "longitude":  
103.839}}]}
```

```
1 import requests  
2  
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"  
4 req = requests.get(url)  
5 print(req.text)
```



```
{  
  "area_metadata": [  
    {  
      "name": "Ang Mo Kio",  
      "label_location": {  
        "latitude": 1.375,  
        "longitude": 103.839  
      }  
    },  
    {  
      "name": "Bedok",  
      "label_location": {  
        "latitude": 1.321,  
        "longitude": 103.924  
      }  
    },  
    {  
      "name": "Bishan",  
      "label_location": {  
        "latitude": 1.350772,  
        "longitude": 103.839  
      }  
    },  
    {  
      "name": "Boon Lay",  
      "label_location": {  
        "latitude": 1.304,  
        "longitude": 103.701  
      }  
    },  
    {  
      "name": "Bukit Batok",  
      "label_location": {  
        "latitude": 1.350772,  
        "longitude": 103.839  
      }  
    }  
  ]  
}
```

Connecting to the Web

- To work with JSON data, import json first
- Use json.loads() to load the data in JSON format
- Extract and retrieve the required data

```
1  import json
2  import requests
3
4  url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
5  req = requests.get(url)
6
7  data = json.loads(req.text)
8
9  forecasts = data["items"][0]["forecasts"]
10
11 for forecast in forecasts:
12     area = forecast["area"]
13     weather = forecast["forecast"]
14     print(area + ": " + weather)
```

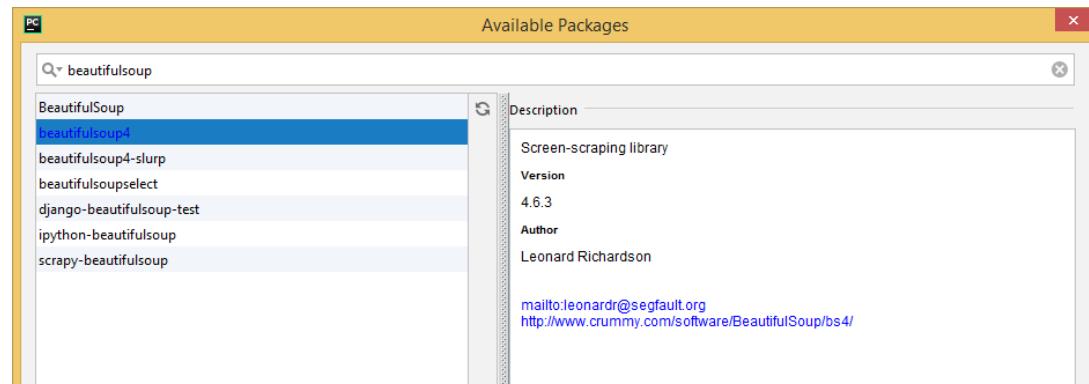


