PSA UNIVERSITY

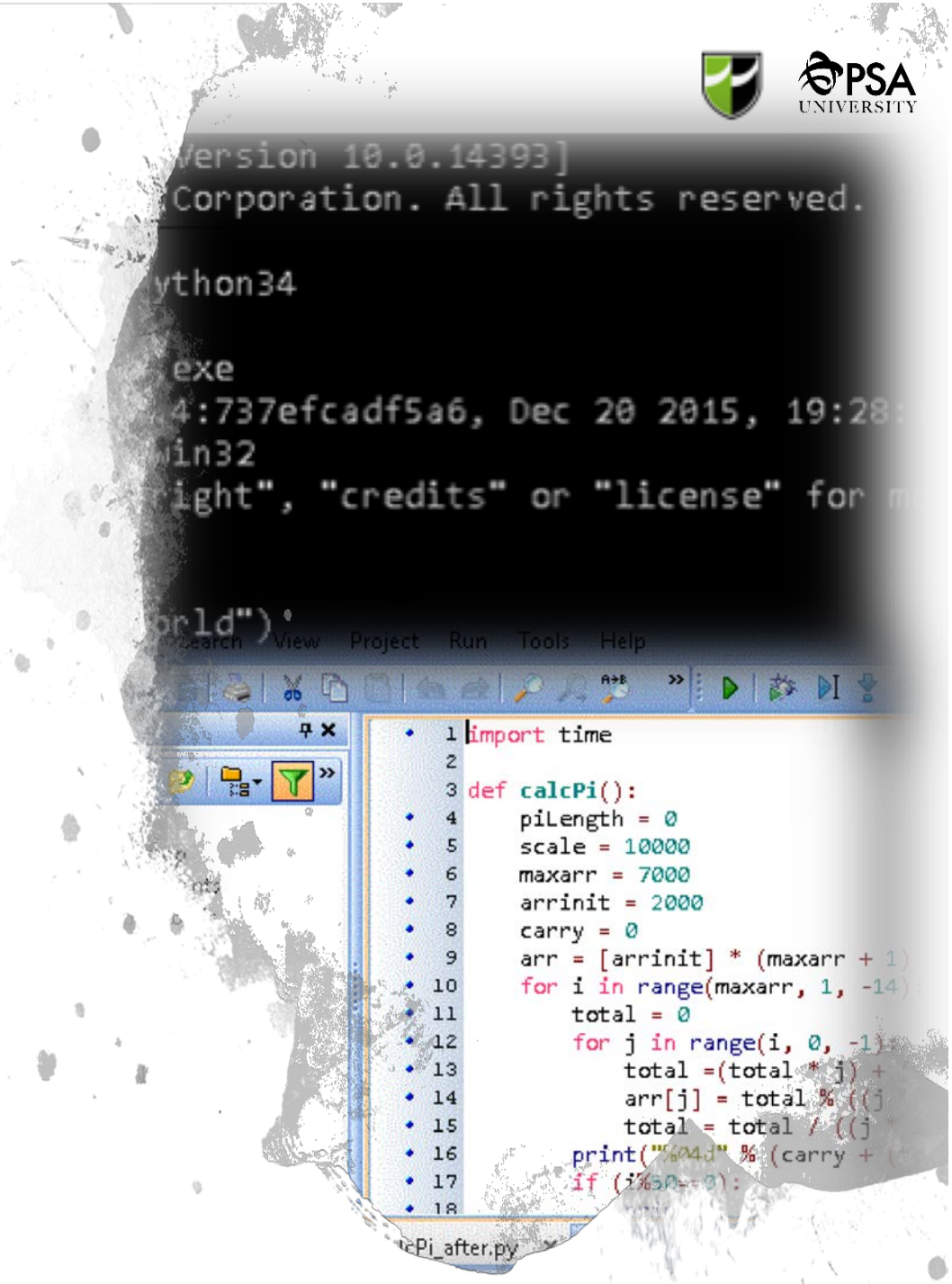
REPUBLIC POLYTECHNIC
DISCOVER. TRANSFORM. ACHIEVE

Introductory Programming in Python

Day 2

Programme
Day One

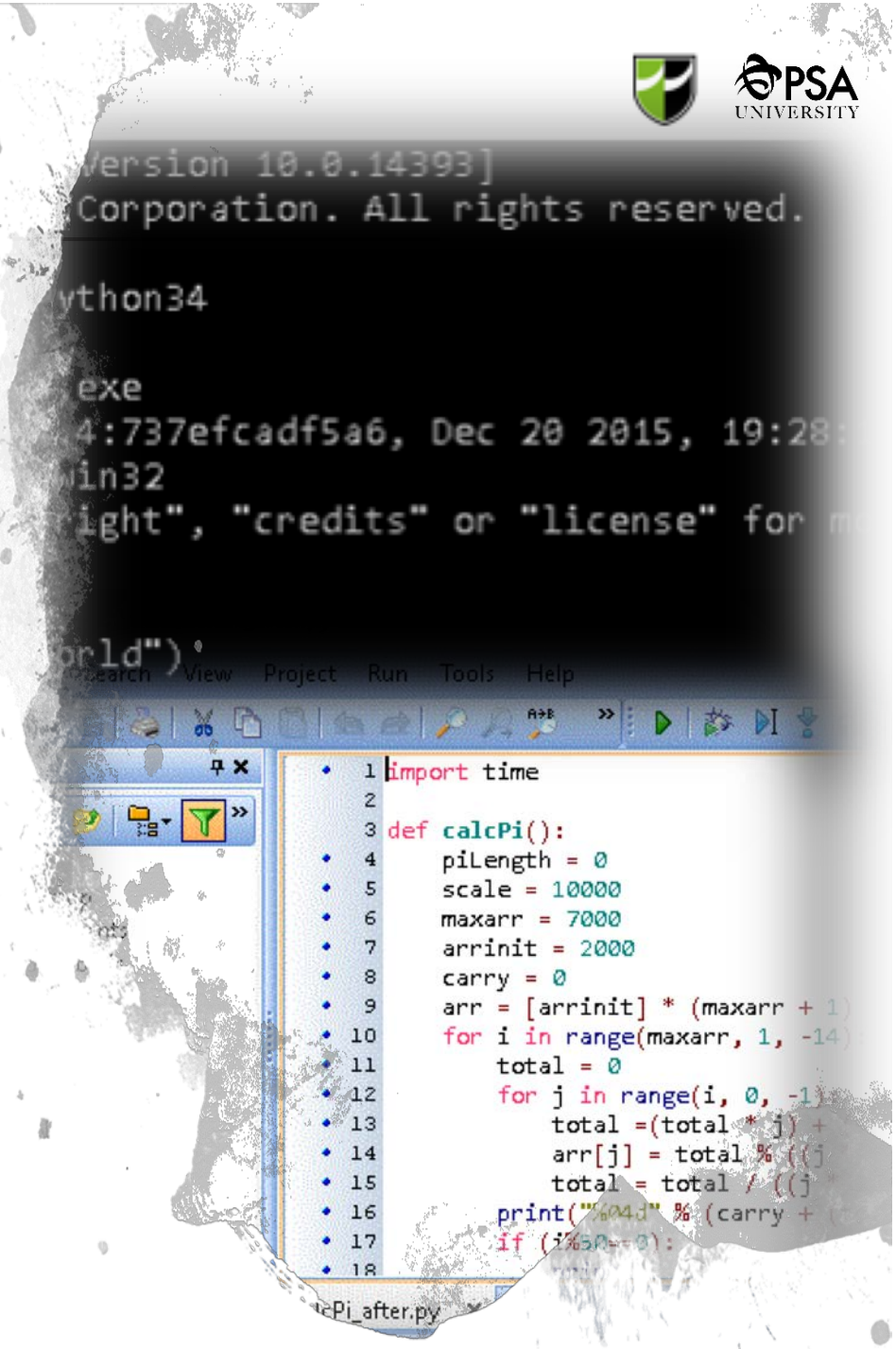
- A brief history of Python
- Setting up python environment
- Learn the basics:
  - Data types
  - Conversions
  - String operations
  - Functions
  - And more!
- String functions, formatting
- Find files by name, by extension, by size, by content and calculate the total size



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



Read files

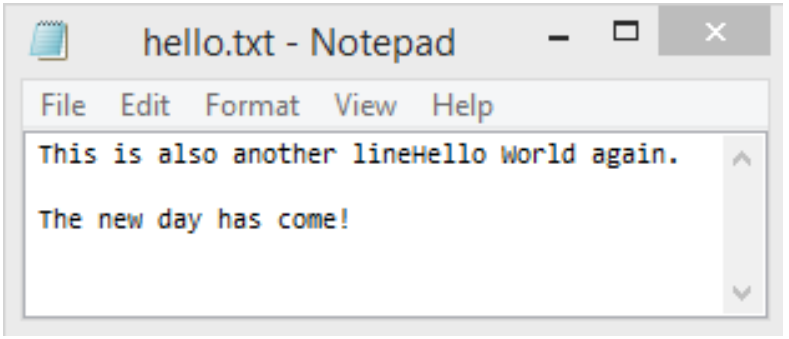
- Open() will return a file object which has reading and writing related methods
- Pass ‘r’ (or nothing) to open() to open the file in read mode.
- Call read() to read the contents of a file
- Call readlines() to return a list of strings of the file’s content.
- Call close() when you are done with the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

