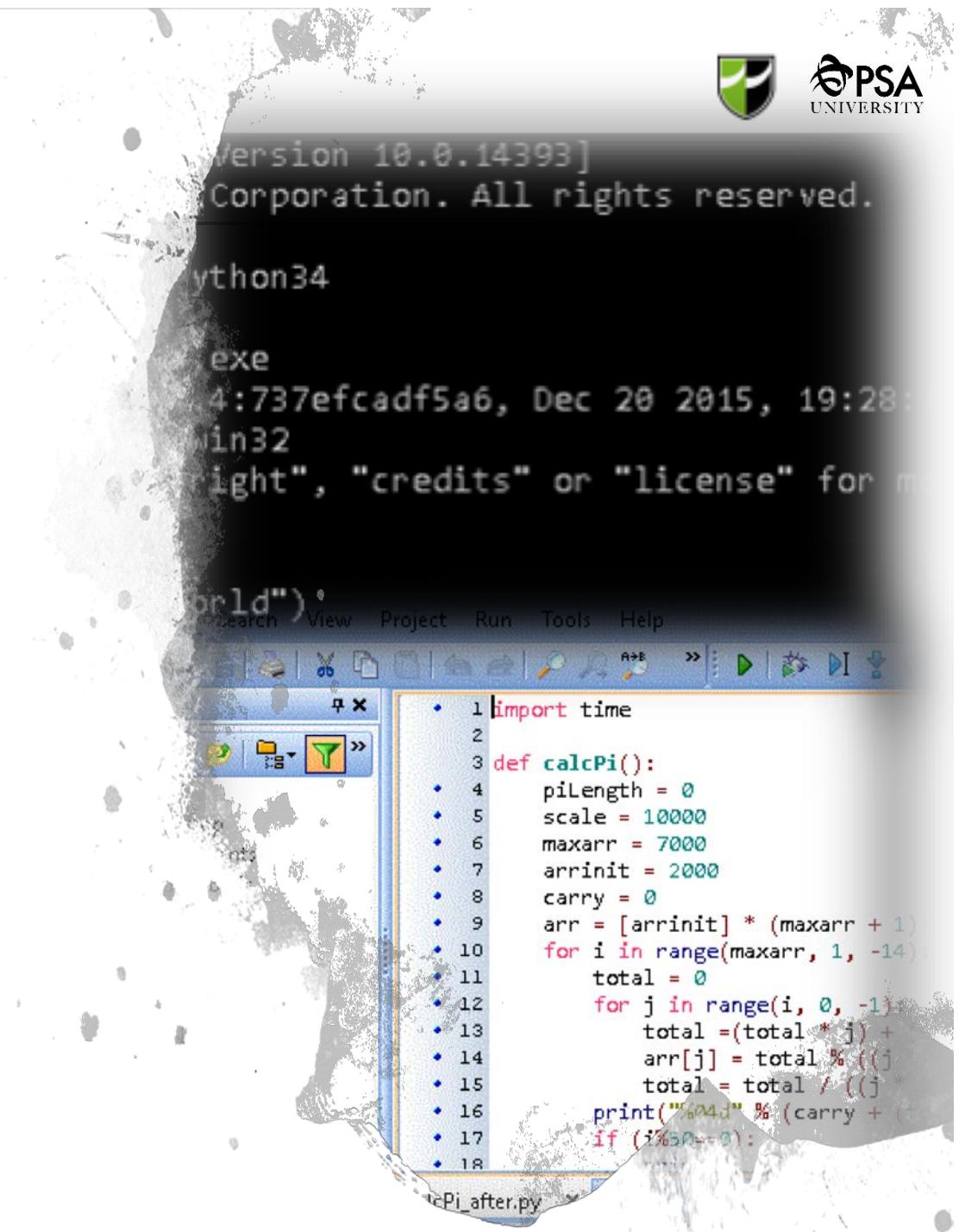


Introductory Programming in Python

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
```

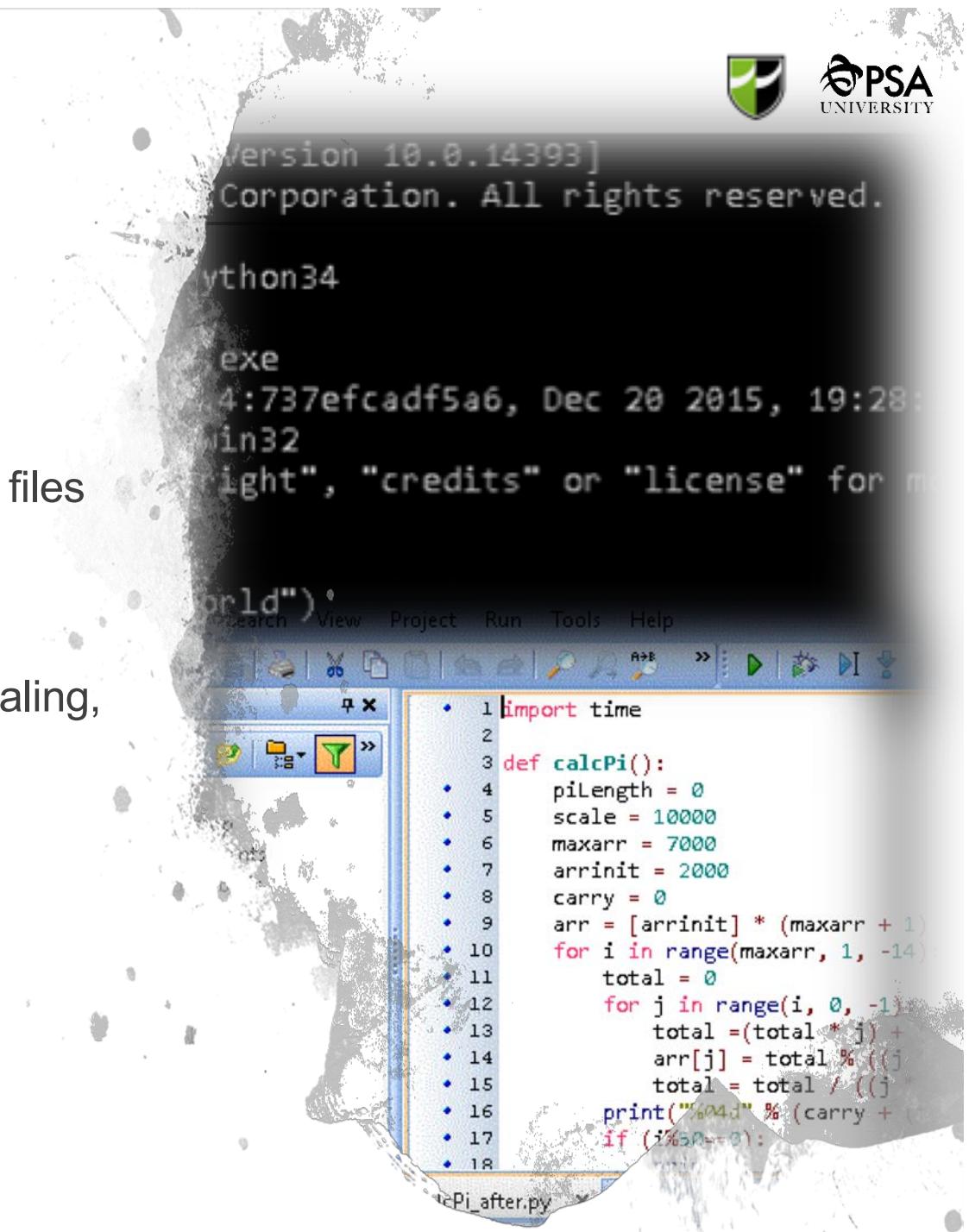
In the foreground, a Python code editor window titled "cpi_after.py" is displayed. The code is a script to calculate pi using a specific algorithm. The code is as follows:

```
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
- Image processing: loading, scaling, watermark, applying filters
- Connecting to the Web
- Sending emails
- Graphical User Interface
- AI – Facial Recognition



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, initializes variables for arrlength, scale, maxarr, arrinit, and carry, and uses nested loops to calculate the value of pi.

```
1 import time
2
3 def calcPi():
4     arrlength = 0
5     scale = 10000
6     maxarr = 7000
7     arrinit = 2000
8     carry = 0
9
10    arr = [arrinit] * (maxarr + 1)
11    for i in range(maxarr, 1, -14):
12        total = 0
13        for j in range(i, 0, -1):
14            total = (total * j) +
15            arr[j] = total % ((j *
16            total = total / ((j *
17            print("%60.4d" % (carry +
18
if (i%50==0):
```

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.

```
>>> helloFile = open("D:\\python_training\\hello.txt")
>>> helloFile.read()
'Hello World\nHow are you?'
>>> helloFile.close()

>>> helloFile = open("D:\\python_training\\hello.txt")
>>> content = helloFile.read()
>>> print(content)
Hello World
How are you?

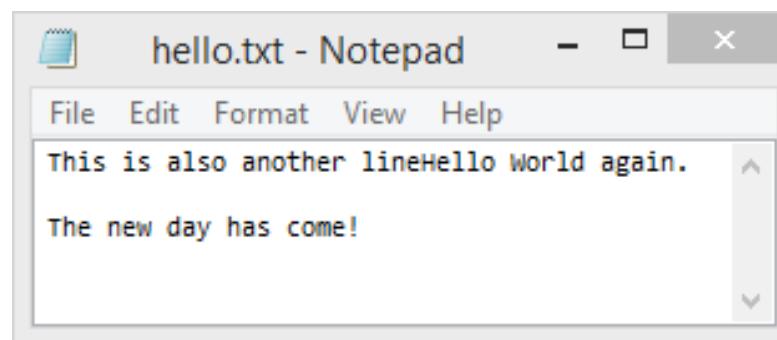
>>> helloFile = open("D:\\python_training\\hello.txt")
>>> helloFile.readlines()
['Hello World\\n', 'How are you?']
>>> helloFile.close()
```