```
C:\Users\denise_quek\AppData\Local\Programs\Python\Py
Ang Mo Kio: Thundery Showers
Bedok: Thundery Showers
Bishan: Heavy Thundery Showers with Gusty Winds
Boon Lay: Heavy Thundery Showers with Gusty Winds
Bukit Batok: Heavy Thundery Showers with Gusty Winds
Bukit Merah: Heavy Thundery Showers with Gusty Winds
```

Connecting to the Web

- Beautiful Soup – a third party module that parses HTML (web pages)
- Install Beautiful Soup 4

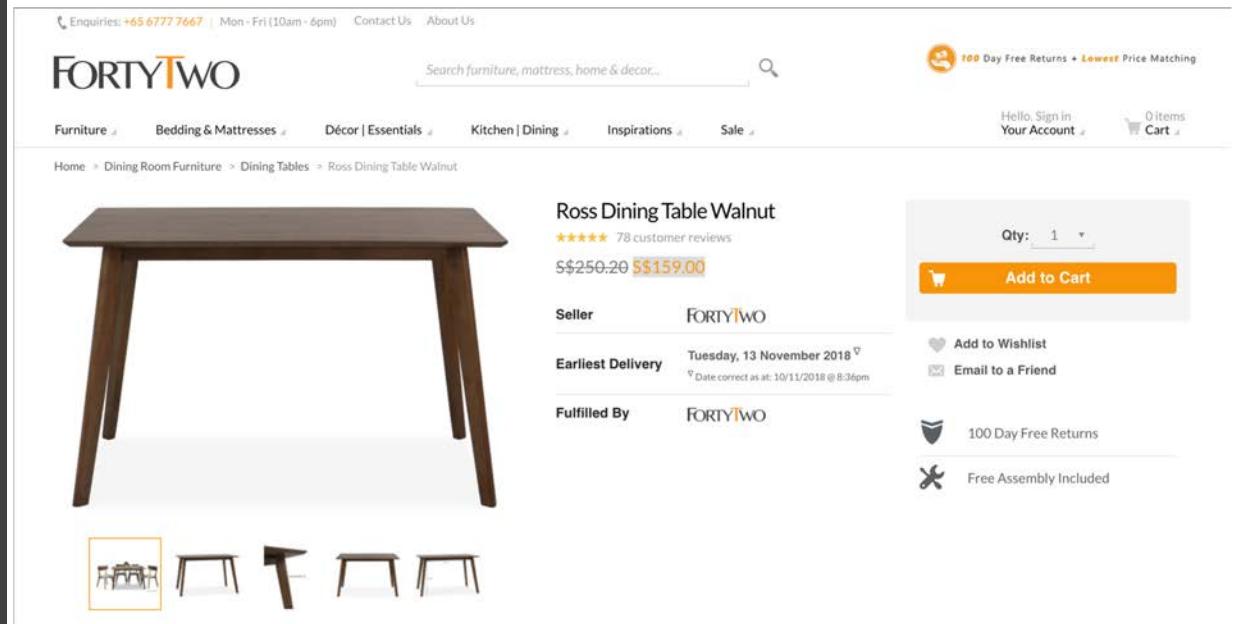
Web Scraping – download and process Web content



Connecting to the Web

- Get the url

<https://www.fortytwo.sg/dining/dining-tables/ross-dining-table-walnut.html>



The screenshot shows a product page for the Ross Dining Table Walnut on the FORTYTWO website. The page includes a large image of the dark wood dining table, a price of \$250.20 reduced to \$159.00, and delivery information. It also features social sharing icons and links for wishlist and friend email.

FORTYTWO

Enquiries: +65 6777 7667 | Mon - Fri (10am - 6pm) Contact Us About Us

Search furniture, mattress, home & decor...

100 Day Free Returns • Lowest Price Matching

Hello, Sign in Your Account 0 Items Cart

Furniture Bedding & Mattresses Décor | Essentials Kitchen | Dining Inspirations Sale

Home > Dining Room Furniture > Dining Tables > Ross Dining Table Walnut

Ross Dining Table Walnut

★★★★★ 78 customer reviews

\$250.20 **\$159.00**

Seller FORTYTWO

Earliest Delivery Tuesday, 13 November 2018

Fulfilled By FORTYTWO

Add to Cart

Add to Wishlist Email to a Friend

100 Day Free Returns

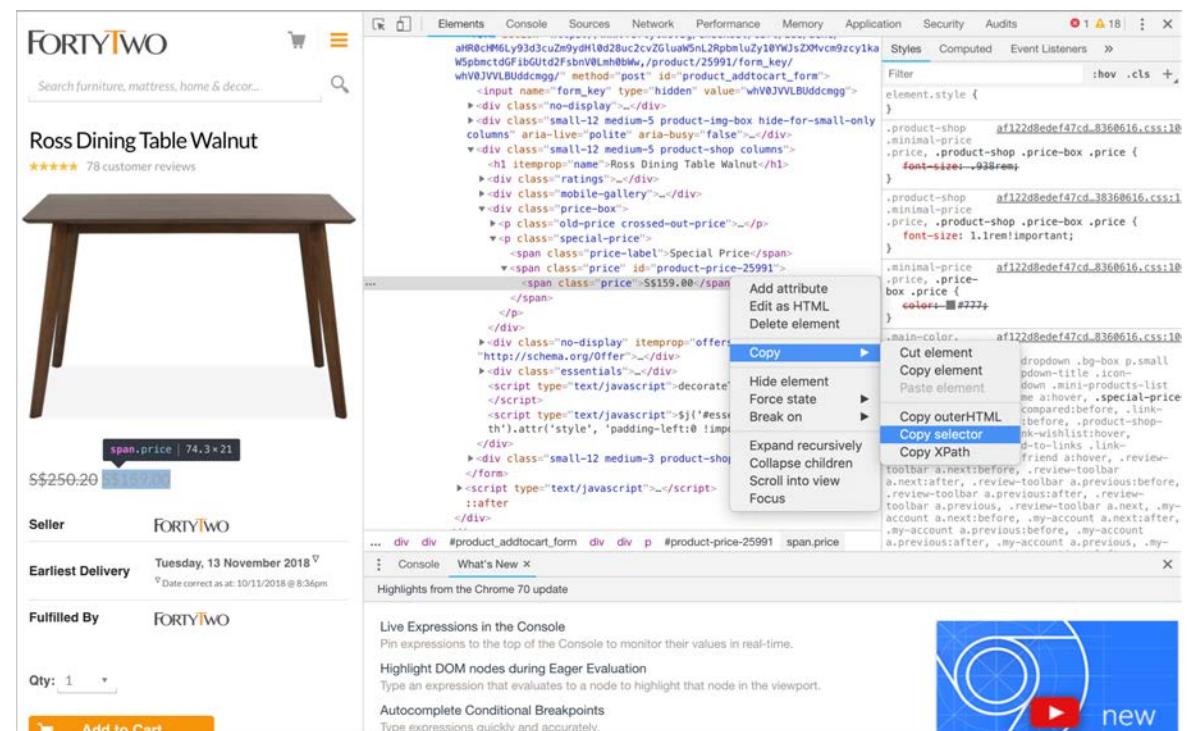
Free Assembly Included

Connecting to the Web

- Get the url

<https://www.fortytwo.sg/dining/dining-tables/ross-dining-table-walnut.html>

- Select the element to extract, right-click "Inspect"
- Right-click "Copy" → "Copy selector"



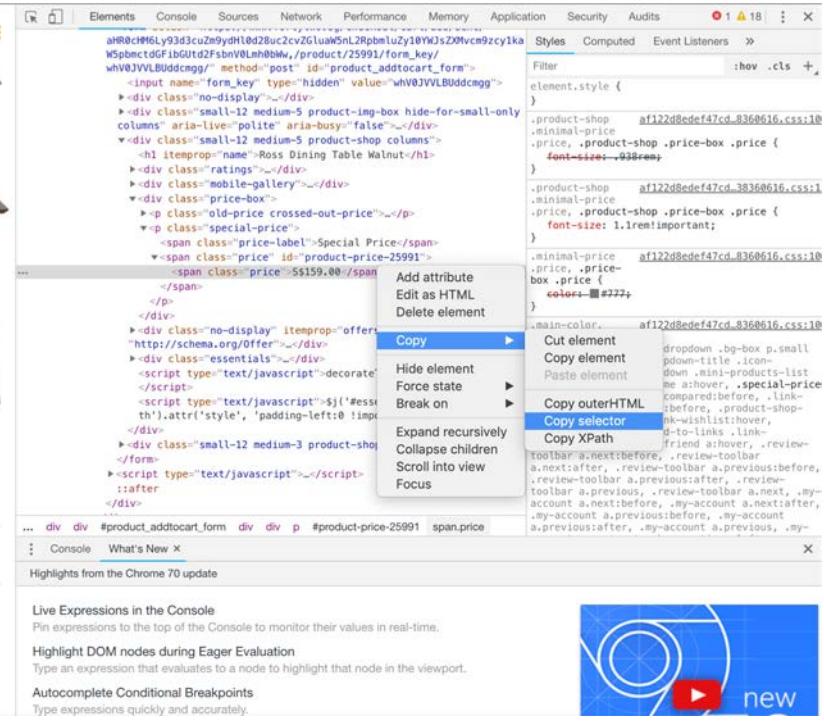
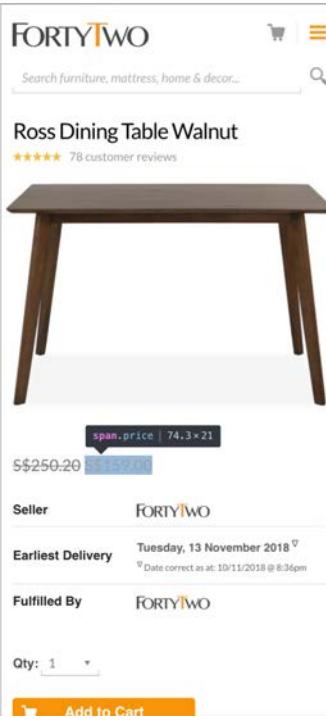
The screenshot shows a web browser displaying a product page for a Ross Dining Table Walnut on the FORTYTWO website. The page includes a search bar, product title, reviews, price (\$\$250.20), seller information, delivery details, and an 'Add to Cart' button. The Chrome DevTools Elements tab is open, highlighting the price element with a tooltip 'span.price | 74.3x21'. A context menu is open over the price element, with 'Copy selector' highlighted.

Connecting to the Web

- Get the url
- Select the element to extract, right-click "Inspect"
- Right-click "Copy" → "Copy selector"