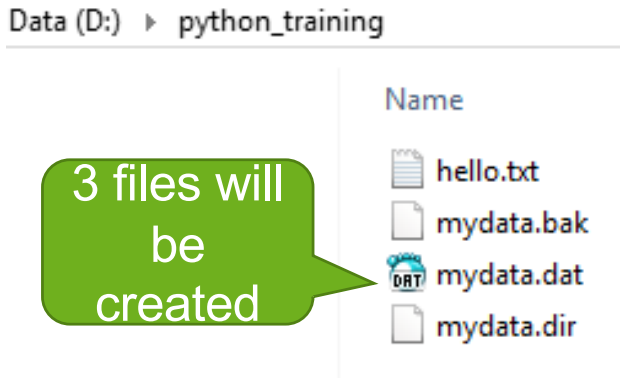
```
>>> helloFile = open("D:\\python_training\\hello.txt", "w")
>>> helloFile.write("This is also another line")
25
>>> helloFile.close()
>>> helloFile = open("D:\\python_training\\hello.txt")
>>> helloFile.read()
'This is also another line'

>>> 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()
```



https://docs.python.org/3/library/shelve.html#

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'
```

https://docs.python.org/3/library/shutil.html?highlight=shutil

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

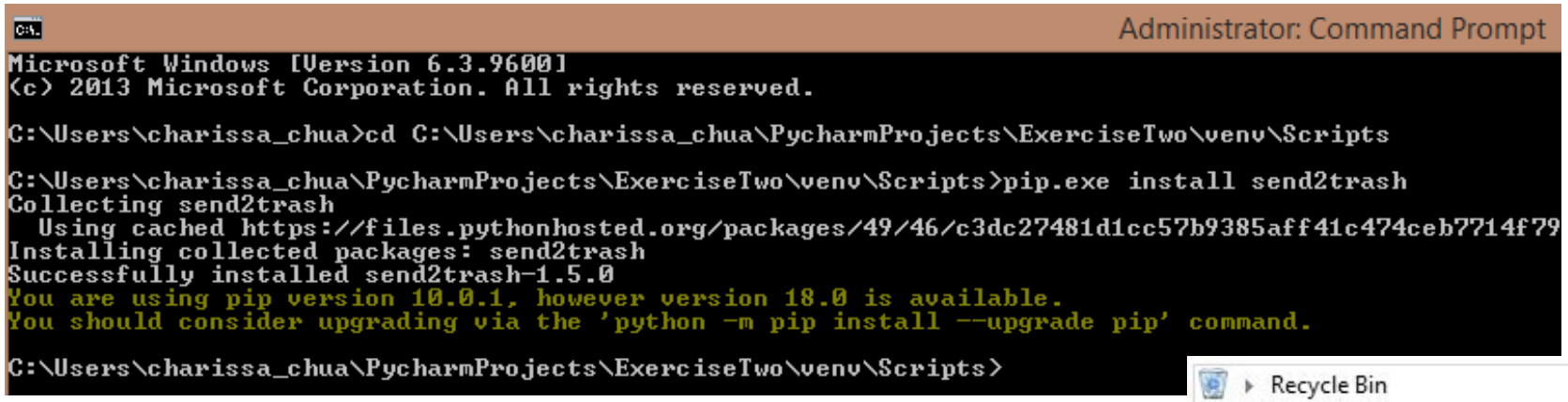
```
>>> import os
>>> os.getcwd()
'C:\\Users\\charissa_chua\\PycharmProjects\\ExerciseOne'
>>> os.rmdir("D:\\folder2_backup")
Traceback (most recent call last):
  File "<input>", line 1, in <module>
OSError: [WinError 145] The directory is not empty: 'D:\\folder2_backup'
>>> import shutil
>>> shutil.rmtree("D:\\folder2_backup")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

RemoveFiles.py

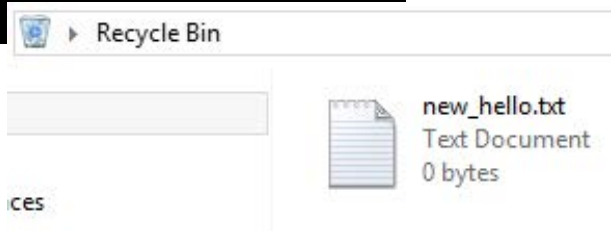
```
1  import os
2
3  os.chdir("C:\\Users\\charissa_chua\\Downloads")
4
5  for filename in os.listdir():
6      if filename.endswith(".docx"):
7          #os.unlink(filename)
8          print(filename)
```

send2trash module

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




```
>>> import send2trash
>>> send2trash.send2trash("D:\\folder2\\new_hello.txt")
```



https://pypi.org/project/Send2Trash/

Walk a Directory

FileUtils.py

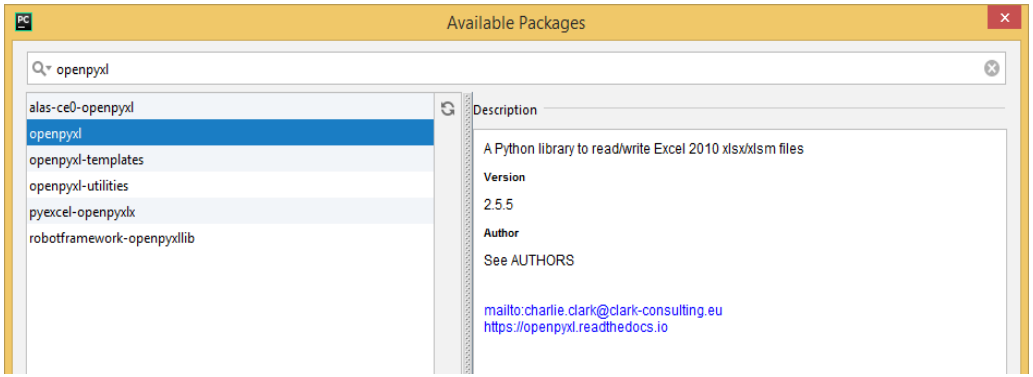
```
1   import os
2   import shutil
3
4   for folderName, subfolders, filenames in os.walk("C:\\Users\\charissa_chua\\Downloads"):
5       print("The folder is " + folderName)
6       print("The subfolders in " + folderName + " are: " + str(subfolders))
7       print("The filenames in " + folderName + " 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**	**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			

Sheet1

https://openpyxl.readthedocs.io/en/stable/

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

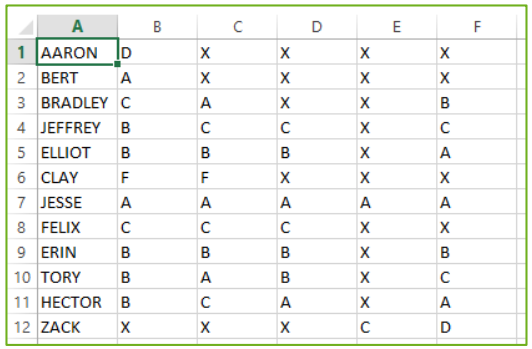
```python
#! python3

import openpyxl

workbook = openpyxl.load_workbook("D:\CET_Python\students_attendance.xlsx")
sheet = workbook["Sheet1"]

max_row = sheet.max_row
max_column = sheet.max_column

for i in range(1, max_row+1):

    attendance = sheet.cell(row=i, column=3).value

    if attendance == "Absent":
        name = sheet.cell(row=i, column=1).value
        email = sheet.cell(row=i, column=2).value
        print(name + " is absent.")

```

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?

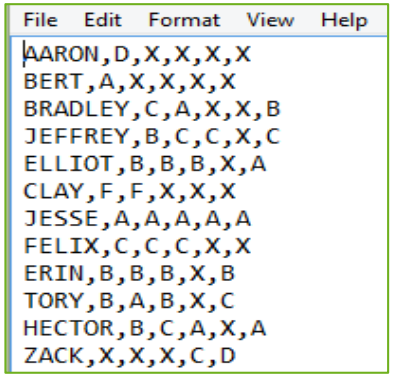
- The same data when viewed with Excel …



	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) …



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()
```