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

```
>>> helloFile = open("D:\\python_training\\hello.txt", "w")
>>> helloFile.write("This is also another line")
25
>>> helloFile.close()
>>> helloFile = open("D:\\python_training\\hello.txt")
>>> helloFile.read()
'This is also another line'

>>> helloFile = open("D:\\python_training\\hello.txt", "a")
>>> helloFile.write("Hello World again.")
18
>>> helloFile.write("\n\nThe new day has come!")
23
>>> helloFile.close()
```



Shelve module

- The `shelve` module can store Python values in a binary file.
- `shelve.open()` returns a dictionary-like shelf value

```
>>> import shelve
>>> shelfFile = shelve.open("D:\\python_training\\mydata")
>>> shelfFile["books"] = ["java", "python", "php"]
>>> shelfFile.close()
>>> shelfFile = shelve.open("D:\\python_training\\mydata")
>>> shelfFile["books"]
['java', 'python', 'php']
>>> shelfFile.close()
```

Data (D:) ▶ python_training

Name

 hello.txt
 mydata.bak
 mydata.dat
 mydata.dir

3 files will
be
created

Copying and Moving Files

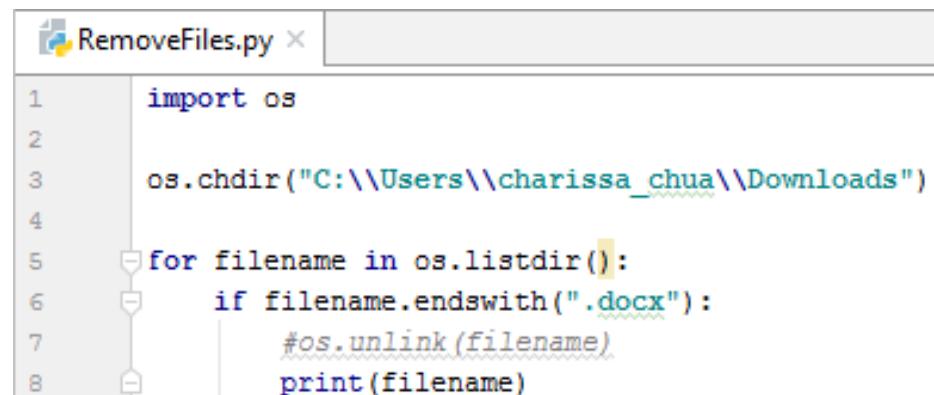
- Shutil.copy(src, dst) – Copy the file *src* to the file or directory *dst*
- Shutil.copytree(src, dst) - Recursively copy an entire directory tree rooted at *src*.
- Shutil.move(src, dst) - Recursively move a file or directory (*src*) to another location (*dst*).

```
>>> import shutil
>>> shutil.copy("D:\\\\folder1\\\\hello.txt", "D:\\\\folder2")
'D:\\\\folder2\\\\hello.txt'
>>> shutil.copy("D:\\\\folder1\\\\hello.txt", "D:\\\\folder2\\\\new_hello.txt")
'D:\\\\folder2\\\\new_hello.txt'
>>> shutil.copytree("D:\\\\folder2", "D:\\\\folder2_backup")
'D:\\\\folder2_backup'
>>> shutil.move("D:\\\\folder2\\\\hello.txt", "D:\\\\folder2\\\\anotherfolder")
'D:\\\\folder2\\\\anotherfolder\\\\hello.txt'
>>> shutil.move("D:\\\\folder2\\\\anotherfolder\\\\hello.txt", "D:\\\\folder2\\\\anotherfolder\\\\newhello.txt")
'D:\\\\folder2\\\\anotherfolder\\\\newhello.txt'
```

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

```
>>> import os
>>> os.getcwd()
'C:\\\\Users\\\\charissa_chua\\\\PycharmProjects\\\\ExerciseOne'
>>> os.rmdir("D:\\\\folder2_backup")
Traceback (most recent call last):
  File "<input>", line 1, in <module>
OSError: [WinError 145] The directory is not empty: 'D:\\\\folder2_backup'
>>> import shutil
>>> shutil.rmtree("D:\\\\folder2_backup")
```



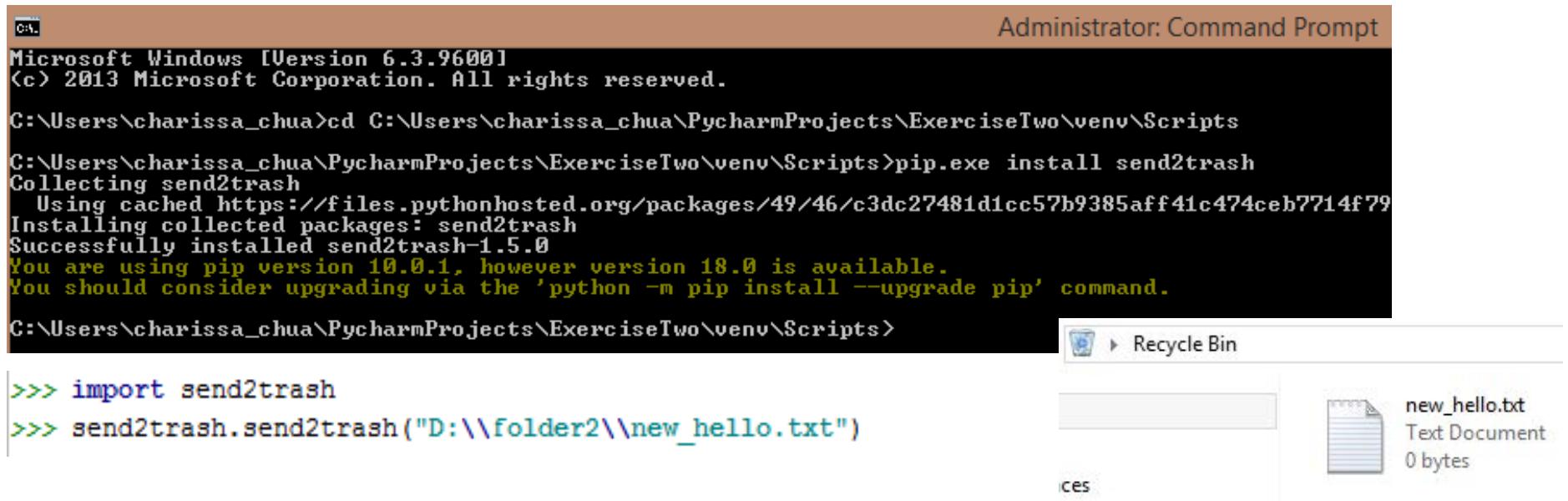
The screenshot shows a PyCharm code editor window titled "RemoveFiles.py". The code is as follows:

```
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)
```

The line `#os.unlink(filename)` is highlighted with a yellow background.

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

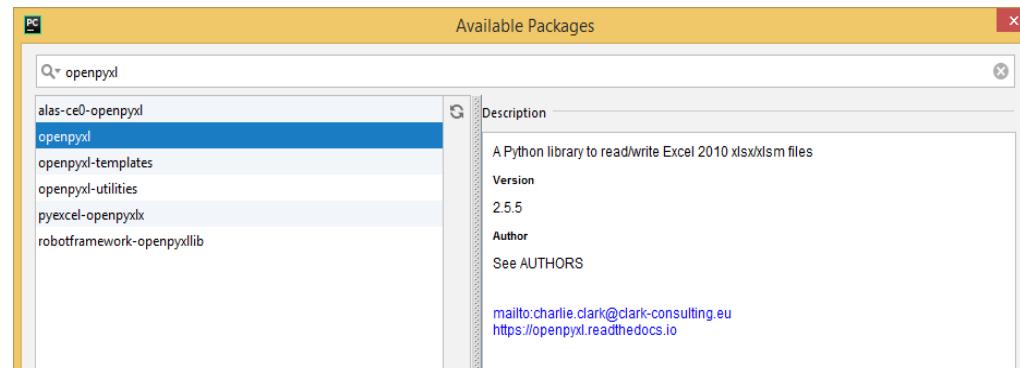
```
FileUtils.py x
1 import os
2 import shutil
3
4 for folderName, subfolders, filenames in os.walk("C:\\\\Users\\\\charissa_chua\\\\Downloads"):
5     print("The folder is " + folderName)
6     print("The subfolders in " + folderName + " are: " + str(subfolders))
7     print("The filenames in " + folderName + " 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
- Make sure the file is available - students_attendance.xlsx



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

The screenshot shows the PyCharm IDE's package manager interface. A search bar at the top left contains the text "openpyxl". Below the search bar, a list of packages is displayed, with "openpyxl" highlighted. To the right of the list, a detailed view panel shows the following information:
Description: A Python library to read/write Excel 2010 xlsx/xlsm files
Version: 2.5.5
Author: See AUTHORS
Contact: mailto:charlie.clark@clark-consulting.eu, <https://openpyxl.readthedocs.io>

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  #! python3
2
3  import openpyxl
4
5  workbook = openpyxl.load_workbook("D:\\CET_Python\\students_attendance.xlsx")
6  sheet = workbook["Sheet1"]
7
8  max_row = sheet.max_row
9  max_column = sheet.max_column
10
11 for i in range(1, max_row+1):
12
13     attendance = sheet.cell(row=i, column=3).value
14
15     if attendance == "Absent":
16         name = sheet.cell(row=i, column=1).value
17         email = sheet.cell(row=i, column=2).value
18         print(name + " is absent.")
19
```

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.
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)

```

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()

W65Za.csv - Notepad

AARON,D,X,X,X,X
BERT,A,X,X,X,X
BRADLEY,C,A,X,X,B

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>

 **pillow**

The friendly PIL fork

Welcome

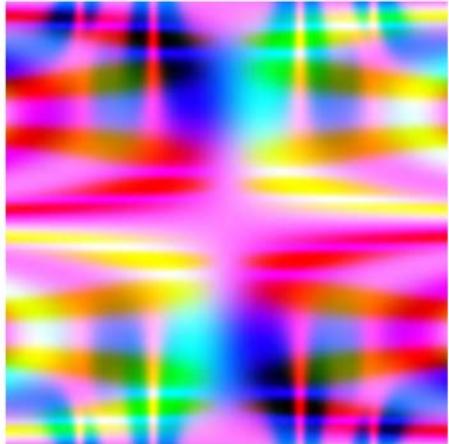
This is the home of [Pillow](#), the friendly PIL fork. PIL is the [Python Imaging Library](#). If you have ever worried or wondered about the future of PIL, please stop. [We're here](#) to save the day.

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\Temp\tmp10000\image-processing\> python process_all_images.py
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/Day2/day2.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

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

```

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 - 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 - converting

Maybe you want to keep all your photos in the same format.
We have some gif files and maybe you would have bmp or png images.