```
import bs4
import requests

requestObj = requests.get("https://www.fortytwo.sg/dining/dining-tables/ross-dining-table-walnut.html")
requestObj.raise_for_status()
soup = bs4.BeautifulSoup(requestObj.text, 'html.parser')
elements = soup.select("#product-price-25991")
print(elements[0].text)
```

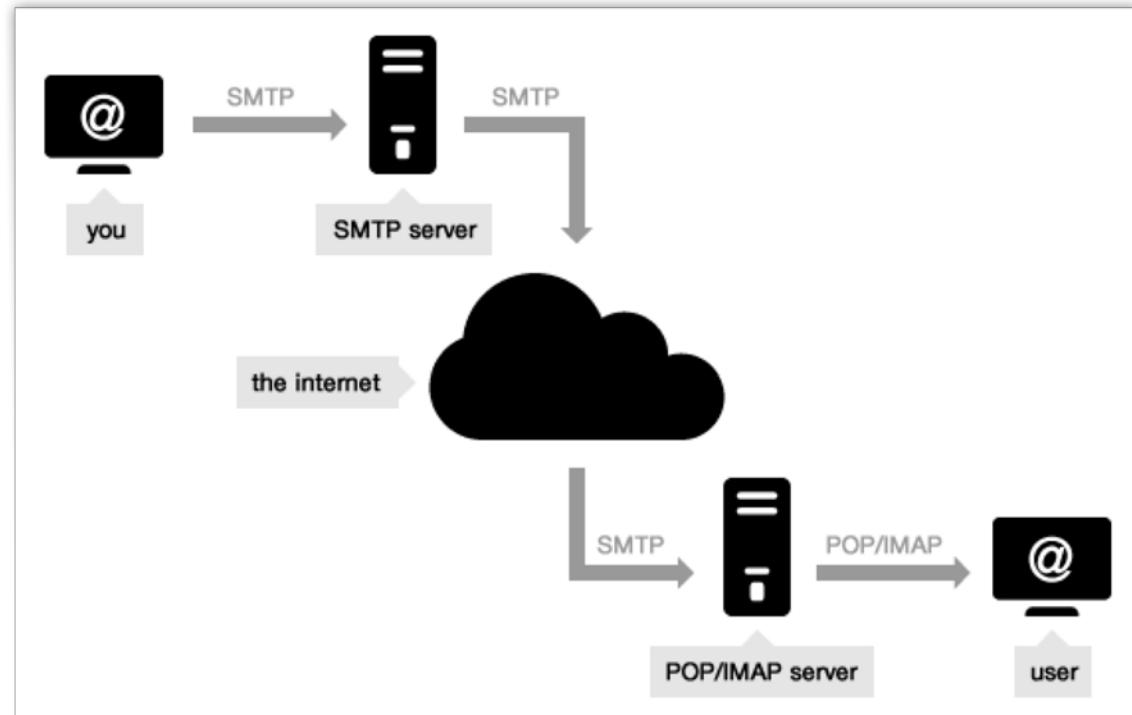


```
C:\Users\kwseow\PycharmProjects\PSA
S$159.00

Process finished with exit code 0
```

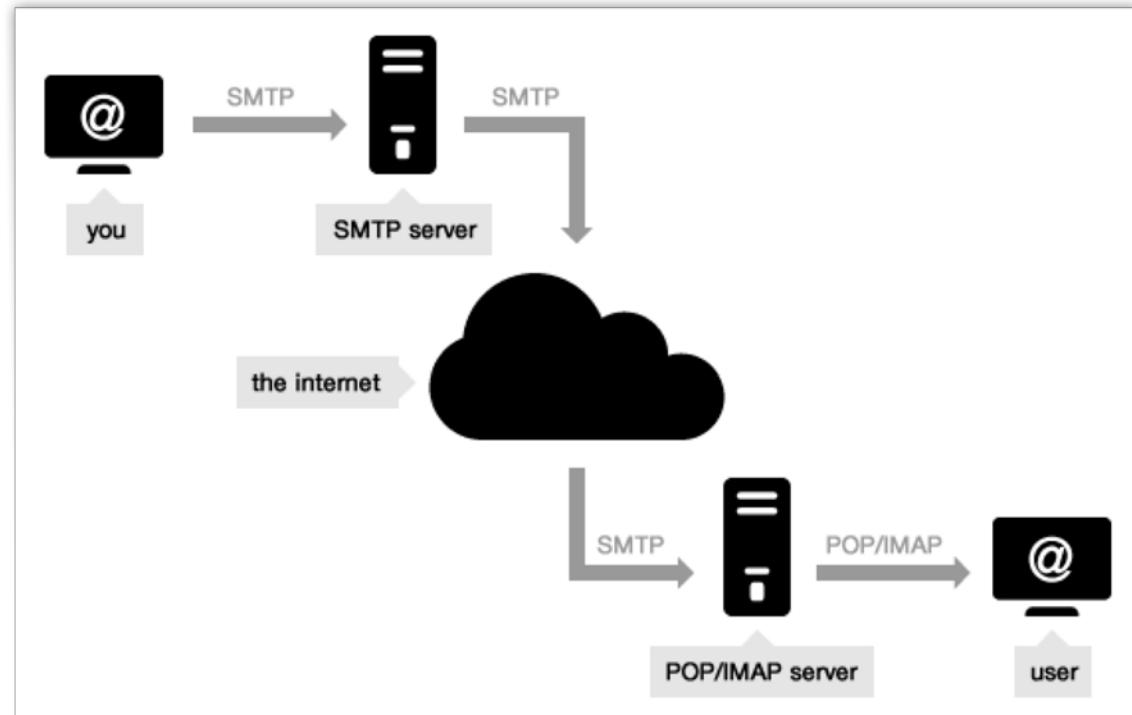
Sending Email

- SMTP (Simple Mail Transfer Protocol) is used for sending and delivering from a client to a server via port 25: it's the **outgoing server**.
- IMAP and POP are two methods to access email. IMAP is the recommended method when you need to check your emails from several different devices, such as a phone, laptop, and tablet.



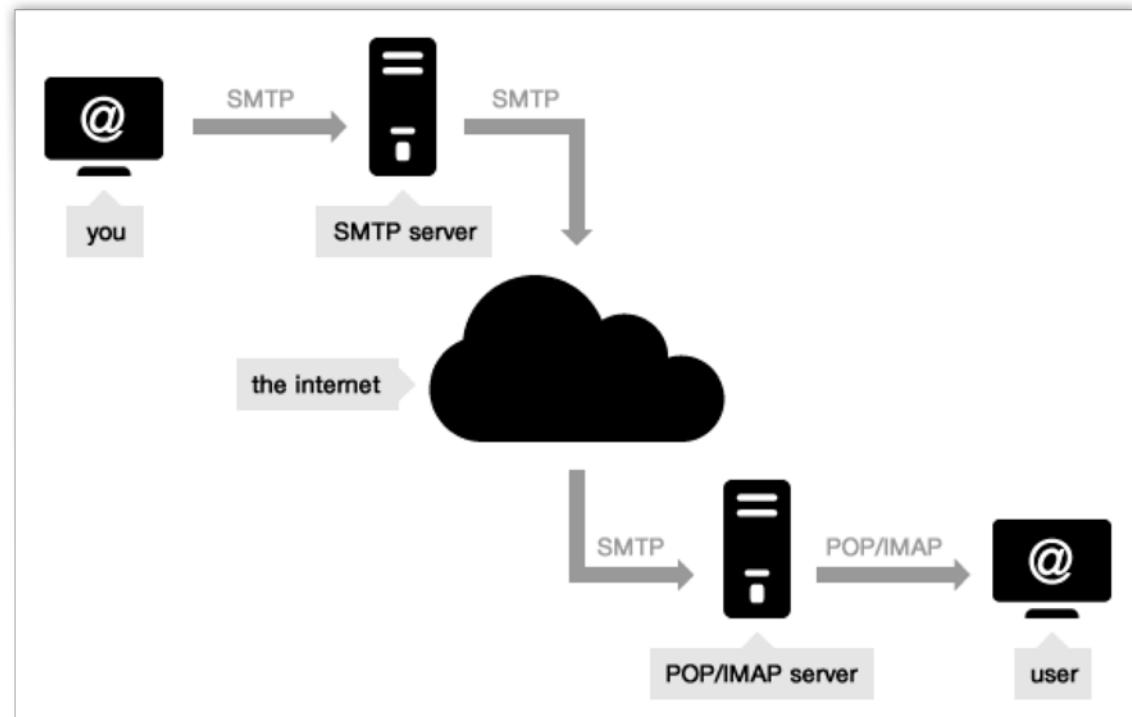
Sending Email

- POP (“Post Office Protocol”) allows the user to pick up the message and download it into his own inbox: it’s the **incoming server**.
- The latest version of the Post Office Protocol is named POP3, and it’s been used since 1996; it uses port 110.



Sending Email

- **Note:** The SMTP servers used when you send your emails- Hotmail, Gmail , Yahoo Mail – are **shared among users**
- Common providers establish some **strict limits** on the number of emails you can send (e.g. Yahoo's restriction is 100 emails per hour).



If you plan to send a bulk email or set up an email campaign you should opt for a professional outgoing email server like turboSMTP, which guarantees a controlled IP and ensure that all your messages reach their destination.

<https://serversmtp.com/what-is-smtp-server/>

Send Email using Gmail

Incoming Mail (IMAP) Server	imap.gmail.com
	Requires SSL: Yes
	Port: 993
Outgoing Mail (SMTP) Server	smtp.gmail.com
	Requires SSL: Yes
	Requires TLS: Yes (if available)
	Requires Authentication: Yes
	Port for SSL: 465
	Port for TLS/STARTTLS: 587
Full Name or Display Name	Your name
Account Name, User name, or Email address	Your full email address
Password	Your Gmail password

Send Email using Gmail

- Import smtplib module
- Specify Gmail email & password, receiver's email address, email title & content
- Connect to SMTP server using Port 587
- Call starttls() to enable encryption for your connection
- Login using email and password
- Call sendmail()
- Call quit() to disconnect from the SMTP server