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', 'F', 'F', 'X', 'X', 'X']
```

5) Close the file (remove the link)

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)

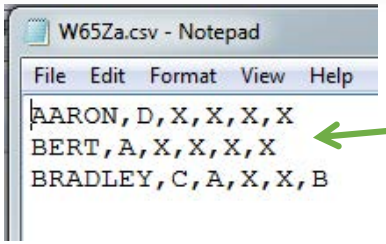


Image processing

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

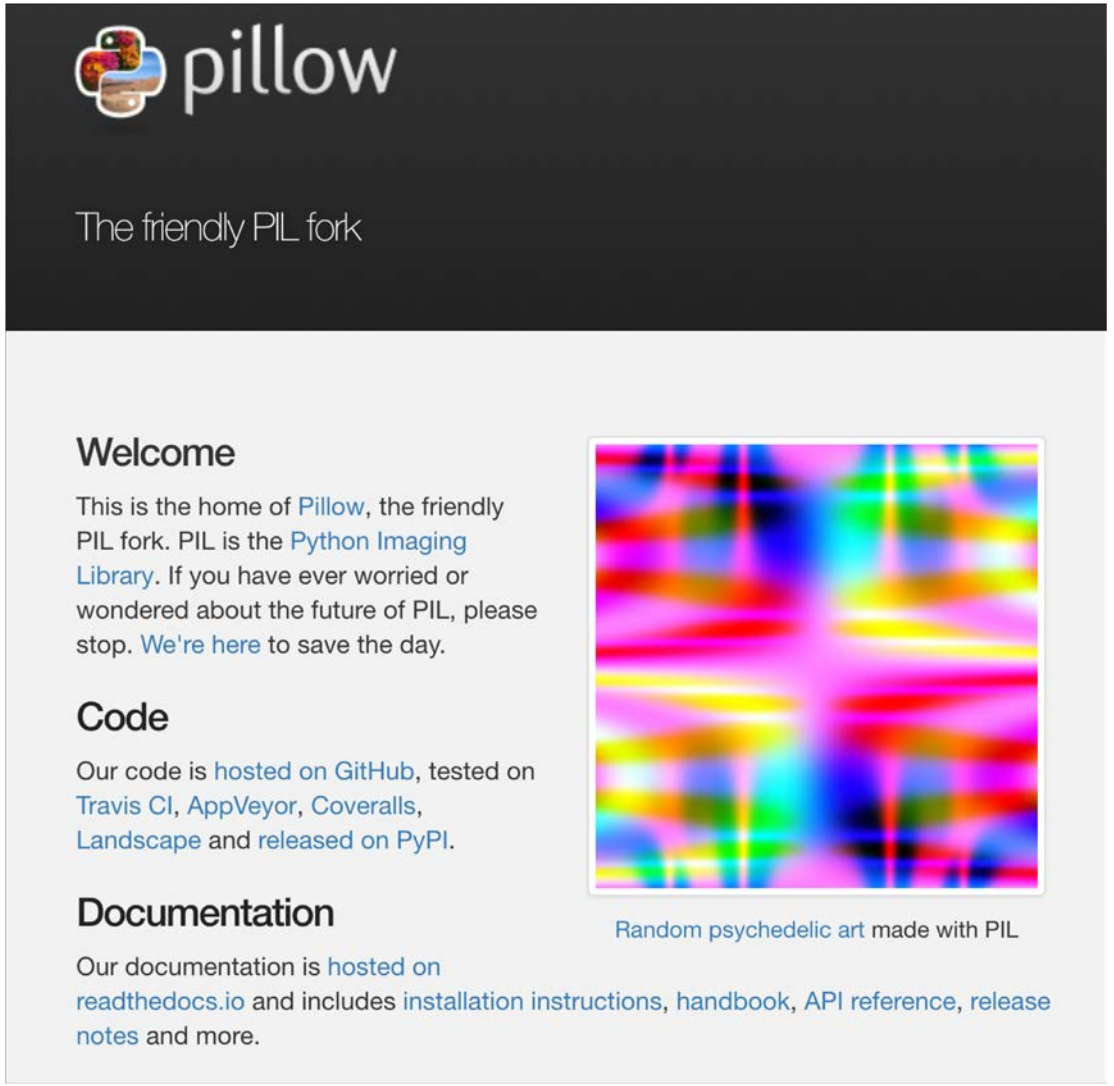


Image processing

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")
```

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.

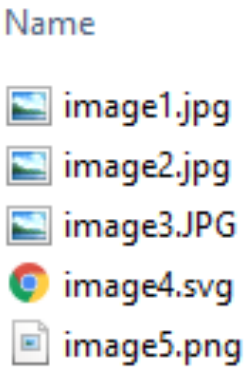
Image processing

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(...)




```
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\Proc
 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("%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 - 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.

```
import os
from PIL import Image, ImageFilter

where = ".\\Day2Resource\\img\\"
outFolder = ".\\Day2Resource\\img\\out\\"

def processAllImages(onlyFirst):
    c = 1
    for root, dirs, files in os.walk(where):
        for file in files:
            fullname = os.path.join(root, file)
            if (file.lower().endswith("jpg") or \
                    file.lower().endswith("bmp") or \
                    file.lower().endswith("png") or \
                    file.lower().endswith("svg")):
                im = Image.open(fullname)
                outFilename = os.path.join(outFolder, file)
                print("%2d %s size:%s (%s)" \
                      % (c, fullname, im.size, im.mode))

                out = im.filter(ImageFilter.BLUR)
                im.show()
                out.show()
                out.save(outFilename)
                c += 1
                if (onlyFirst):
                    return

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 holday.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 outFileName 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.splitext(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

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

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

In this case we can use the \\Day2Resource\\watermark.png
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.

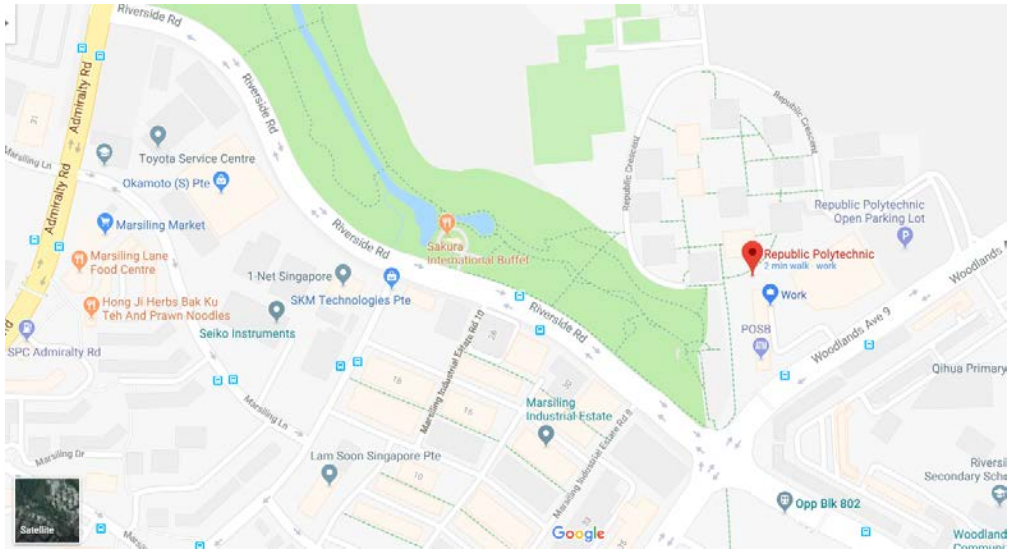
```
1   from PIL import Image
2
3   def watermark(im, mark, position):
4       layer = Image.new("RGBA", im.size, (0,0,0,0))
5       layer.paste(mark, position)
6       return Image.composite(layer, im, layer)
7
8
9   im = Image.open("Day2Resource\\img\\clungup.jpg")
10  mark = Image.open("Day2Resource\\watermark.png")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

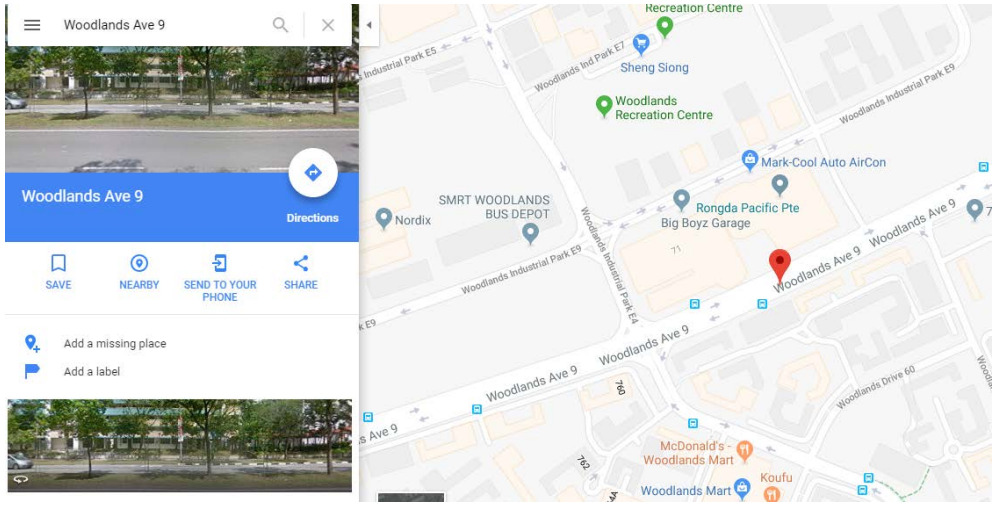


Connecting to the Web

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  address = input('Enter Address: ')
7  webbrowser.open(url + "place/" + address)
```

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



Connecting to the Web

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

Get the required information from the given URL

```
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)
```

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
```

downloadedFile.txt

```
1   {"area_metadata":[{"name":"Ang Mo
    Kio","label_location":{"latitude":1.375,"
    longitude":103.839}},{"name":"Bedok","lab
    el_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",
```


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

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

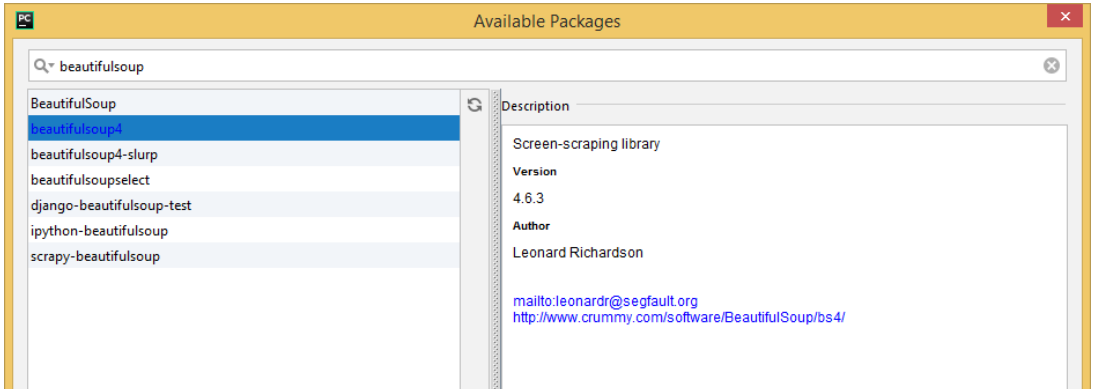
```python
import json
import requests

url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
req = requests.get(url)

data = json.loads(req.text)

forecasts = data["items"][0]["forecasts"]

for forecast in forecasts:
    area = forecast["area"]
    weather = forecast["forecast"]
    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