Pillow understands the output file, and will convert if the output file is different from the input.

fname1	fname2	
holiday.gif	->	holiday.jpg

How can we convert the string holiday.gif to holiday.jpg ?

Let's open the real time Interpreter

```
>>> fname1 = "holiday.gif"
>>> fname2 = fname1.split(".")[0] + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

```
>>> fname1 = "holiday.gif"
>>> f, e = os.path.splitext(fname1)
>>> fname2 = f + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

Image processing - converting

Whichever method you pick, we can use it in our function.

Change the outFileNames declaration so that file contains similar content as what we did with fname2.

When you run your program, you should see that all the images in the output folder are .jpg files.

```
>>> fname1 = "holiday.gif"
>>> fname2 = fname1.split(".")[0] + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

```
>>> fname1 = "holiday.gif"
>>> f, e = os.path.splitext(fname1)
>>> fname2 = f + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

Image processing - converting

You can shorten those 3 lines into one by :

```
outFileName =  
os.path.join(outFolder,os.path.split  
ext(file)[0]+".jpg")
```

`os.path.splitext(file)` returns a list.
We are only interested in `f`, which is the first item in the list.
Hence `os.path.splitext(file)[0]` is equal to `f`.

```
1 import os  
2 from PIL import Image  
3  
4 where = "Day2Resource\\img\\"  
5 outFolder = "Day2Resource\\out"  
6  
7 def convertImages():  
8     for root, dirs, files in os.walk(where):  
9         for file in files:  
10             fullname = os.path.join(root,file)  
11             print(fullname)  
12             if (file.lower().endswith("jpg") or  
13                 file.lower().endswith("bmp")):  
14                 im = Image.open(fullname)  
15                 f, e = os.path.splitext(file)  
16                 fname2 = f + ".jpg"  
17                 outfilename = os.path.join(outFolder, fname2)  
18                 print(outfilename)  
19                 im.save(outfilename, "jpeg")  
20  
21 convertImages()
```

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 - 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.

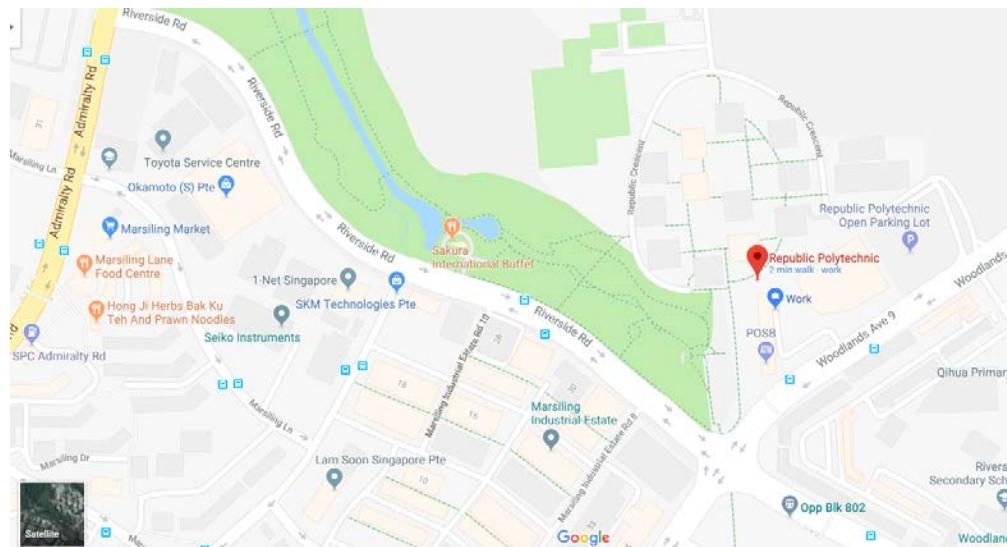
```
1 from PIL import Image
2
3 def watermark(im, mark, position):
4     layer = Image.new("RGBA", im.size, (0,0,0,0))
5     layer.paste(mark, position)
6     return Image.composite(layer, im, layer)
7
8
9 im = Image.open("Day2Resource\\img\\clungup.jpg")
10 mark = Image.open("Day2Resource\\watermark.png")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

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
```

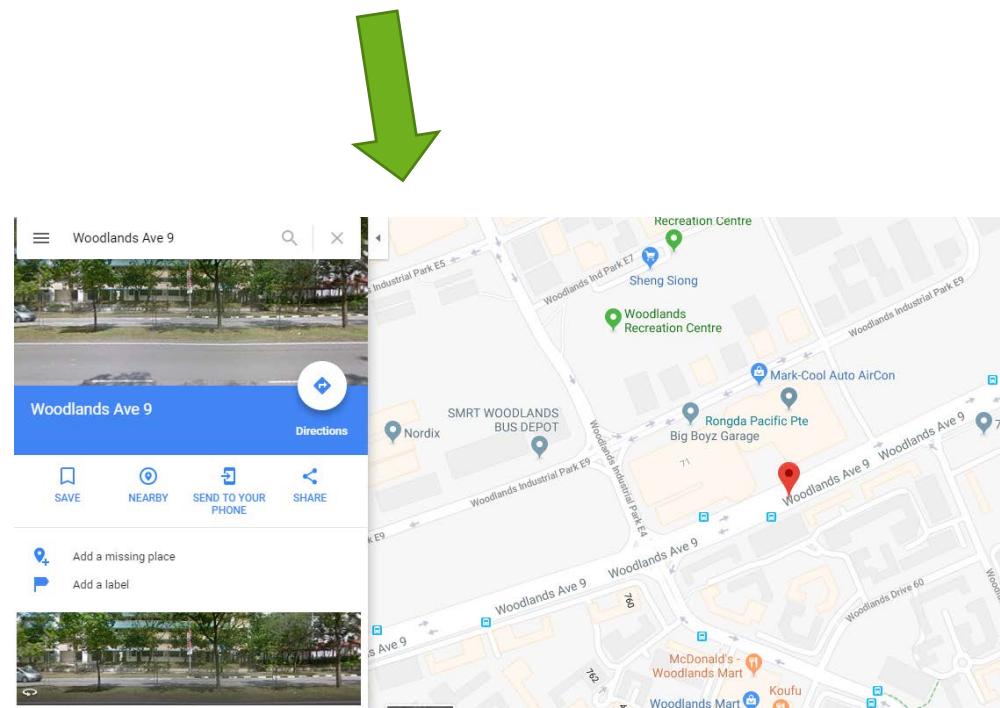


Connecting to the Web

```
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\Pl
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

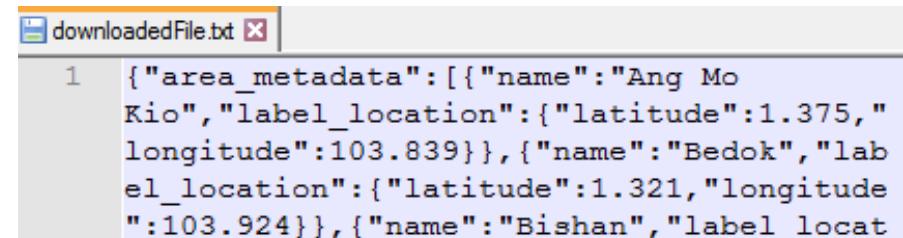
- Use `requests.get()` to get web content from specified URL
- Use `raise_for_status()` to ensure that download is successful before we continue
- Call `open()` with "wb" to create a new file in write binary mode
- Loop over the Response object using `iter_content()`
- Call `write()` on each iteration 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 named "downloadedFile.txt" containing JSON data. The data is an array of objects, each representing a location with its name and coordinates:

```
[{"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": 1.338, "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": 1.338,  
"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)
```

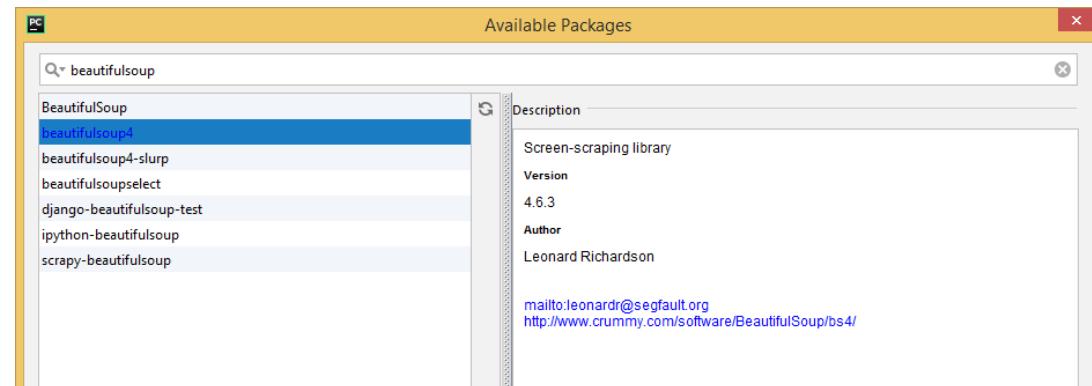


