PSA UNIVERSITY

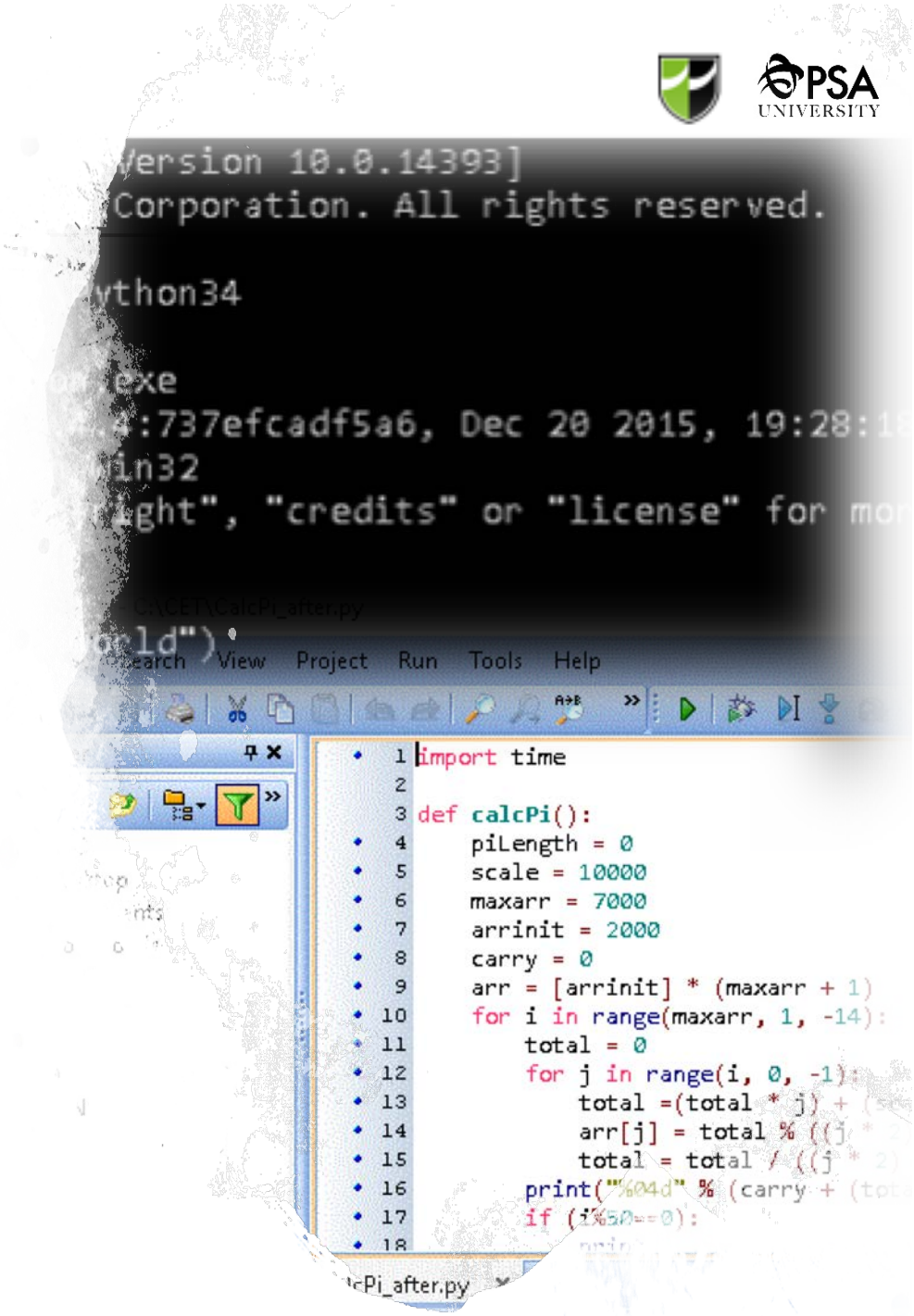
REPUBLIC POLYTECHNIC
DISCOVER. TRANSFORM. ACHIEVE

Introductory Programming in Python

Day 2

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
- Telegram bot



File Paths

Absolute file paths are notated by a **leading forward slash or drive label**.
For example,
`/home/example_user/example_directory` or
`C:/system32/cmd.exe`

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

Relative file paths are notated by a **lack of a leading forward slash**.
For example,
`example_directory`.