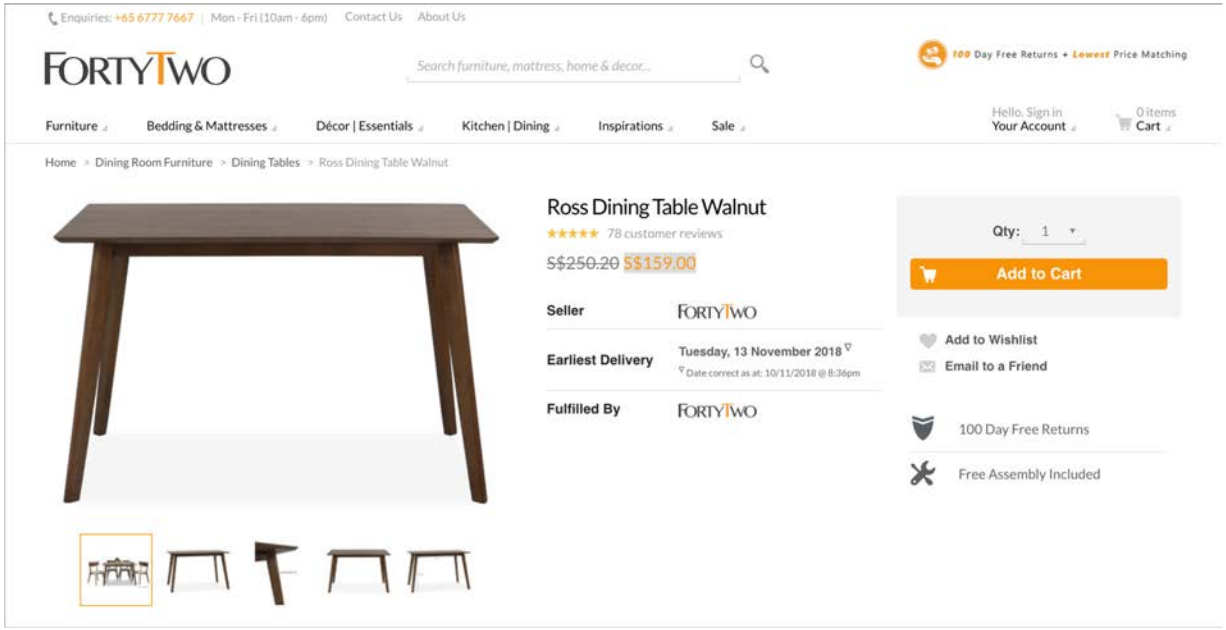
Web Scraping – download and process Web content



Connecting to the Web

- Get the url

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

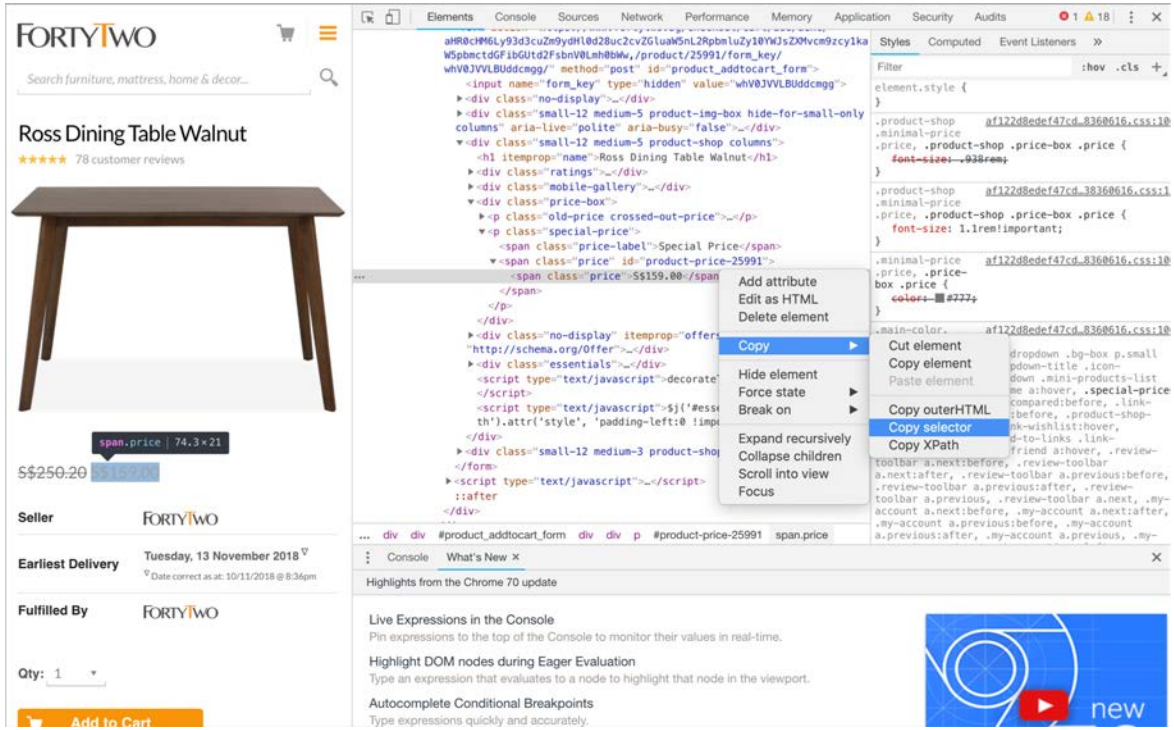


Connecting to the Web

- Get the url

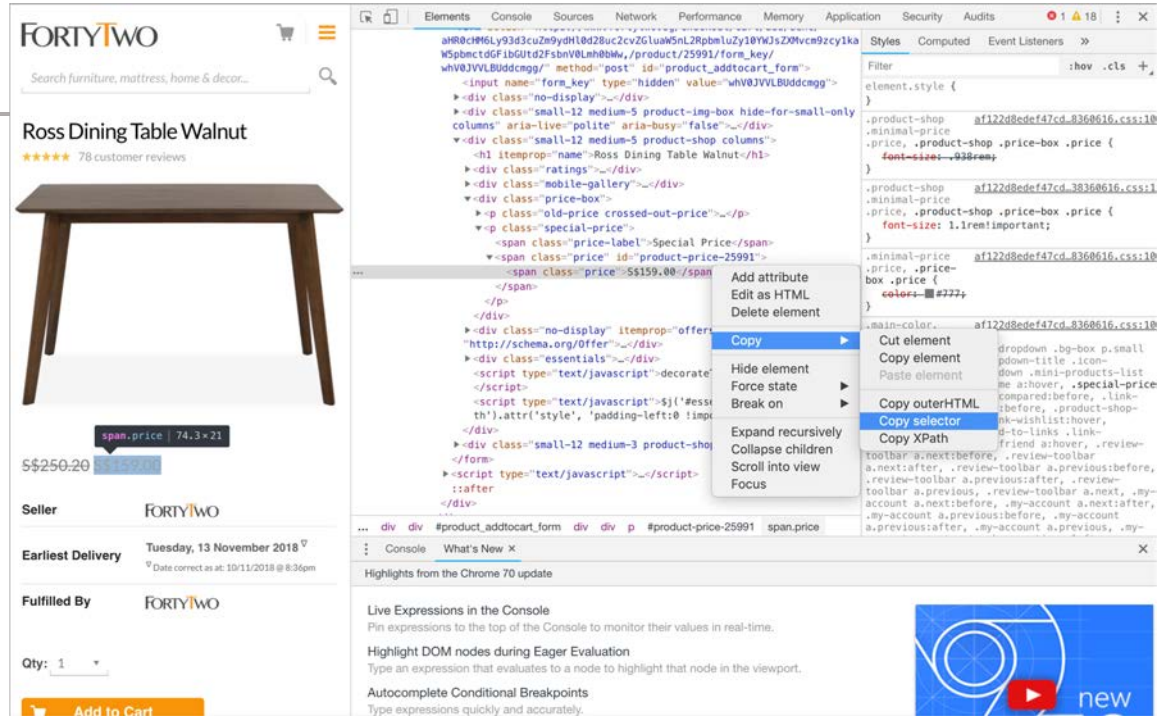
https://www.fortytwo.sg/dining/dining-tables/ross-dining-table-walnut.html