```
C:\Users\denise_quek\AppData\Local\Programs\Python\Py
Ang Mo Kio: Thundery Showers
Bedok: Thundery Showers
Bishan: Heavy Thundery Showers with Gusty Winds
Boon Lay: Heavy Thundery Showers with Gusty Winds
Bukit Batok: Heavy Thundery Showers with Gusty Winds
Bukit Merah: Heavy Thundery Showers with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

Connecting to the Web

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

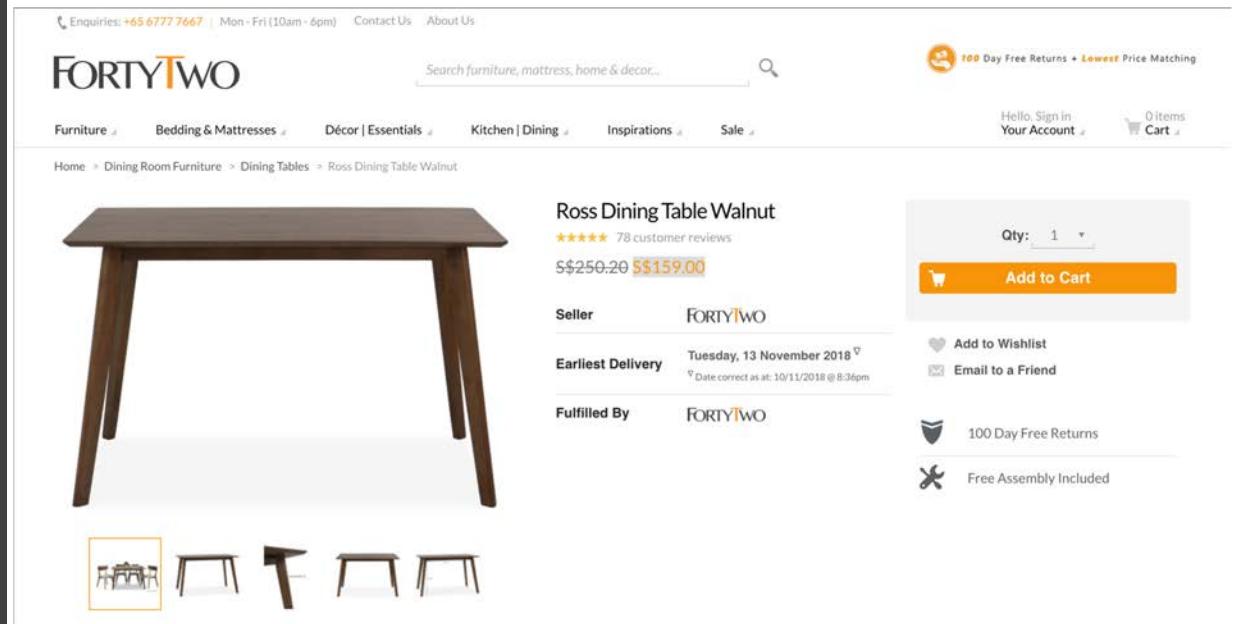
Web Scraping – download and process Web content



Connecting to the Web

- Get the url

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



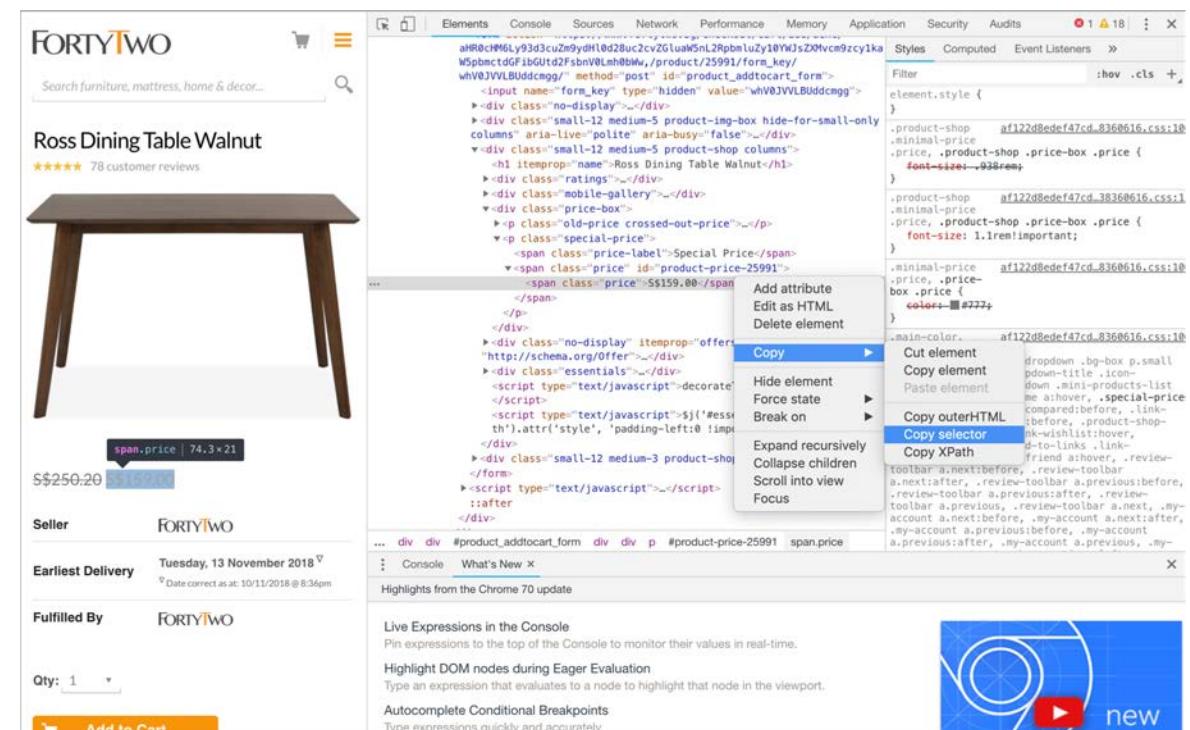
The screenshot shows a product page for the Ross Dining Table Walnut on the FORTYTWO website. At the top, there's a navigation bar with links for Enquiries, Mon - Fri (10am - 6pm), Contact Us, About Us, Furniture, Bedding & Mattresses, Décor | Essentials, Kitchen | Dining, Inspirations, Sale, Hello, Sign in Your Account, and a Cart with 0 items. The main content area features a large image of the dark wood dining table with four tapered legs. Below the image, the product name "Ross Dining Table Walnut" is displayed along with a 4-star rating and 78 customer reviews. The original price of \$250.20 is crossed out, and the discounted price of \$159.00 is shown in orange. The seller is listed as FORTYTWO, and the earliest delivery date is Tuesday, 13 November 2018. The item is fulfilled by FORTYTWO. To the right, there's a "Qty: 1" dropdown and an "Add to Cart" button. Other options like "Add to Wishlist" and "Email to a Friend" are also present. At the bottom, there are smaller images of the table from different angles and icons for 100 Day Free Returns and 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"