```
import smtplib

sender_email_address = "your_email_address@gmail.com"
sender_email_password = "xxxxxxxxxxxxxx"
receiver_email_address = "another_email_address@gmail.com"
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."
```

```
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."
```

➤ The start of the email body must begin with "Subject: " for the subject line. The "\n" newline character separates the subject line from the main body content.

```
print("Trying to connect to Gmail SMTP server")
smtpObj = smtplib.SMTP("smtp.gmail.com", 587)
smtpObj.starttls()

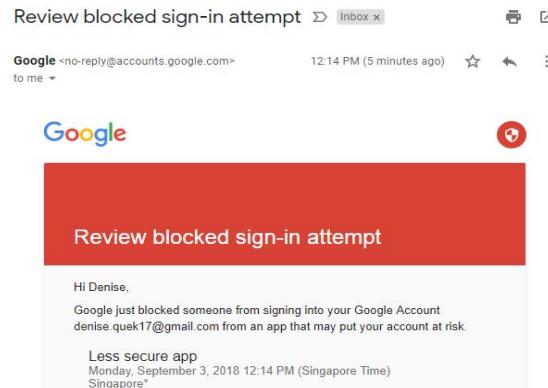
print("Connected. Logging in...")
smtpObj.login(sender_email_address, sender_email_password)

smtpObj.sendmail(sender_email_address, receiver_email_address, email_title_content)
print("Email sent successfully...")

smtpObj.quit()
```

Send Email using Gmail

- Google may block attempted sign-in from unknown devices that don't meet their security standards!



```
C:\Users\denise_quek\AppData\Local\Programs\Python\Python37\python.exe D:/CET_Python/Denise/TestEmail.py
Trying to connect to Gmail SMTP server
Connected. Logging in...
Traceback (most recent call last):
  File "D:/CET_Python/Denise/TestEmail.py", line 13, in <module>
    smtpObj.login(sender_email_address, sender_email_password)
  File "C:\Users\denise_quek\AppData\Local\Programs\Python\Python37\lib\smtplib.py", line 730, in login
    raise last_exception
  File "C:\Users\denise_quek\AppData\Local\Programs\Python\Python37\lib\smtplib.py", line 721, in login
    initial_response_ok=initial_response_ok)
  File "C:\Users\denise_quek\AppData\Local\Programs\Python\Python37\lib\smtplib.py", line 642, in auth
    raise SMTPAuthenticationError(code, resp)
smtplib.SMTPAuthenticationError: (534, b'5.7.9 Application-specific password required. Learn more at\n5.7.9')