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



Connecting to the Web

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

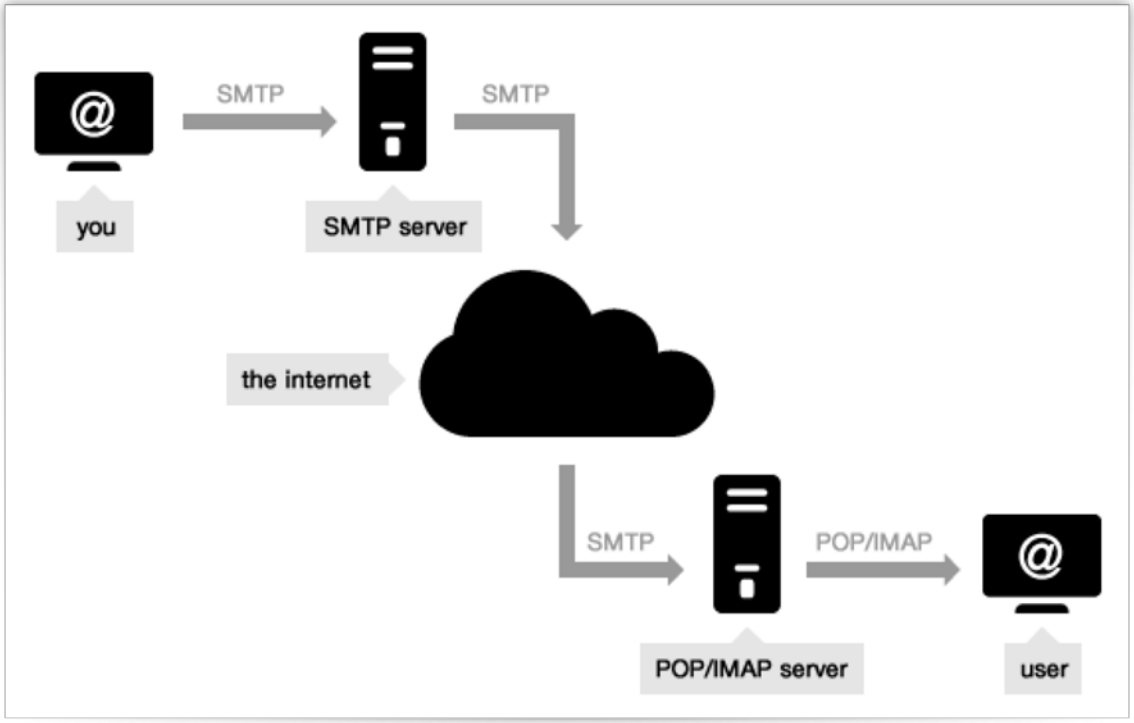



```
import bs4
import requests

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
```

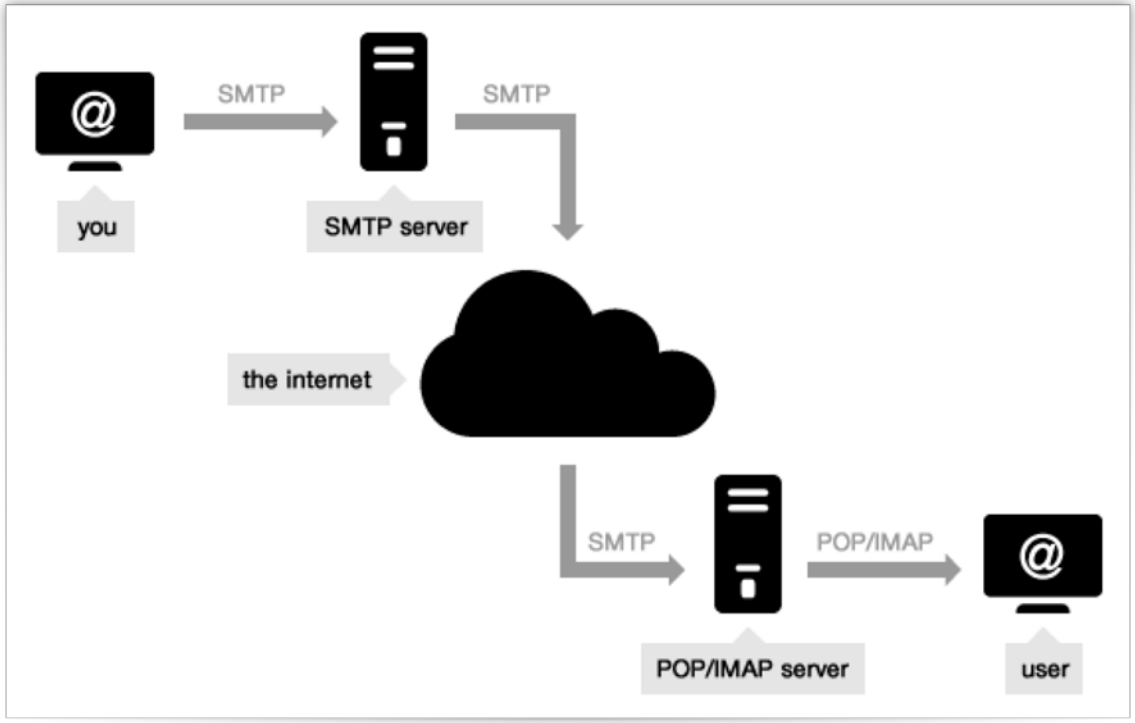
Sending Email

- 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.



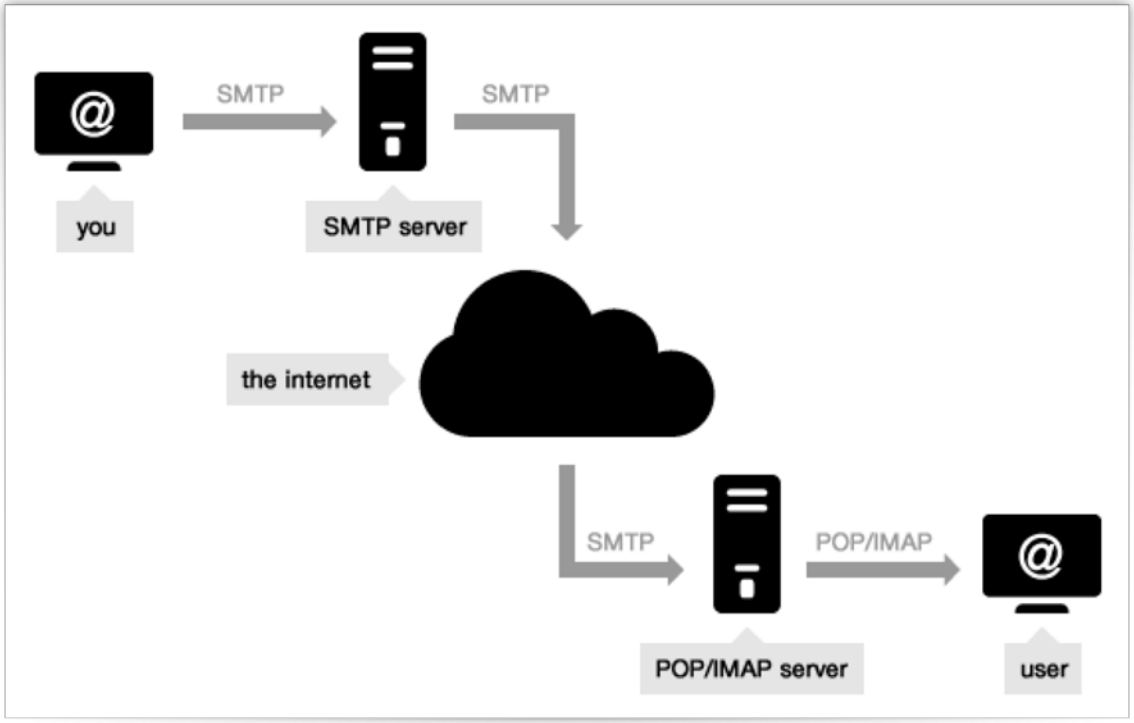
Sending Email

- 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

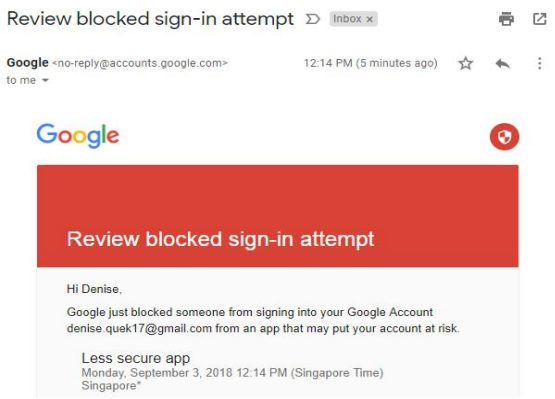
```
import smtplib

sender_email_address = "your_email_address@gmail.com"
sender_email_password = "xxxxxxxxxxxxxxx"
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
```