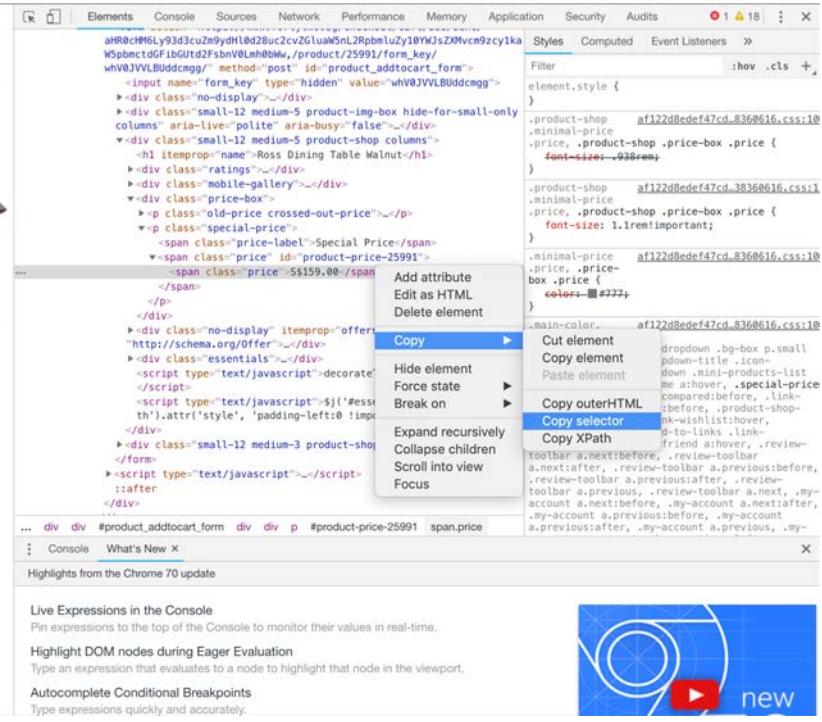
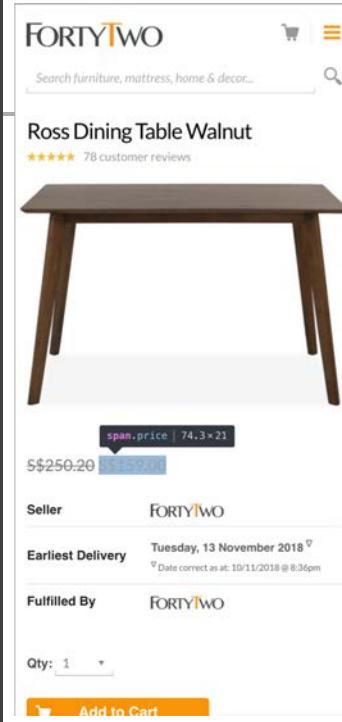
The screenshot shows a web browser displaying a product page for a Ross Dining Table Walnut on the FORTYTWO website. The page features a search bar, product details like '78 customer reviews', and an 'Add to Cart' button. The Chrome DevTools Elements tab is active, focusing on the price element. A context 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)
```

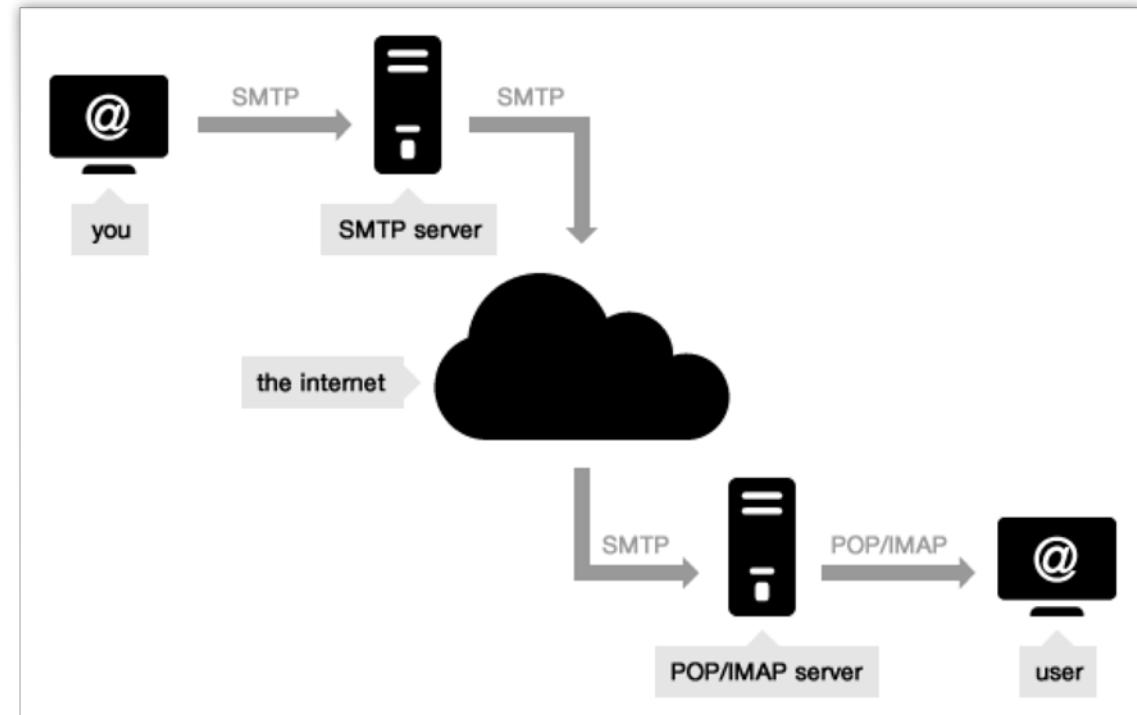


```
C:\Users\kwseow\PycharmProjects\PSA
S$159.00

Process finished with exit code 0
```

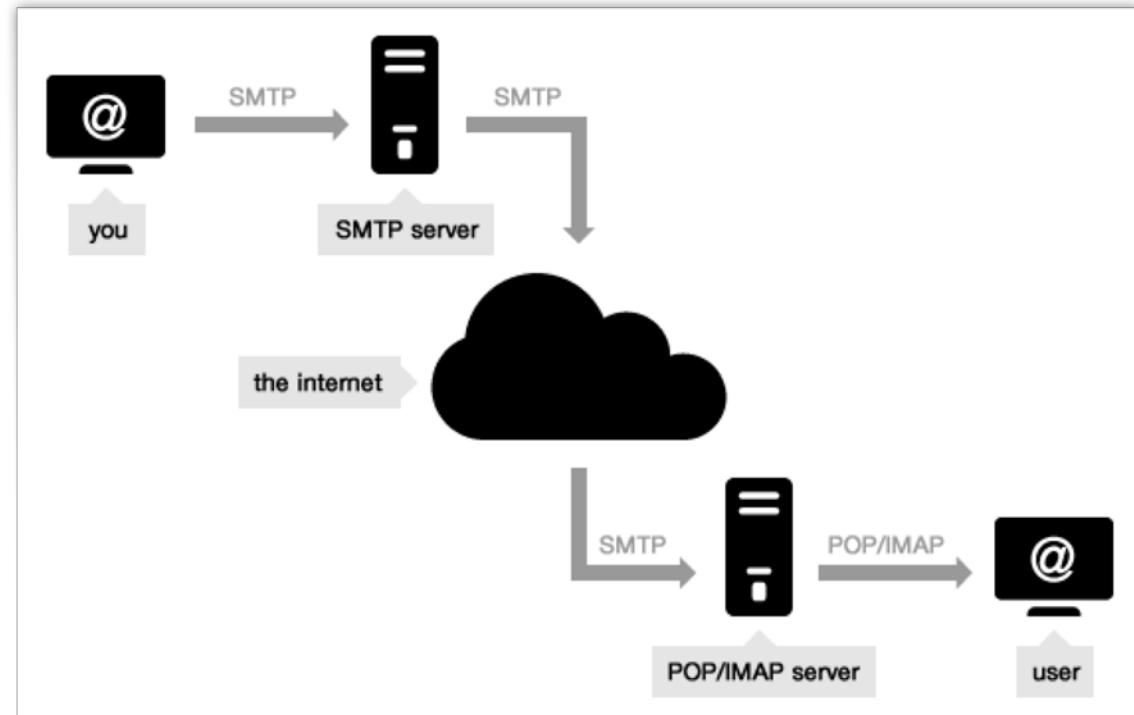
Sending Email

- 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.



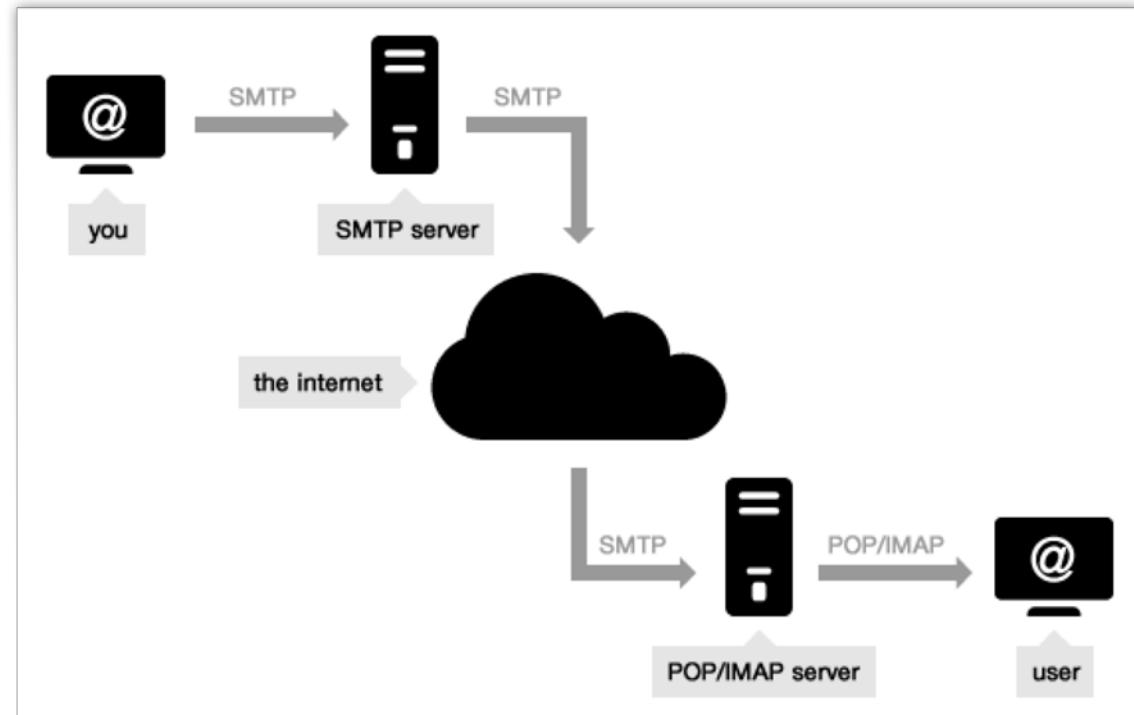
Sending Email

- 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()

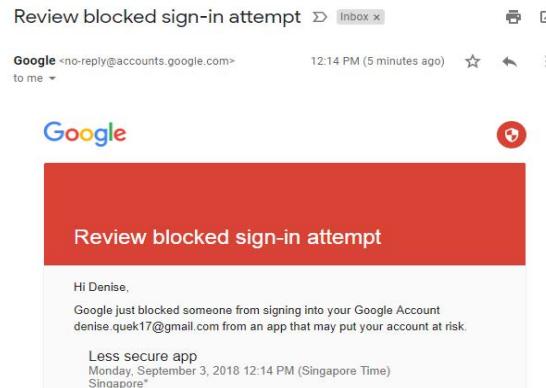
print("Connected. Logging in...")
smtpObj.login(sender_email_address, sender_email_password)

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
```