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

Read files

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

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 close() when you are done with the file.

- Call read() to read the contents of a file

```
Search | Stack Data
Search:
Replace:
Case sensitive  Whole words  In Selection

Debug I/O | Python Shell
Commands execute without debug.  Use arrow keys for history.  Options

3.7.4 (tags/v3.7.4:e09359112e, Jul  8
Python Type "help", "copyright", "cre
>>> [evaluate file_read_01.py]
THis is also another line
Hello world again

['THis is also another line\n', 'Hello world again\n']
```

Write files

```
1  # make sure you have a hello.txt in your current working director
2  # same directory as your python script
3  helloFile = open("hello.txt", "w")
4  helloFile.write("This is also another line\n")
5  helloFile.close()
6
7  # reopen to display content
8  helloFile = open("hello.txt")
9  print(helloFile.read())
10 helloFile.close()
11
12 # open the file for adding next text
13 helloFile = open("hello.txt", "a")
14 helloFile.write("Hello world again\n")
15 helloFile.close()
16
17 # reopen to display content
18 helloFile = open("hello.txt")
19 print(helloFile.read())
20 helloFile.close()
```

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.

Search | Stack Data

Search:

Replace:

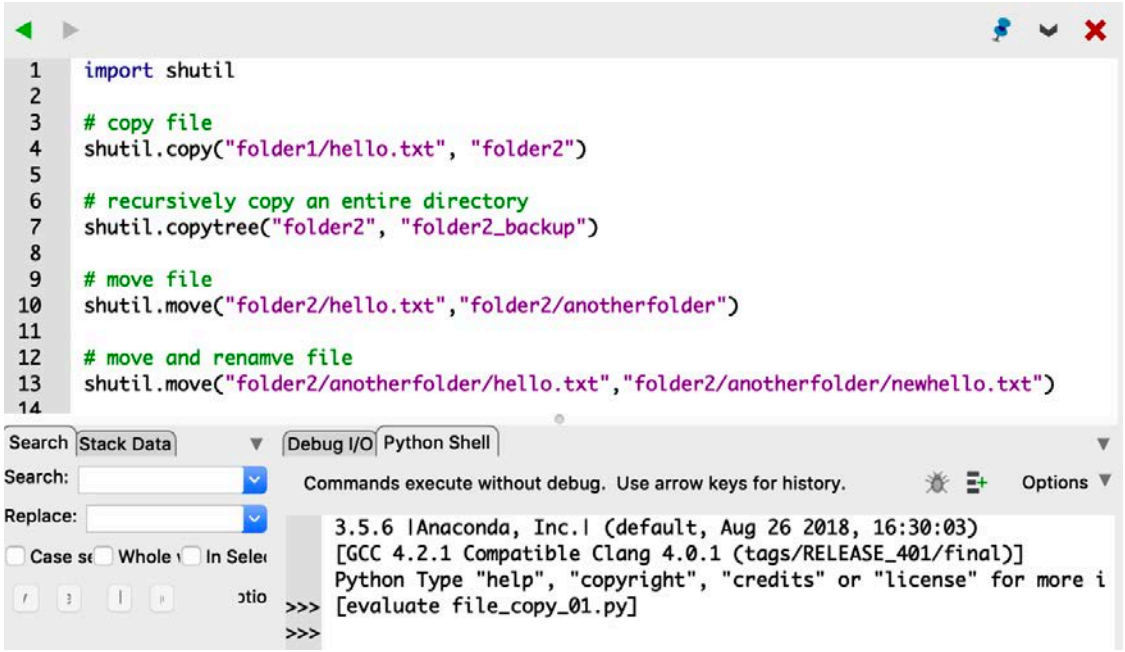
Case sensitive | Whole words | In Selection

Debug I/O | Python Shell

Commands execute without debug. Use arrow keys for history.

```
3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC
Python Type "help", "copyright", "credits" or "license" for
>>> [evaluate file_write..py]
THis is also another line

THis is also another line
Hello world again
```