Process finished with exit code 1
```

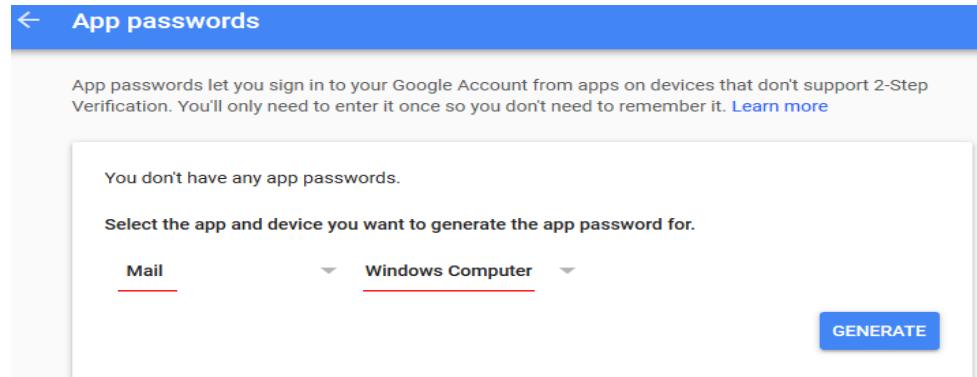
Send Email using Gmail

Steps To Create Google App Password

Step 1: Login to Gmail. Go to Account → Signing in to Google

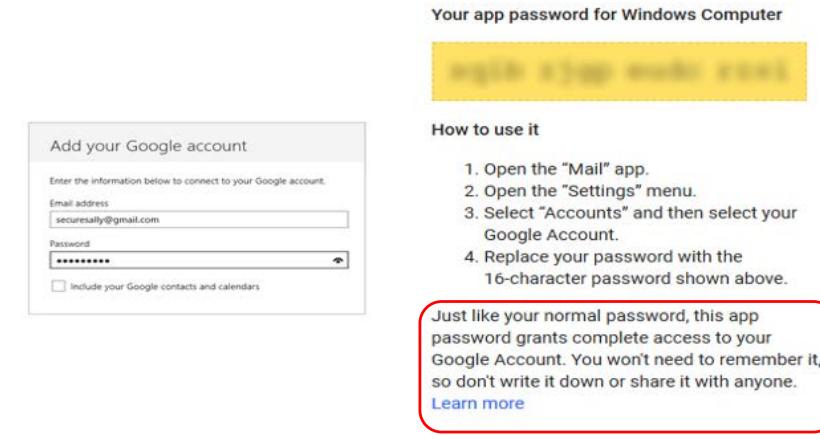
Step 2: Make sure that 2-Step Verification is on

Step 3: Create an App password



The screenshot shows the 'App passwords' section of the Google Account settings. At the top, there's a blue header bar with a back arrow and the text 'App passwords'. Below it, a message states: 'App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it.' A 'Learn more' link is provided. The main content area has a message: 'You don't have any app passwords.' It includes a dropdown menu to 'Select the app and device you want to generate the app password for.' The 'Mail' tab is selected, and 'Windows Computer' is chosen from the dropdown. A 'GENERATE' button is located at the bottom right.

Generated app password



The screenshot shows the 'Generated app password' page. At the top, it says 'Your app password for Windows Computer' and displays a long, yellow-redacted password. Below it, a section titled 'How to use it' lists four steps: 1. Open the "Mail" app. 2. Open the "Settings" menu. 3. Select "Accounts" and then select your Google Account. 4. Replace your password with the 16-character password shown above. A red box highlights a note in a callout bubble: 'Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.' A 'Learn more' link is also present in the note. At the bottom right, there's a 'DONE' button.

<https://support.google.com/accounts/answer/185833?hl=en>

Send Email using Gmail

- Replace your actual password with the App password

```
import smtplib

sender_email_address = "your_email_address@gmail.com"
sender_email_password = "xxxxxxxxxxxxxx"
receiver_email_address = "another_email_address@gmail.com"
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."
```

- Run your email program

```
C:\Users\denise quek\AppData\Local\Programs\Python\Python37\python.exe D:/CET_Python/Denise/TestEmail.py
Trying to connect to Gmail SMTP server
Connected. Logging in...
Email sent successfully...

Process finished with exit code 0
```

Send Email using Gmail

- Send email to students who were absent

	A	B	C
1	Student	Email	Status
2	Alicia	code.musically@gmail.com	Present
3	Bryan	code.musically@gmail.com	Present
4	Carol	code.musically@gmail.com	Absent
5	David	code.musically@gmail.com	Absent
6	Evelyn	code.musically@gmail.com	Present
7			

```
1  #! python3
2
3  import openpyxl, smtplib
4
5  def sendEmail(name, emailTo):
6      email_body = "Subject: Your attendance. \nDear %s, \nYou were absent for class.\n" %(name)
7
8      smtpObj = smtplib.SMTP("smtp.gmail.com", 587)
9      smtpObj.starttls()
10     smtpObj.login("code.musically@gmail.com", "xxxxxxxxxxxx")
11     smtpObj.sendmail('code.musically@gmail.com', emailTo, email_body)
12
13     smtpObj.quit()
```

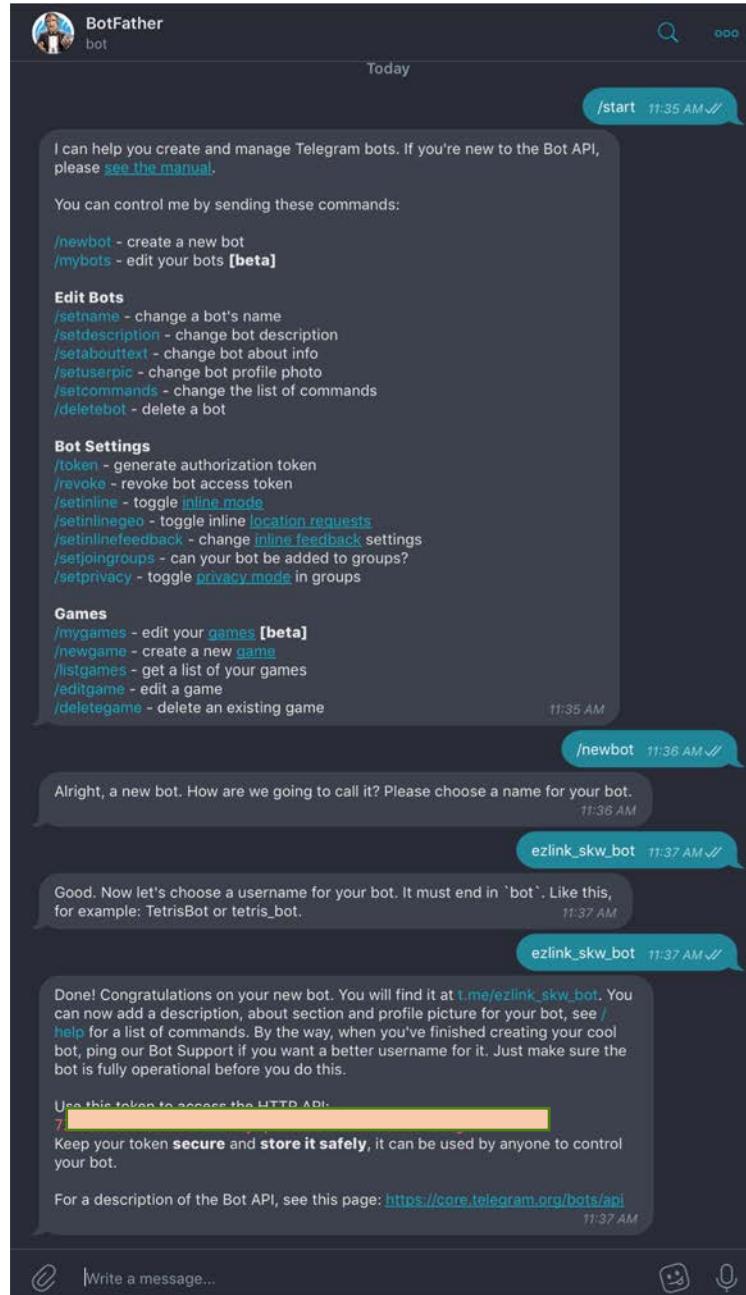
Send Email using Gmail

- Send email to students who were absent