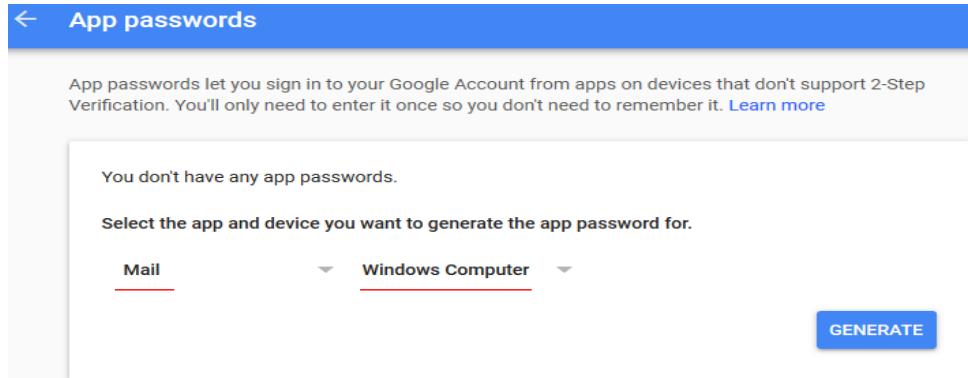
Send Email using Gmail

Steps To Create Google App Password

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

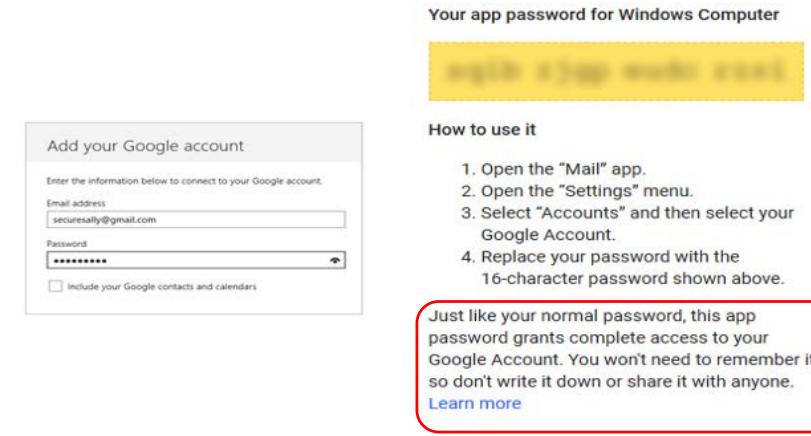
Step 2: Make sure that 2-Step Verification is on

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 under 'App', and 'Windows Computer' is selected under 'Device'. 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 yellowed-out password. Below that, a 'How to use it' section 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. 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

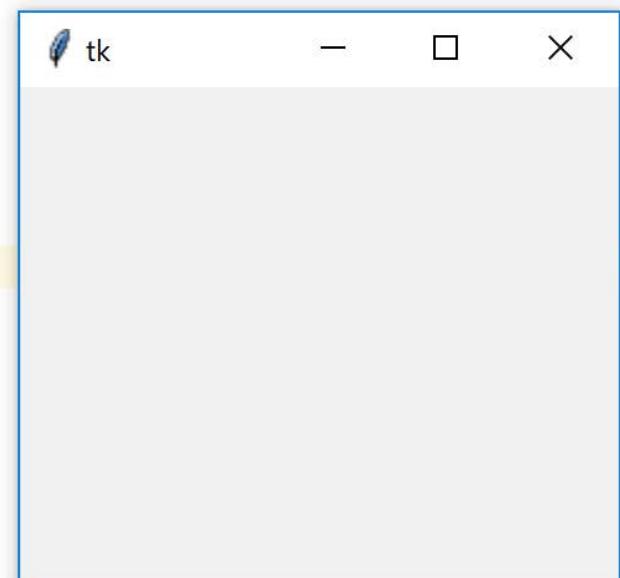
```
16    workbook = openpyxl.load_workbook("D:\\CET_Python\\students_attendance.xlsx")
17    sheet = workbook["Sheet1"]
18
19    max_row = sheet.max_row
20    max_column = sheet.max_column
21
22    for i in range(1, max_row+1):
23
24        attendance = sheet.cell(row=i, column=3).value
25
26        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
```

Graphical User Interface

<https://wiki.python.org/moin/GuiProgramming>

Tkinter – Python's standard GUI library
It is a commonly used GuiProgramming toolkit for Python.

```
1 import tkinter  
2  
3 window = tkinter.Tk()  
4 window.mainloop()  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```



Graphical User Interface

Add a button

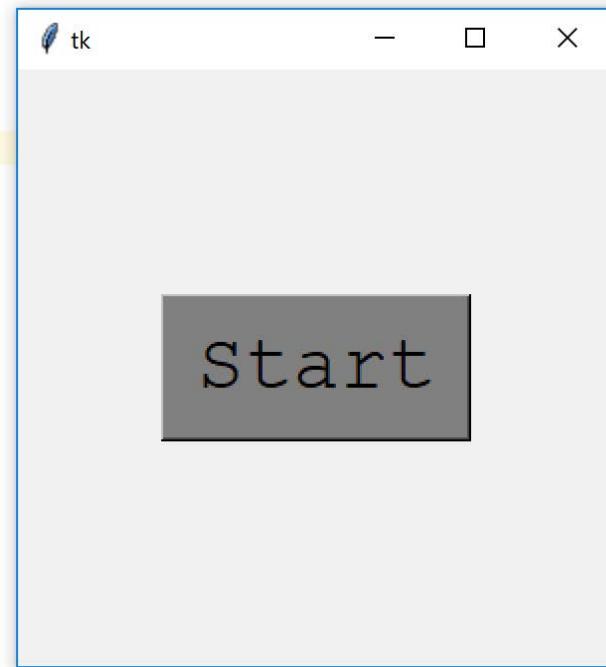
```
1 import tkinter  
2  
3 window = tkinter.Tk()  
4  
5 #Add a button  
6 button = tkinter.Button(window, text="Start")  
7 button.pack()  
8  
9 window.mainloop()  
10  
11
```



Graphical User Interface

- Set the window's size.
- Configure the colour and position of the button.

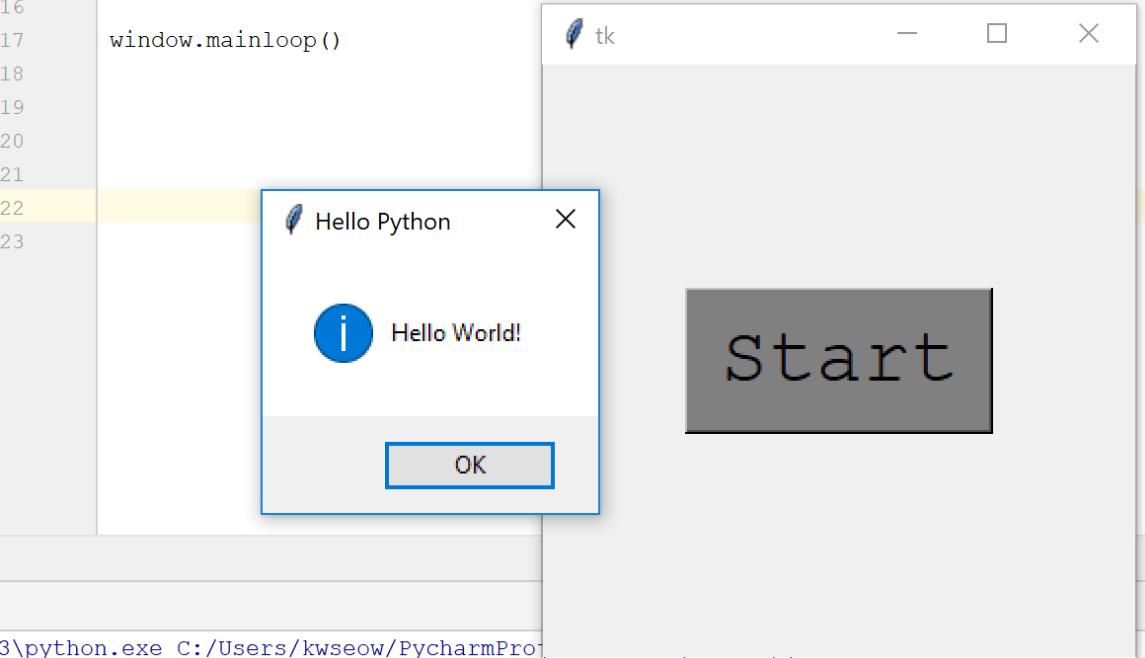
```
1 import tkinter
2
3 window = tkinter.Tk()
4
5 #set the window's size
6 window.geometry("300x300")
7
8 #Add a button
9 button = tkinter.Button(window, text="Start", bg="gray")
10 button.config(font=("Courier", 30))
11 button.pack(side="top", expand=tkinter.YES)
12
13 window.mainloop()
14
15
16
17
```



Graphical User Interface

- Getting the button to response to click.

```
1 import tkinter
2 import tkinter.messagebox
3
4 window = tkinter.Tk()
5
6 #set the window's size
7 window.geometry("300x300")
8
9 def displayMsg():
10     tkinter.messagebox.showinfo("Hello Python", "Hello World!")
11
12 #Add a button
13 button = tkinter.Button(window, text="Start", bg="gray", command=displayMsg)
14 button.config(font=("Courier", 30))
15 button.pack(side="top", expand=tkinter.YES)
16
17 window.mainloop()
```



3\python.exe C:/Users/kwseow/PycharmProj

py2EXE

py2exe turns Python programs into packages that can be run on other Windows computers without needing to install Python on those computers. It's included with your Python distribution.

Install py2exe via pip

Note: py2exe will not run in virtual environment