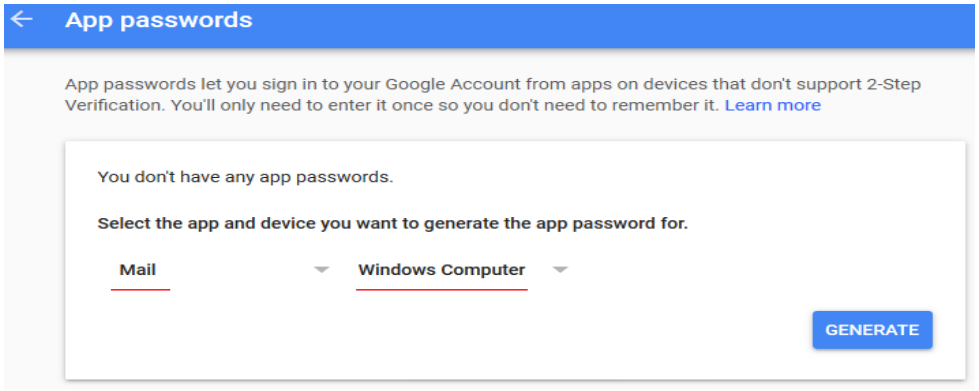
Send Email using Gmail

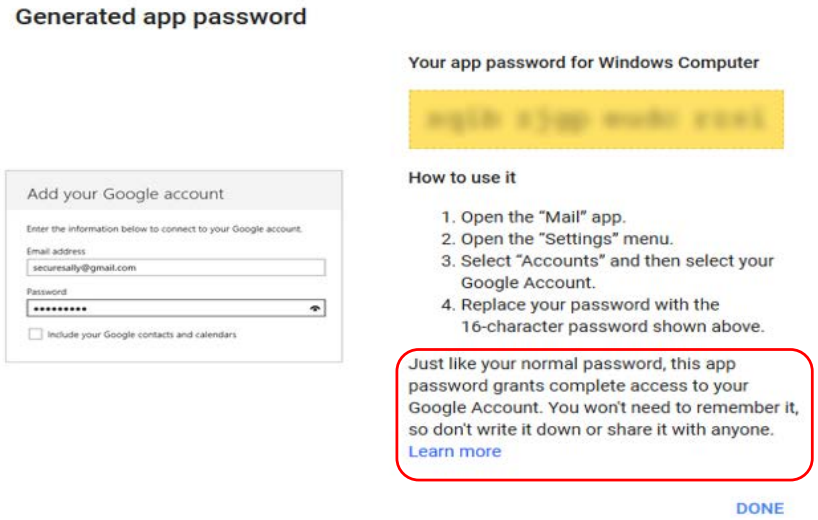
Steps To Create Google App Password

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





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 = "xxxxxxxxxxxxxxx"
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			

Sheet1

```
#! python3

import openpyxl, smtplib

def sendEmail(name, emailTo):
    email_body = "Subject: Your attendance. \nDear %s, \nYou were absent for class.\n" %(name)

    smtpObj = smtplib.SMTP("smtp.gmail.com", 587)
    smtpObj.starttls()
    smtpObj.login("code.musically@gmail.com", "xxxxxxxxxxx")
    smtpObj.sendmail('code.musically@gmail.com', emailTo, email_body)

    smtpObj.quit()
```

Send Email using Gmail

- Send email to students who were absent

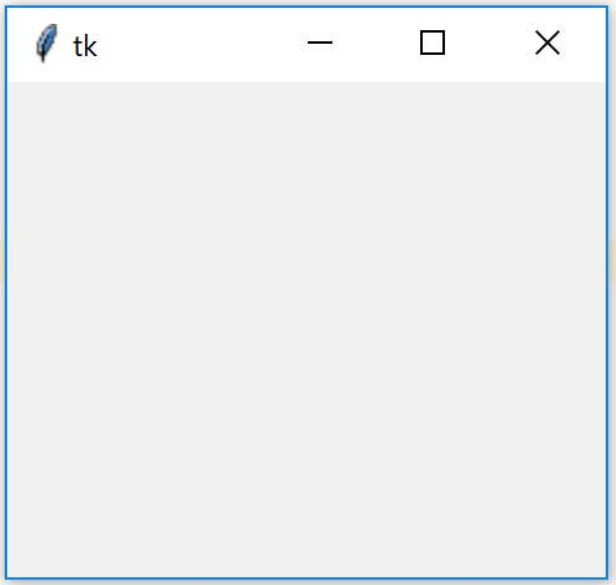
```
workbook = openpyxl.load_workbook("D:\CET_Python\students_attendance.xlsx")
sheet = workbook["Sheet1"]

max_row = sheet.max_row
max_column = sheet.max_column

for i in range(1, max_row+1):

    attendance = sheet.cell(row=i, column=3).value

    if attendance == "Absent":
        name = sheet.cell(row=i, column=1).value
        email = sheet.cell(row=i, column=2).value

        print(name + " is absent.")
        sendEmail(name, email)
        print("Email sent to " + email)
        print()
```

Graphical User Interface

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

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

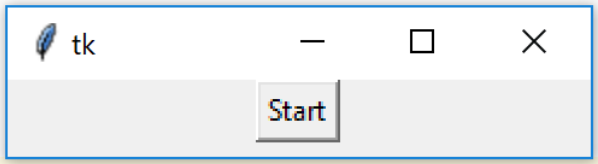
```
1  import tkinter
2
3  window = tkinter.Tk()
4  window.mainloop()
5
6
7
8
9
10
11
12
13
14
15
```



Graphical User Interface

Add a button

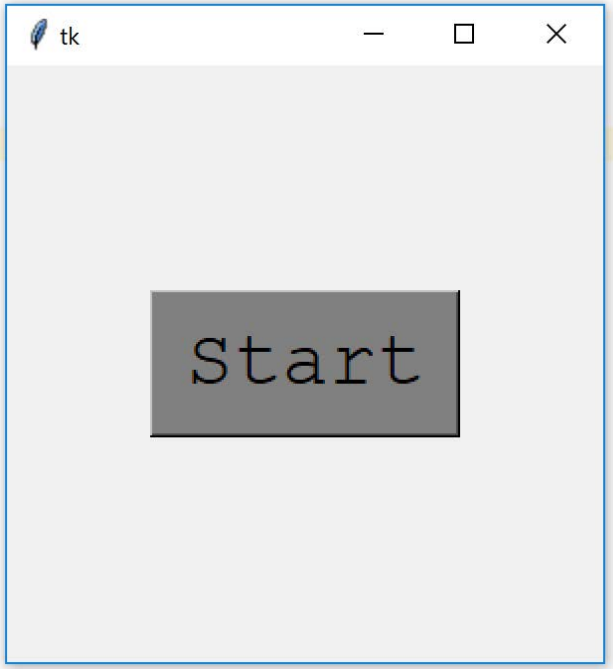
```
1  import tkinter
2
3  window = tkinter.Tk()
4
5  #Add a button
6  button = tkinter.Button(window, text="Start")
7  button.pack()
8
9  window.mainloop()
10
11
```



Graphical User Interface

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

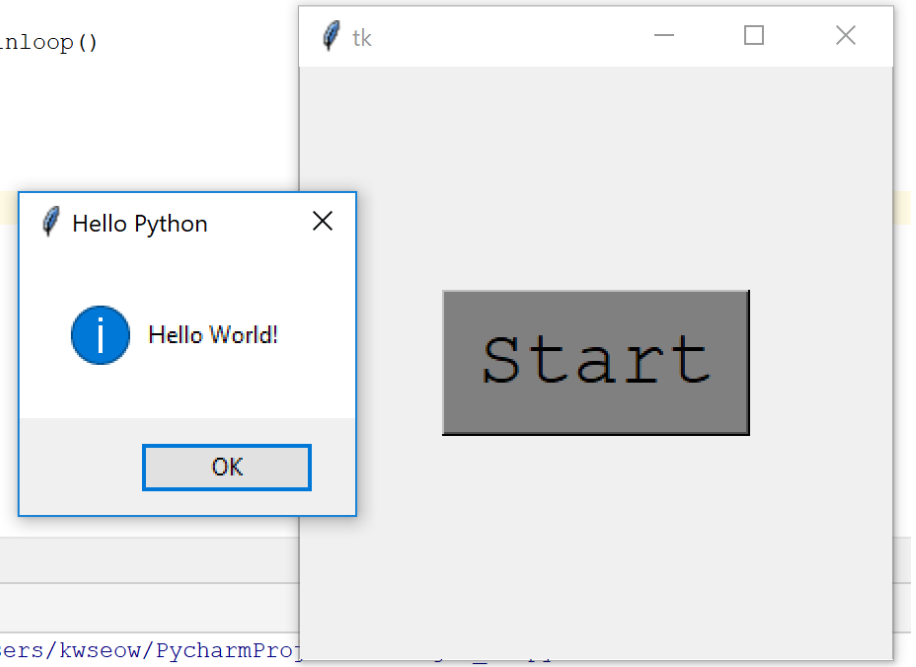
```
1  import tkinter
2
3  window = tkinter.Tk()
4
5  #set the window's size
6  window.geometry("300x300")
7
8  #Add a button
9  button = tkinter.Button(window, text="Start", bg="gray")
10 button.config(font=("Courier", 30))
11 button.pack(side="top", expand=tkinter.YES)
12
13 window.mainloop()
14
15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



Graphical User Interface

- Getting the button to response to click.

```python
import tkinter
import tkinter.messagebox

window = tkinter.Tk()

#set the window's size
window.geometry("300x300")

def displayMsg():
    tkinter.messagebox.showinfo("Hello Python","Hello World!")