```
16    workbook = openpyxl.load_workbook("D:\\CET_Python\\students_attendance.xlsx")
17    sheet = workbook["Sheet1"]
18
19    max_row = sheet.max_row
20    max_column = sheet.max_column
21
22    for i in range(1, max_row+1):
23
24        attendance = sheet.cell(row=i, column=3).value
25
26        if attendance == "Absent":
27            name = sheet.cell(row=i, column=1).value
28            email = sheet.cell(row=i, column=2).value
29
30            print(name + " is absent.")
31            sendEmail(name, email)
32            print("Email sent to " + email)
33            print()
34
```

Telegram Bot

- Create a new bot using BotFather:
<https://telegram.me/botfather>
- Run /start to start the interface and then create a new bot with /newbot
- The interface will ask you the name of the bot and a username, which should be unique
- The Telegram channel of your bot—
<https://t.me/<yourusername>>
- A token to allow access the bot. Copy it as it will be used later



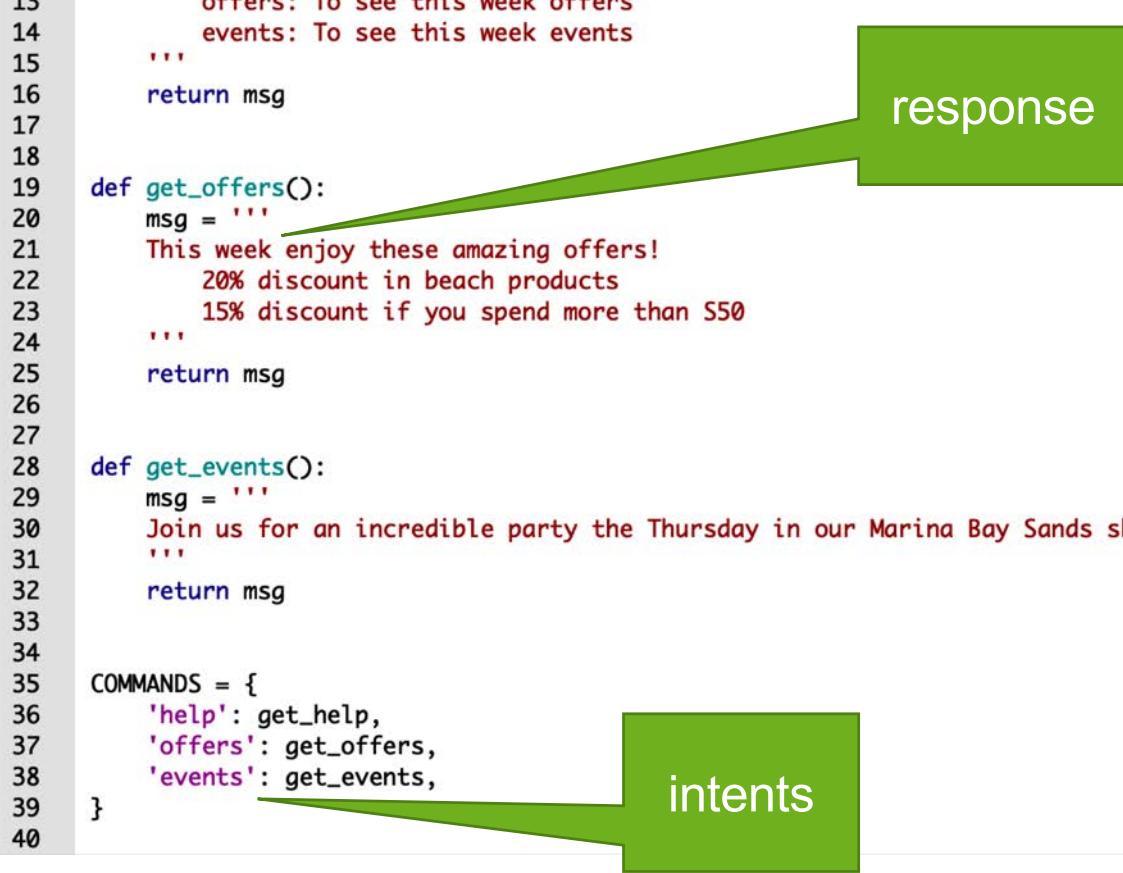
Telegram Bot

- Install telepot
 - pip install telepot
- Update your TOKEN
- Define intent
- Define response

```

1 import time
2 import telepot
3 from telepot.loop import MessageLoop
4 from telepot.delegate import per_chat_id, create_open, pave_event_space
5
6 TOKEN = '<YOUR TOKEN>'
7
8 # Define the information to return per command
9 def get_help():
10     msg = """
11         Use one of the following commands:
12             help: To show this help
13             offers: To see this week offers
14             events: To see this week events
15         ...
16     return msg
17
18
19 def get_offers():
20     msg = """
21         This week enjoy these amazing offers!
22             20% discount in beach products
23             15% discount if you spend more than $50
24         ...
25     return msg
26
27
28 def get_events():
29     msg = """
30         Join us for an incredible party the Thursday in our Marina Bay Sands shop!
31         ...
32     return msg
33
34
35 COMMANDS = {
36     'help': get_help,
37     'offers': get_offers,
38     'events': get_events,
39 }
40

```



The code is annotated with two green callout boxes. One box, labeled "response", points to the section where each command returns a string message. The other box, labeled "intents", points to the section where each command is mapped to a specific function in the COMMANDS dictionary.

Telegram Bot

```

44 class MarketingBot(telepot.helper.ChatHandler):
45
46     def open(self, initial_msg, seed):
47         self.sender.sendMessage(get_help())
48         # prevent on_message() from being called on the initial message
49         return True
50
51     def on_chat_message(self, msg):
52         # If the data sent is not test, return an error
53         content_type, chat_type, chat_id = telepot.glance(msg)
54
55         if content_type != 'text':
56             self.sender.sendMessage("I don't understand you. "
57                                   "Please type 'help' for options")
58             return
59
60         # Make the commands case insensitive
61         command = msg['text'].lower()
62         if command not in COMMANDS:
63             self.sender.sendMessage("I don't understand you. "
64                                   "Please type 'help' for options")
65             return
66
67         message = COMMANDS[command]()
68         self.sender.sendMessage(message)
69
70     def on_idle(self, event):
71         self.close()
72
73     def on_close(self, event):
74         # Add any required cleanup here
75         pass
76
77
78     # Create and start the bot
79     bot = telepot.DelegatorBot(TOKEN, [
80         pave_event_space(),
81             per_chat_id(), create_open, MarketingBot, timeout=10],
82     ])
83     MessageLoop(bot).run_as_thread()
84     print('Listening ...')
85
86     while 1:
87         time.sleep(10)