Copy 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*).

https://docs.python.org/3/library/shutil.html

Deleting files

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

Search | Stack Data | Debug I/O | Python Shell

Search:

Replace:

Case s | Whole | In Sel | tio

Commands execute without debug. Use arrow keys for hist | Options

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" fo
>>> [evaluate file_delete_01.py]
/Users/kwseow/Dropbox/Projects/V7.PSA/Day2Resources
>>>
```

- 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

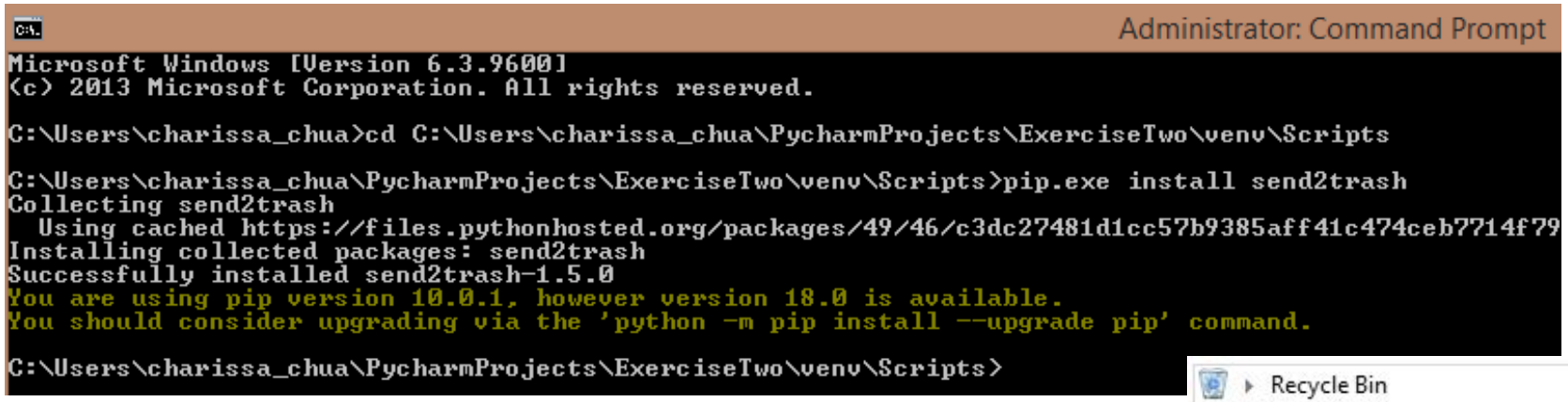
RemoveFiles.py

```
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)
```

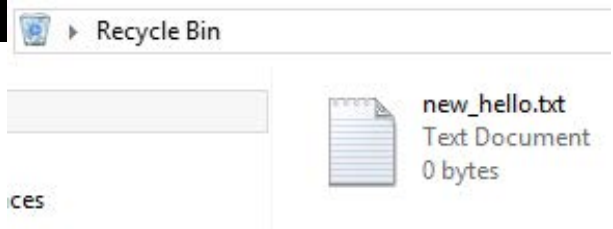
⚠ Deleting can be dangerous, so do a dry run first

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 to perform some tasks

```
D:\animals
|   animals.txt
|
+---cats
|       cute_kitten.jpg
|
\---dogs
    |   dogs.txt
    |
    \---retriever
            golden-retriever.jpg
```

```
1   import os
2
3   for folderName, subfolders, filenames in os.walk('D:\\animals'):
4       print('The current folder is ' + folderName)
5
6       for subfolder in subfolders:
7           print('SUBFOLDER OF ' + folderName + ': ' + subfolder)
8       for filename in filenames:
9           print('FILE INSIDE ' + folderName + ': '+ filename)
10
11      print('')
```

```
Python Type "help", "copyright", "credits" or "license" for m
[evaluate dir_walk.py]
The current folder is D:\animals
SUBFOLDER OF D:\animals: cats
SUBFOLDER OF D:\animals: dogs
FILE INSIDE D:\animals: animals.txt