#Add a button
button = tkinter.Button(window, text="Start", bg="gray", command=displayMsg)
button.config(font=("Courier", 30))
button.pack(side="top", expand=tkinter.YES)

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
```

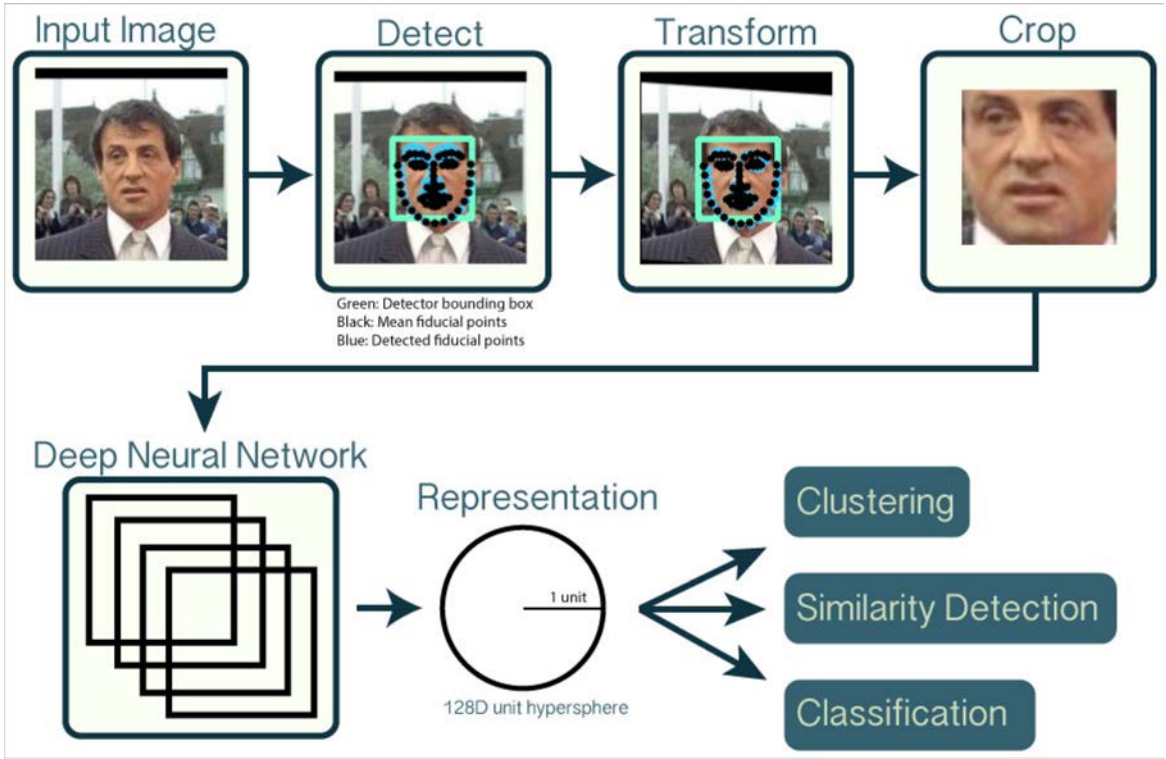
Run python setup.py py2exe and you have created a standalone executable
More at
http://www.py2exe.org/index.cgi/Tutorial#Step1

```
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



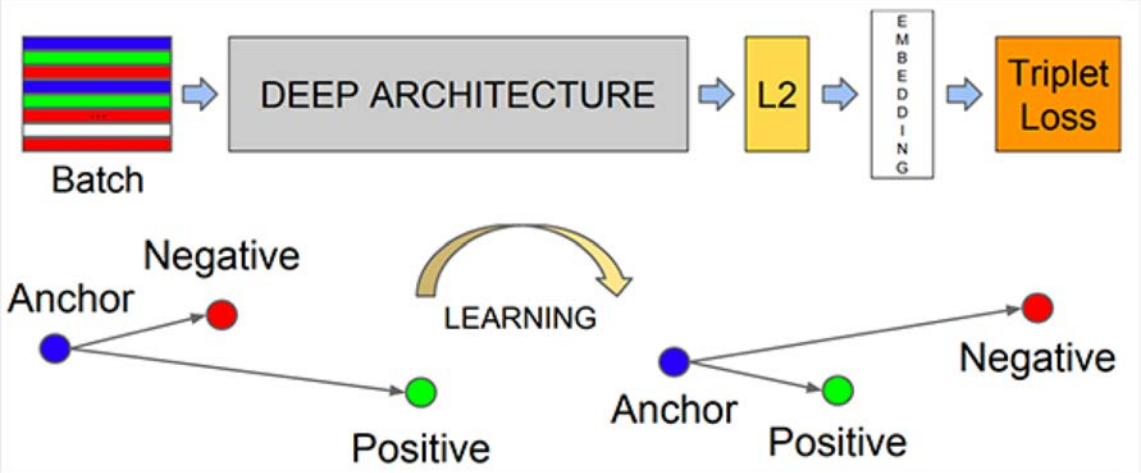
https://cmusatyalab.github.io/openface/

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
```

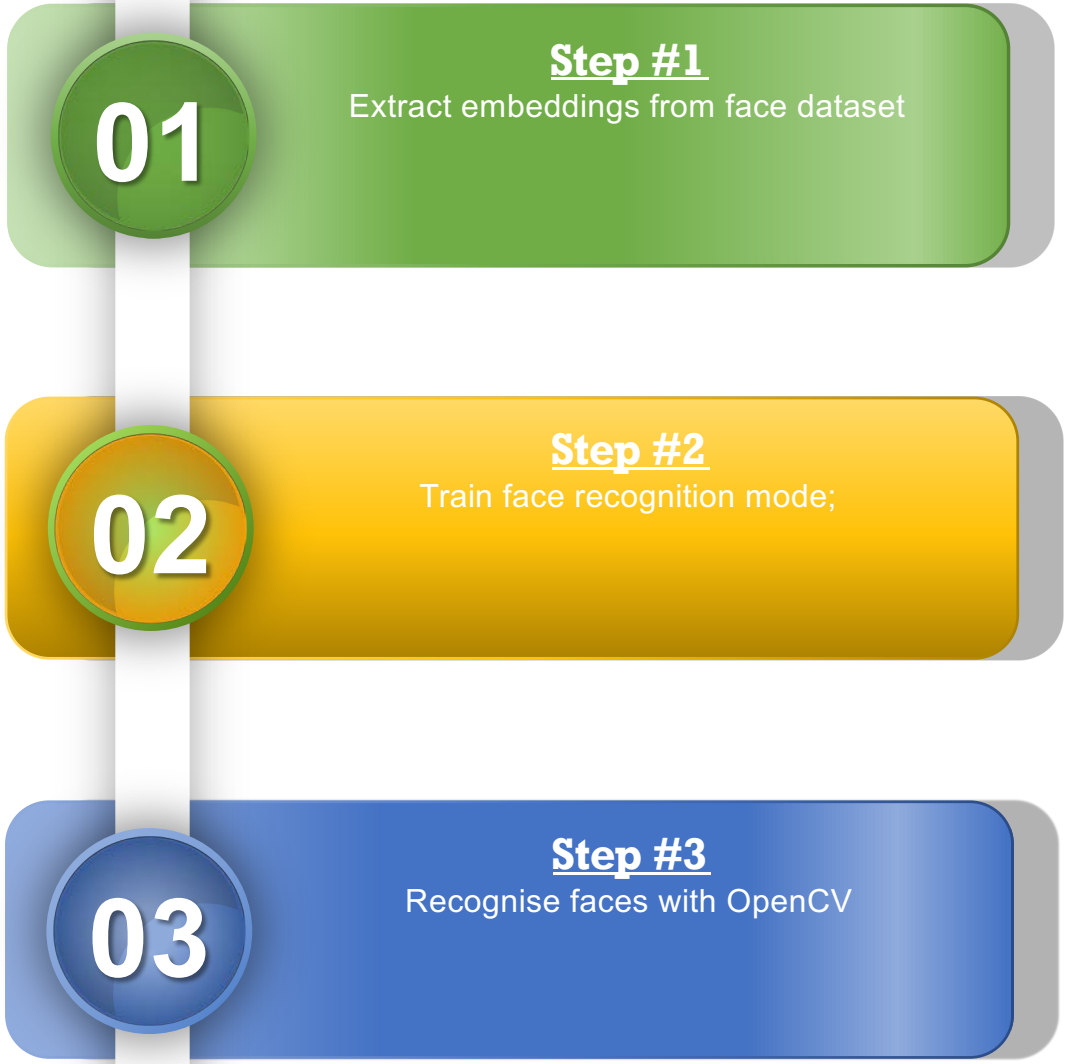


The model responsible for actually quantifying each face in an image is from the OpenFace project, a Python and Torch implementation of face recognition with deep learning. This implementation comes from Schroff et al.'s 2015 CVPR publication, *FaceNet: A Unified Embedding for Face Recognition and Clustering*.

Deep Learning Facial Recognition

- 3 Step Process

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



Deep Learning Facial Recognition

- Step 1 – Extract embeddings from face dataset

OpenCV Face Recognition — Python

```python
    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(image, (300, 300)), 1.0, (300, 300),
        (104.0, 177.0, 123.0), swapRB=False, crop=False)

    # apply OpenCV's deep learning-based face detector to localize
    # faces in the input image
    detector.setInput(imageBlob)
    detections = detector.forward()
```

OpenCV Face Recognition — Python

```python
    # ensure at least one face was found
    if len(detections) > 0:
        # we're making the assumption that each image has only ONE
        # face, so find the bounding box with the largest probability
        i = np.argmax(detections[0, 0, :, 2])
        confidence = detections[0, 0, i, 2]

        # ensure that the detection with the largest probability also
        # means our minimum probability test (thus helping filter out
        # weak detections)
        if confidence > args["confidence"]:
            # compute the (x, y)-coordinates of the bounding box for
            # the face
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # extract the face ROI and grab the ROI dimensions
            face = image[startY:endY, startX:endX]
            (fH, fW) = face.shape[:2]