```
from distutils.core import setup
import py2exe
setup(console=['hello.py']) ← Save as setup.py
```

Run python setup.py py2exe and you have created a standalone executable

More at

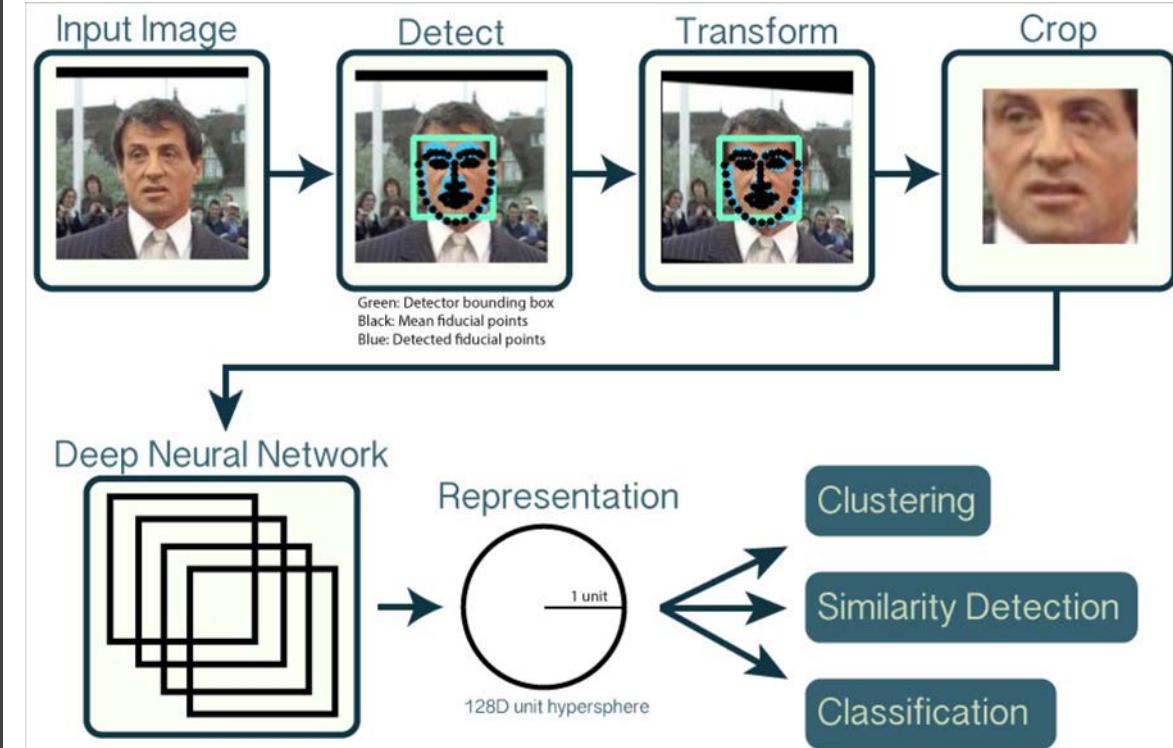
<http://www.py2exe.org/index.cgi/Tutorial#Step1>

```
C:\CET>python setup.py py2exe
```

Deep Learning Facial Recognition

We will:

- Detect faces
- Compute 128-d face embeddings to quantify a face
- Train a Support Vector Machine (SVM) on top of the embeddings
- Recognize faces in images and video streams



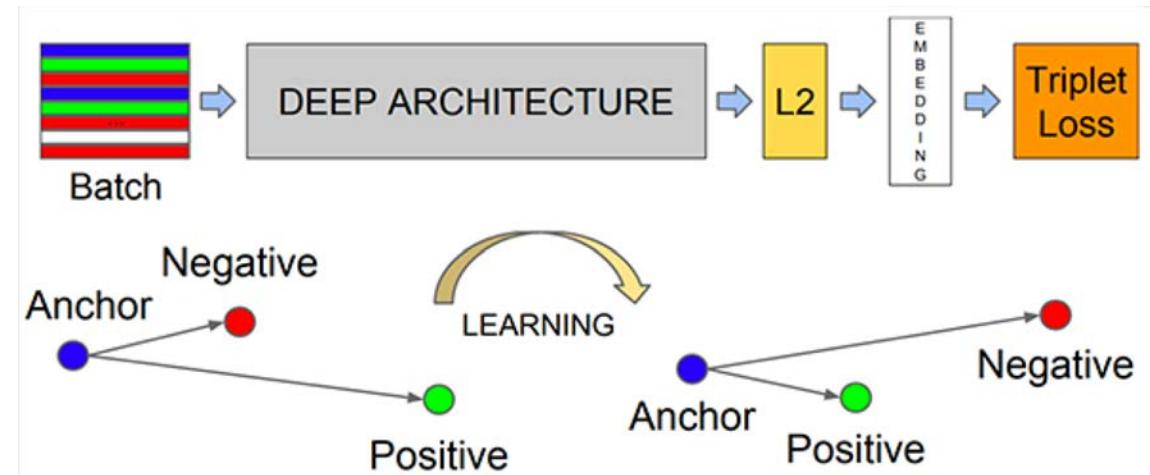
Deep Learning Facial Recognition

- In order to build our OpenCV face recognition pipeline, we'll be applying deep learning in two key steps:
- To apply *face detection*, which detects the *presence* and location of a face in an image, but does not identify it
- To extract the 128-d feature vectors (called "embeddings") that *quantify* each face in an image

Create a new virtual environment:

```
conda create --name cv2_py37 --
yes python=3.7 opencv scikit-learn
```

```
pip install imutils
```



The model responsible for actually quantifying each face in an image is from the [OpenFace project](#), a Python and Torch implementation of face recognition with deep learning. This implementation comes from Schroff et al.'s 2015 CVPR publication, [*FaceNet: A Unified Embedding for Face Recognition and Clustering*](#).

Deep Learning Facial Recognition

- 3 Step Process

01

Step #1

Extract embeddings from face dataset

02

Step #2

Train face recognition mode;

03

Step #3

Recognise faces with OpenCV

Download source code at:
<http://bit.ly/2Qr68DW>

Deep Learning Facial Recognition

- Step 1 – Extract embeddings from face dataset

```
OpenCV Face Recognition Python
61 # construct a blob from the image
62 imageBlob = cv2.dnn.blobFromImage(
63     cv2.resize(image, (300, 300)), 1.0, (300, 300),
64     (104.0, 177.0, 123.0), swapRB=False, crop=False)
65
66 # apply OpenCV's deep learning-based face detector to localize
67 # faces in the input image
68 detector.setInput(imageBlob)
69 detections = detector.forward()
```

```
OpenCV Face Recognition Python
71 # ensure at least one face was found
72 if len(detections) > 0:
73     # we're making the assumption that each image has only ONE
74     # face, so find the bounding box with the largest probability
75     i = np.argmax(detections[0, 0, :, 2])
76     confidence = detections[0, 0, i, 2]
77
78     # ensure that the detection with the largest probability also
79     # means our minimum probability test (thus helping filter out
80     # weak detections)
81     if confidence > args["confidence"]:
82         # compute the (x, y)-coordinates of the bounding box for
83         # the face
84         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
85         (startX, startY, endX, endY) = box.astype("int")
86
87         # extract the face ROI and grab the ROI dimensions
88         face = image[startY:endY, startX:endX]
89         (fH, fW) = face.shape[:2]
90
91         # ensure the face width and height are sufficiently large
92         if fW < 20 or fH < 20:
93             continue
```

Deep Learning Facial Recognition

- Step 1 – Extract embeddings from face dataset
- Open a command window
- cd PSA\opencv-face-recognition
- python extract_embeddings.py --dataset dataset --embeddings output/embeddings.pickle --detector face_detection_model --embedding-model openface_nn4.small2.v1.t7

```
OpenCV Face Recognition Python
95      # construct a blob for the face ROI, then pass the blob
96      # through our face embedding model to obtain the 128-d
97      # quantification of the face
98      faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
99          (96, 96), (0, 0, 0), swapRB=True, crop=False)
100     embedder.setInput(faceBlob)
101     vec = embedder.forward()
102
103     # add the name of the person + corresponding face
104     # embedding to their respective lists
105     knownNames.append(name)
106     knownEmbeddings.append(vec.flatten())
107     total += 1
```

```
OpenCV Face Recognition Python
109    # dump the facial embeddings + names to disk
110    print("[INFO] serializing {} encodings...".format(total))
111    data = {"embeddings": knownEmbeddings, "names": knownNames}
112    f = open(args["embeddings"], "wb")
113    f.write(pickle.dumps(data))
114    f.close()
```

Deep Learning Facial Recognition

- Step 2 – Train face recognition model
- python train_model.py --embeddings output/embeddings.pickle --recognizer output/recognizer.pickle --le output/le.pickle

OpenCV Face Recognition

```

17 # load the face embeddings
18 print("[INFO] loading face embeddings...")
19 data = pickle.loads(open(args["embeddings"], "rb").read())
20
21 # encode the labels
22 print("[INFO] encoding labels...")
23 le = LabelEncoder()
24 labels = le.fit_transform(data["names"])

```

OpenCV Face Recognition

```

26 # train the model used to accept the 128-d embeddings of the face and
27 # then produce the actual face recognition
28 print("[INFO] training model...")
29 recognizer = SVC(C=1.0, kernel="linear", probability=True)
30 recognizer.fit(data["embeddings"], labels)

```

OpenCV Face Recognition

```

32 # write the actual face recognition model to disk
33 f = open(args["recognizer"], "wb")
34 f.write(pickle.dumps(recognizer))
35 f.close()
36
37 # write the label encoder to disk
38 f = open(args["le"], "wb")
39 f.write(pickle.dumps(le))
40 f.close()

```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV
- python recognize.py --detector face_detection_model --embedding-model openface_nn4.small2.v1.t7 --recognizer output/recognizer.pickle --le output/le.pickle --image images/adrian.jpg
- python recognize_video.py --detector face_detection_model --embedding-model openface_nn4.small2.v1.t7 --recognizer output/recognizer.pickle --le output/le.pickle

```
OpenCV Face Recognition Python
25 # load our serialized face detector from disk
26 print("[INFO] loading face detector...")
27 protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
28 modelPath = os.path.sep.join([args["detector"],
29     "res10_300x300_ssd_iter_140000.caffemodel"])
30 detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
31
32 # load our serialized face embedding model from disk
33 print("[INFO] loading face recognizer...")
34 embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])
35
36 # load the actual face recognition model along with the label encoder
37 recognizer = pickle.loads(open(args["recognizer"], "rb").read())
38 le = pickle.loads(open(args["le"], "rb").read())
```