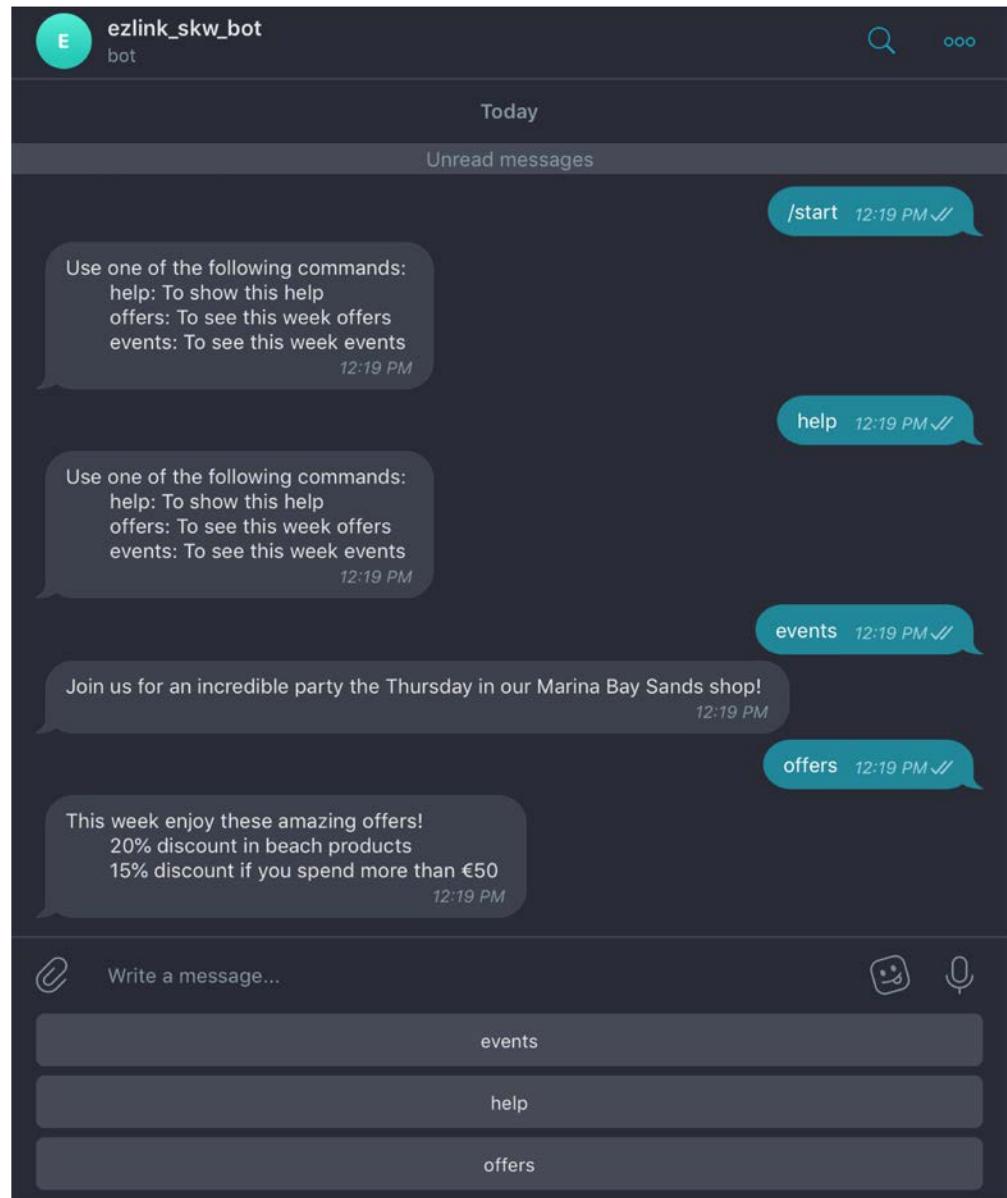
```

Process the
message

Telegram Bot



Telegram Bot – Custom Keyboard



Telegram Bot – Custom Keyboard

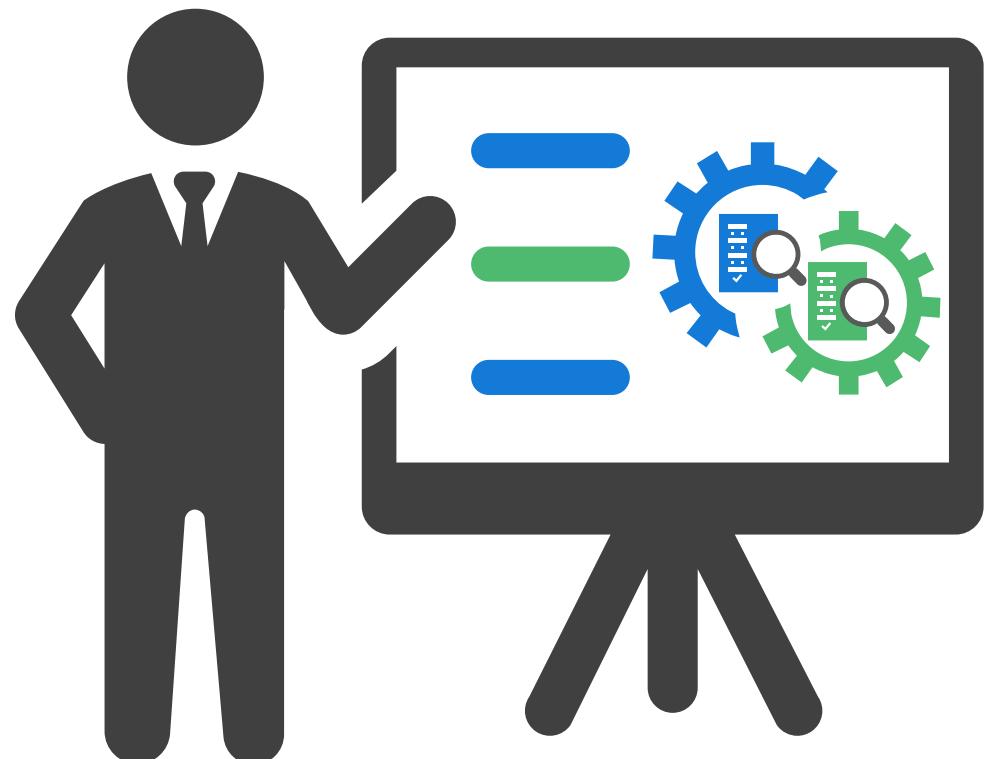
```