            # ensure the face width and height are sufficiently large
            if fW < 20 or fH < 20:
                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 — Python

```python
# load the face embeddings
print("[INFO] loading face embeddings...")
data = pickle.loads(open(args["embeddings"], "rb").read())

# encode the labels
print("[INFO] encoding labels...")
le = LabelEncoder()
labels = le.fit_transform(data["names"])
```

OpenCV Face Recognition — Python

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

OpenCV Face Recognition — Python

```python
# write the actual face recognition model to disk
f = open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer))
f.close()

# write the label encoder to disk
f = open(args["le"], "wb")
f.write(pickle.dumps(le))
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

```python
# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)

# load our serialized face embedding model from disk
print("[INFO] loading face recognizer...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])

# load the actual face recognition model along with the label encoder
recognizer = pickle.loads(open(args["recognizer"], "rb").read())
le = pickle.loads(open(args["le"], "rb").read())
```

OpenCV Face Recognition — Python

```python
# load the image, resize it to have a width of 600 pixels (while
# maintaining the aspect ratio), and then grab the image dimensions
image = cv2.imread(args["image"])
image = imutils.resize(image, width=600)
(h, w) = image.shape[:2]

# construct a blob from the image
imageBlob = cv2.dnn.blobFromImage(
    cv2.resize(image, (300, 300)), 1.0, (300, 300),
    (104.0, 177.0, 123.0), swapRB=False, crop=False)

# apply OpenCV's deep learning-based face detector to localize
# faces in the input image
detector.setInput(imageBlob)
detections = detector.forward()
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

OpenCV Face Recognition — Python

```python
# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with the
    # prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections
    if confidence > args["confidence"]:
        # compute the (x, y)-coordinates of the bounding box for the
        # face
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # extract the face ROI
        face = image[startY:endY, startX:endX]
        (fH, fW) = face.shape[:2]

        # ensure the face width and height are sufficiently large
        if fW < 20 or fH < 20:
            continue
```

OpenCV Face Recognition — Python

```python
        # construct a blob for the face ROI, then pass the blob
        # through our face embedding model to obtain the 128-d
        # quantification of the face
        faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
            (0, 0, 0), swapRB=True, crop=False)
        embedder.setInput(faceBlob)
        vec = embedder.forward()

        # perform classification to recognize the face
        preds = recognizer.predict_proba(vec)[0]
        j = np.argmax(preds)
        proba = preds[j]
        name = le.classes_[j]
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

OpenCV Face Recognition — Python

```python
# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with the
    # prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections
    if confidence > args["confidence"]:
        # compute the (x, y)-coordinates of the bounding box for the
        # face
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # extract the face ROI
        face = image[startY:endY, startX:endX]
        (fH, fW) = face.shape[:2]

        # ensure the face width and height are sufficiently large
        if fW < 20 or fH < 20:
            continue
```

OpenCV Face Recognition — Python

```python
        # construct a blob for the face ROI, then pass the blob
        # through our face embedding model to obtain the 128-d
        # quantification of the face
        faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
            (0, 0, 0), swapRB=True, crop=False)
        embedder.setInput(faceBlob)
        vec = embedder.forward()

        # perform classification to recognize the face
        preds = recognizer.predict_proba(vec)[0]
        j = np.argmax(preds)
        proba = preds[j]
        name = le.classes_[j]
```

Deep Learning Facial Recognition

- Step 3 – Recognise faces with OpenCV

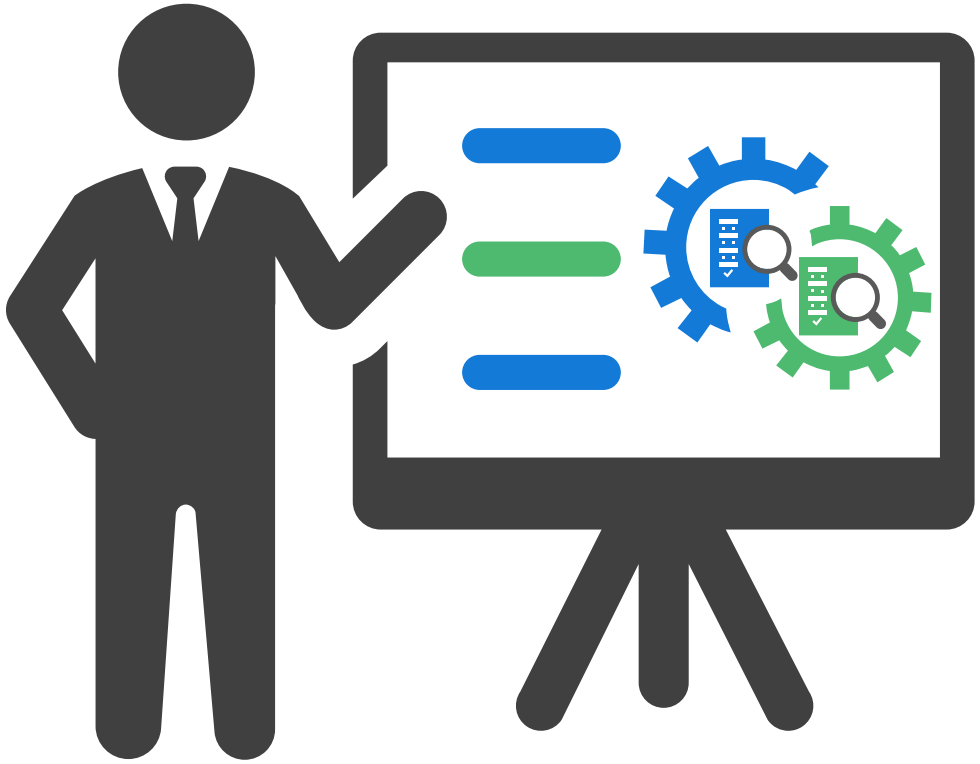
OpenCV Face Recognition

```python
        # draw the bounding box of the face along with the associated
        # probability
        text = "{}: {:.2f}%".format(name, proba * 100)
        y = startY - 10 if startY - 10 > 10 else startY + 10
        cv2.rectangle(image, (startX, startY), (endX, endY),
            (0, 0, 255), 2)
        cv2.putText(image, text, (startX, y),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)

# show the output image
cv2.imshow("Image", image)
cv2.waitKey(0)
```

https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

PSA UNIVERSITY

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

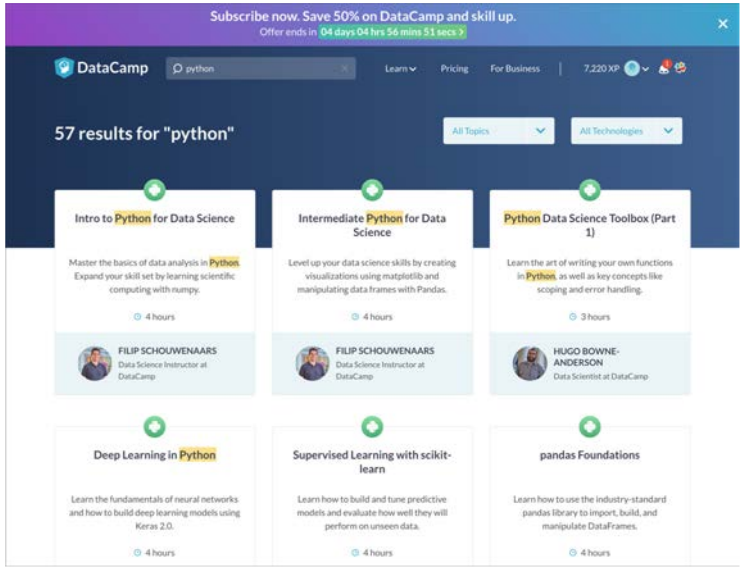
Advanced examples:
http://www.diveintopython.org/toc/index.html

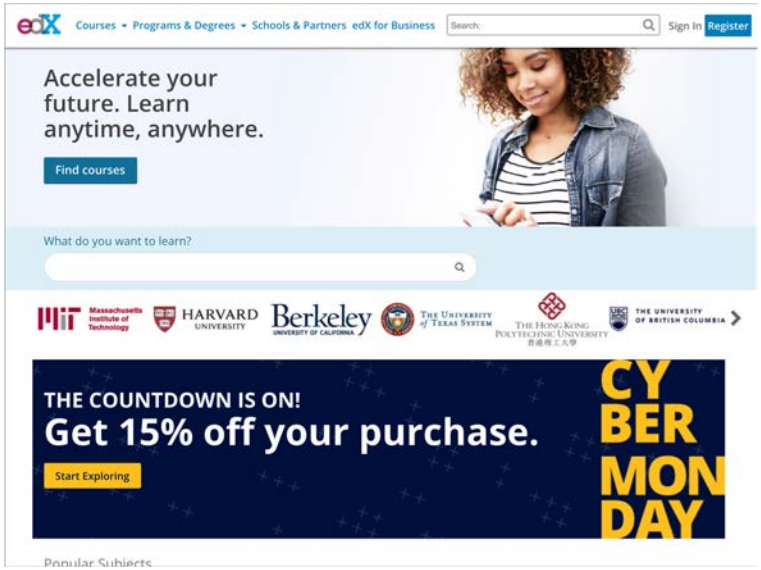
Where to go from here?

MOOC:
DataCamp
https://www.datacamp.com/

Edx
https://www.edx.org/

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





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

https://tinyurl.com/yb3u4tnv

Thank you