```
OpenCV Face Recognition Python
40 # load the image, resize it to have a width of 600 pixels (while
41 # maintaining the aspect ratio), and then grab the image dimensions
42 image = cv2.imread(args["image"])
43 image = imutils.resize(image, width=600)
44 (h, w) = image.shape[:2]
45
46 # construct a blob from the image
47 imageBlob = cv2.dnn.blobFromImage(
48     cv2.resize(image, (300, 300)), 1.0, (300, 300),
49     (104.0, 177.0, 123.0), swapRB=False, crop=False)
50
51 # apply OpenCV's deep learning-based face detector to localize
52 # faces in the input image
53 detector.setInput(imageBlob)
54 detections = detector.forward()
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

```
OpenCV Face Recognition Python
56 # loop over the detections
57 for i in range(0, detections.shape[2]):
58     # extract the confidence (i.e., probability) associated with the
59     # prediction
60     confidence = detections[0, 0, i, 2]
61
62     # filter out weak detections
63     if confidence > args["confidence"]:
64         # compute the (x, y)-coordinates of the bounding box for the
65         # face
66         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
67         (startX, startY, endX, endY) = box.astype("int")
68
69         # extract the face ROI
70         face = image[startY:endY, startX:endX]
71         (fH, fW) = face.shape[:2]
72
73         # ensure the face width and height are sufficiently large
74         if fW < 20 or fH < 20:
75             continue
```

```
OpenCV Face Recognition Python
77     # construct a blob for the face ROI, then pass the blob
78     # through our face embedding model to obtain the 128-d
79     # quantification of the face
80     faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
81                                     (0, 0, 0), swapRB=True, crop=False)
82     embedder.setInput(faceBlob)
83     vec = embedder.forward()
84
85     # perform classification to recognize the face
86     preds = recognizer.predict_proba(vec)[0]
87     j = np.argmax(preds)
88     proba = preds[j]
89     name = le.classes_[j]
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

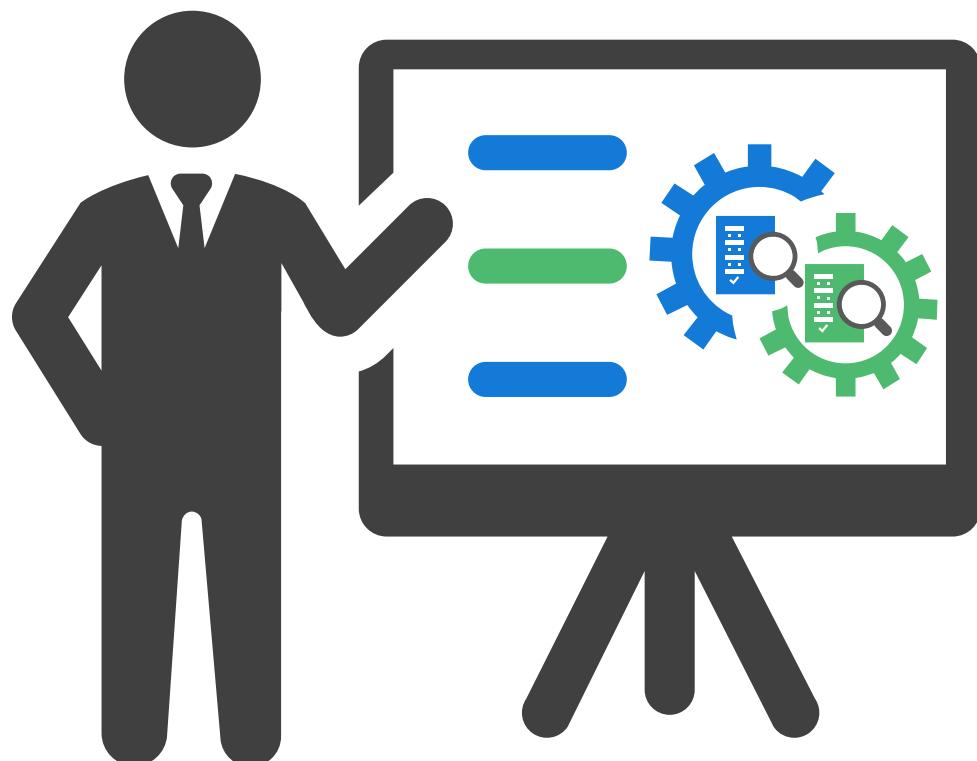
```
OpenCV Face Recognition Python
56 # loop over the detections
57 for i in range(0, detections.shape[2]):
58     # extract the confidence (i.e., probability) associated with the
59     # prediction
60     confidence = detections[0, 0, i, 2]
61
62     # filter out weak detections
63     if confidence > args["confidence"]:
64         # compute the (x, y)-coordinates of the bounding box for the
65         # face
66         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
67         (startX, startY, endX, endY) = box.astype("int")
68
69         # extract the face ROI
70         face = image[startY:endY, startX:endX]
71         (fH, fW) = face.shape[:2]
72
73         # ensure the face width and height are sufficiently large
74         if fW < 20 or fH < 20:
75             continue
```

```
OpenCV Face Recognition Python
77     # construct a blob for the face ROI, then pass the blob
78     # through our face embedding model to obtain the 128-d
79     # quantification of the face
80     faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
81                                     (0, 0, 0), swapRB=True, crop=False)
82     embedder.setInput(faceBlob)
83     vec = embedder.forward()
84
85     # perform classification to recognize the face
86     preds = recognizer.predict_proba(vec)[0]
87     j = np.argmax(preds)
88     proba = preds[j]
89     name = le.classes_[j]
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

```
OpenCV Face Recognition Python
91     # draw the bounding box of the face along with the associated
92     # probability
93     text = "{}: {:.2f}%".format(name, proba * 100)
94     y = startY - 10 if startY - 10 > 10 else startY + 10
95     cv2.rectangle(image, (startX, startY), (endX, endY),
96                   (0, 0, 255), 2)
97     cv2.putText(image, text, (startX, y),
98                 cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
99
100    # show the output image
101    cv2.imshow("Image", image)
102    cv2.waitKey(0)
```



Day 2 Summary

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

- ✓ *Image Processing: loading, scaling, watermark, applying filters*
- ✓ *Connecting to the web, scrapping*
- ✓ *Sending emails*
- ✓ *Graphical User Interface*

- ✓ *AI – Facial Recognition*

- ✓ *Use cases*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: <http://bit.ly/2B2zu2M>

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

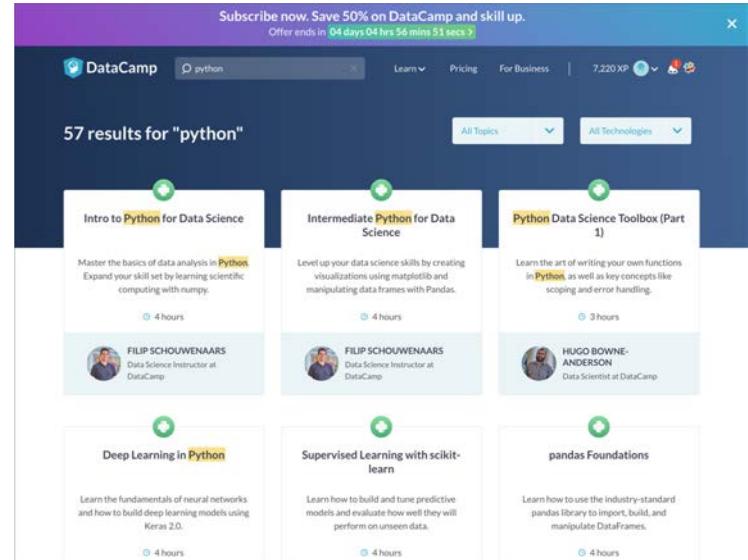
<https://www.datacamp.com/>

Edx

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

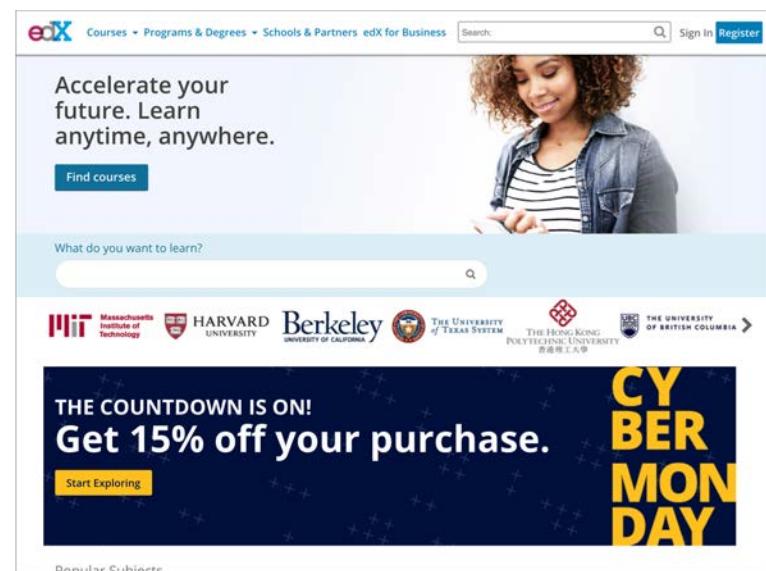
Udemy (freemium course)

<https://t.me/freecourse>



The screenshot shows a DataCamp search results page for "python". At the top, there is a purple banner with the text "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 main heading says "57 results for 'python'". There are six course cards displayed in two rows of three:

- 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, edX for Business, and a search bar. A banner features a woman smiling and the text "Accelerate your future. Learn anytime, anywhere." with a "Find courses" button. 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 universities are displayed at the bottom: MIT, HARVARD, Berkeley, THE UNIVERSITY OF TEXAS SYSTEM, THE HONG KONG POLYTECHNIC UNIVERSITY, and THE UNIVERSITY OF BRITISH COLUMBIA.

Online Evaluation
Fundamentals of Python for Beginners
(26-27 November 2018)



<https://tinyurl.com/yb3u4tnv>

Thank you