The current folder is D:\animals\cats
FILE INSIDE D:\animals\cats: cute_kitten.jpg

The current folder is D:\animals\dogs
SUBFOLDER OF D:\animals\dogs: retriever
FILE INSIDE D:\animals\dogs: dogs.txt

The current folder is D:\animals\dogs\retriever
FILE INSIDE D:\animals\dogs\retriever: golden-retriever.jpg
```

os.walk()

The os.walk() function is passed a single string value: the path of a folder. You can use os.walk() in a for loop statement to walk a directory tree, much like how you can use the range() function to walk over a range of numbers. Unlike range(), the os.walk() function will return three values on each iteration through the loop:

- A string of the current folder's name
- A list of strings of the folders in the current folder
- A list of strings of the files in the current folder

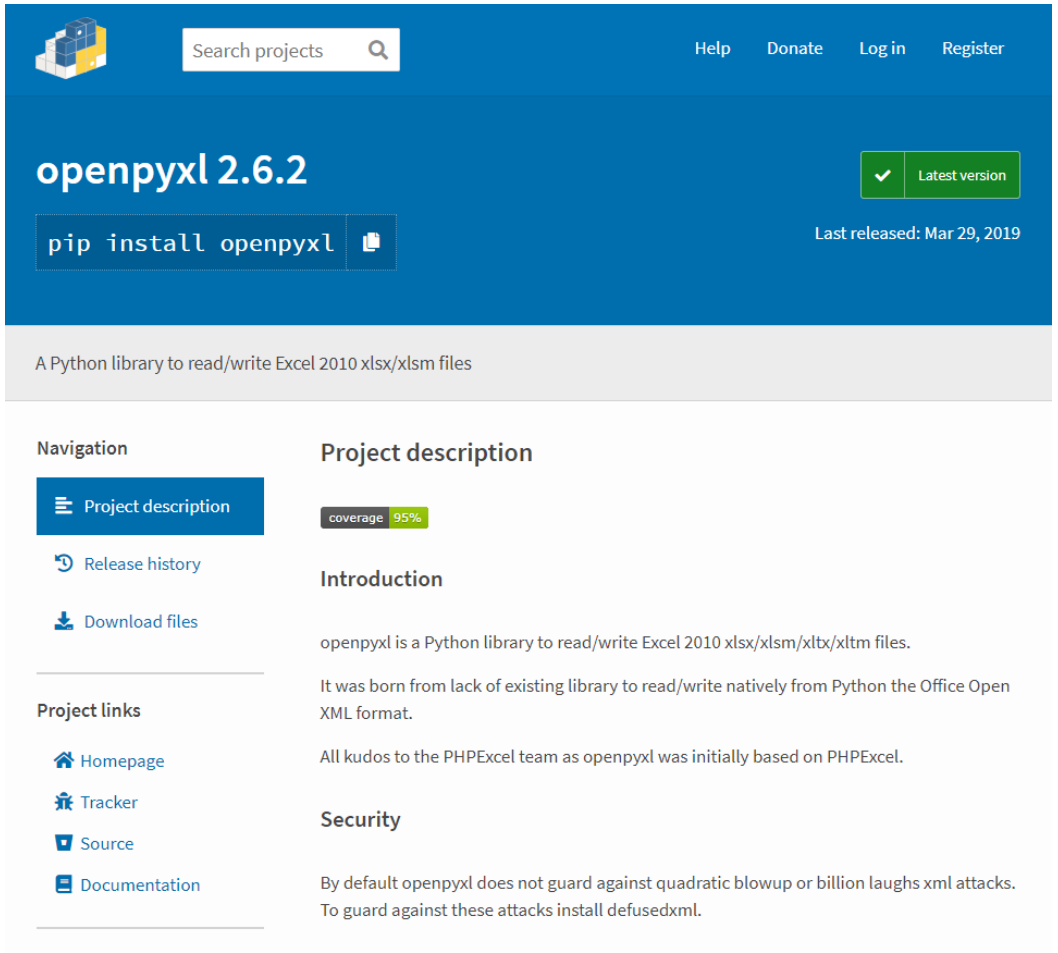
(By current folder, we mean the folder for the current iteration of the for loop. The current working directory of the program is not changed by os.walk().)

Exercise

- Write a script to list all the files in the C:\Users directory

Working with Excel

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



Reading Excel file

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

1) Import openpyxl

2) Load Excel content into "workbook" object by specifying the entire path

3) Get the active worksheet named "Sheet1"

4) Get the number of rows and columns

5) Use For loop to go through every row

6) Extract the status at Column C to check for attendance

Update Excel file

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

1) Import openpyxl

2) Load file into memory & get the sheet