1 import time
2 import telepot
3 from telepot.loop import MessageLoop
4 from telepot.delegate import per_chat_id, create_open, pave_event_space
5 from telepot.namedtuple import ReplyKeyboardMarkup, KeyboardButton
6
7 #TOKEN = '<YOUR TOKEN>'
8 TOKEN = '726962401:AAHoiAXriizyEpBds9cJVW3eJHocb01o2ig'
9
10
11 # Define the information to return per command
12 def get_help():
13     msg = """
14         Use one of the following commands:
15             help: To show this help
16             offers: To see this week offers
17             events: To see this week events
18             ...
19
20     return msg
21
22 def get_offers():
23     msg = """
24         This week enjoy these amazing offers!
25             20% discount in beach products
26             15% discount if you spend more than €50
27             ...
28
29     return msg
30
31 def get_events():
32     msg = """
33         Join us for an incredible party the Thursday in our Marina Bay Sands shop!
34             ...
35
36     return msg
37
38 COMMANDS = {
39     'help': get_help,
40     'offers': get_offers,
41     'events': get_events,
42 }
43
44 # Create a custom keyboard with only the valid responses
45 keys = [[KeyboardButton(text=text)] for text in COMMANDS]
46 KEYBOARD = ReplyKeyboardMarkup(keyboard=keys)

```

Telegram Bot – Custom Keyboard

```
45 class MarketingBot(telepot.helper.ChatHandler):
46
47     def open(self, initial_msg, seed):
48         self.sender.sendMessage(get_help(), reply_markup=KEYBOARD)
49         # prevent on_message() from being called on the initial message
50         return True
51
52     def on_chat_message(self, msg):
53         # If the data sent is not text, return an error
54         content_type, chat_type, chat_id = telepot.glance(msg)
55
56         if content_type != 'text':
57             self.sender.sendMessage("I don't understand you. "
58                                     "Please type 'help' for options",
59                                     reply_markup=KEYBOARD)
60             return
61
62         # Make the commands case insensitive
63         command = msg['text'].lower()
64         if command not in COMMANDS:
65             self.sender.sendMessage("I don't understand you. "
66                                     "Please type 'help' for options",
67                                     reply_markup=KEYBOARD)
68             return
69
70         message = COMMANDS[command]()
71         self.sender.sendMessage(message, reply_markup=KEYBOARD)
72
73     def on__idle(self, event):
74         self.close()
75
76
77     # Create and start the bot
78     bot = telepot.DelegatorBot(TOKEN, [
79         pave_event_space()(  # piped
80             per_chat_id(), create_open, MarketingBot, timeout=10),
81     ])
82     MessageLoop(bot).run_as_thread()
83     print('Listening ...')
84
85     while 1:
86         time.sleep(10)
```



Day 2 Summary

- ✓ *Read and write files*
- ✓ *Copying, moving and deleting files and folders*
- ✓ *Working with Excel*
- ✓ *Processing CSV files*
- ✓ *Working with PDF*

- ✓ *Image Processing: loading, scaling, watermark, applying filters*
- ✓ *Generating charts*
- ✓ *Connecting to the web, scrapping*
- ✓ *Sending emails*
- ✓ *Telegram bots*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: <http://bit.ly/2GVM07s>

Where to go from here?

Getting started step by step

<http://www.python.org/about/gettingstarted/>

Run through the python tutorials:

<http://docs.python.org/tutorial/index.html>

Keep the API doc under your pillow:

<http://docs.python.org/library/index.html>

Advanced examples:

<http://www.diveintopython.org/toc/index.html>

Where to go from here?

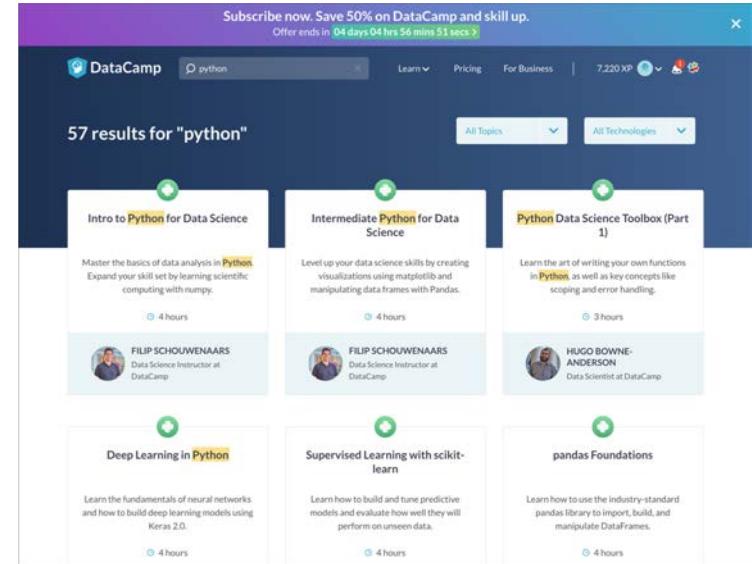
MOOC:
DataCamp

<https://www.datacamp.com/>

Edx

<https://www.edx.org/>

Udemy (freemium course)
<https://t.me/freecourse>



The screenshot shows a DataCamp search results page for "python". At the top, a purple banner says "Subscribe now. Save 50% on DataCamp and skill up. Offer ends in 04 days 04 hrs 56 mins 51 secs". Below the banner, the search bar shows "python". The results section displays six course cards:

- Intro to Python for Data Science**: Master the basics of data analysis in Python. Expand your skill set by learning scientific computing with numpy. 4 hours. Instructor: FILIP SCHOUWENAARS.
- Intermediate Python for Data Science**: Level up your data science skills by creating visualizations using matplotlib and manipulating data frames with Pandas. 4 hours. Instructor: FILIP SCHOUWENAARS.
- Python Data Science Toolbox (Part 1)**: Learn the art of writing your own functions in Python as well as key concepts like scoping and error handling. 3 hours. Instructor: HUGO BOVNE-ANDERSON.
- Deep Learning in Python**: Learn the fundamentals of neural networks and how to build deep learning models using Keras 2.0. 4 hours.
- Supervised Learning with scikit-learn**: Learn how to build and tune predictive models and evaluate how well they will perform on unseen data. 4 hours.
- pandas Foundations**: Learn how to use the industry-standard pandas library to import, build, and manipulate DataFrames. 3 hours.



The screenshot shows the edX homepage. At the top, there is a navigation bar with links for Courses, Programs & Degrees, Schools & Partners, and edX for Business. A search bar and sign-in/register buttons are also present. The main banner features a woman smiling and the text "Accelerate your future. Learn anytime, anywhere." Below the banner is a search bar with the placeholder "What do you want to learn?". At the bottom of the page, there is a promotional banner for Cyber Monday with the text "THE COUNTDOWN IS ON! Get 15% off your purchase." and a "Start Exploring" button. Logos for various partner institutions like MIT, Harvard, Berkeley, and The University of Texas System are displayed at the bottom.

**Online Evaluation
Fundamentals of Python for Beginners
(7-8 March 2019)**



<https://tinyurl.com/yb3u4tnv>

Thank you