3) Add value to cell

4) Add comments to cell

5) Save the spreadsheet

Create Excel file

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

1) Import openpyxl

2) Create new workbook

3) Get default sheet

4) Create dataset - a list of lists

5) Insert value into cells

6) Create a new sheet

7) Insert value into cells

8) Save the spreadsheet

Format Excel

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

Import necessary functions

Setup colors and styles

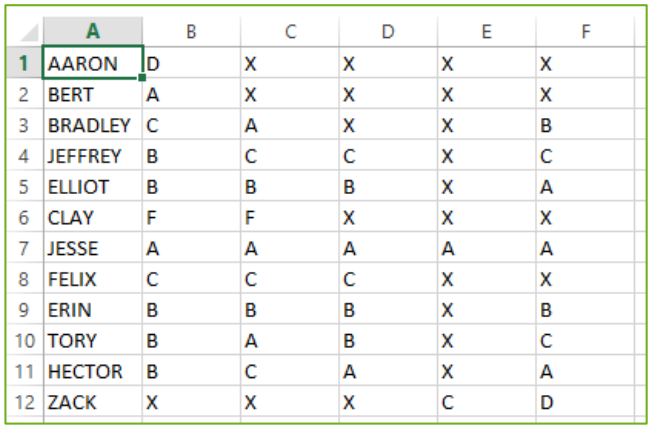
Loop through cell and set properties

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

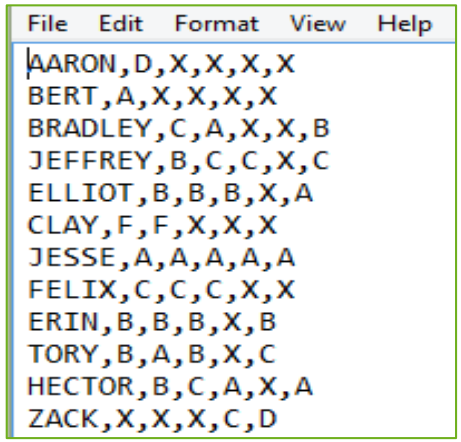
- The same data when viewed with Excel …



	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 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   readerFileHandle = open("W65Z.csv","r", newline='')
4   reader1 = csv.reader(readerFileHandle)
5   #using for loop to retrieve from the CSV file lune by line
6   for row in reader1:
7           print(row)
8
9   readerFileHandle.close()
10
```

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

5) Close the file (remove the link)

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)*

Writing 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.

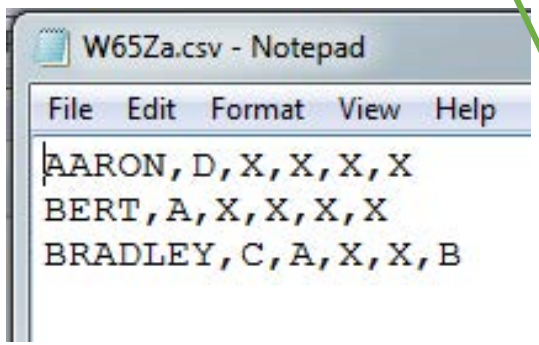
```
1   import csv
2
3   writerFileHandle = open("new.csv", "w", newline='')
4   writer1 = csv.writer(writerFileHandle)
5   row1 = ["Arron","D","X","X","X","X"]
6   row2 = ["Bert","A","X","C","B","X"]
7   row3 = ["Bradley","C","A","C","X","X"]
8   rowlist = [row1,row2,row3]
9
10  for row in rowlist:
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.

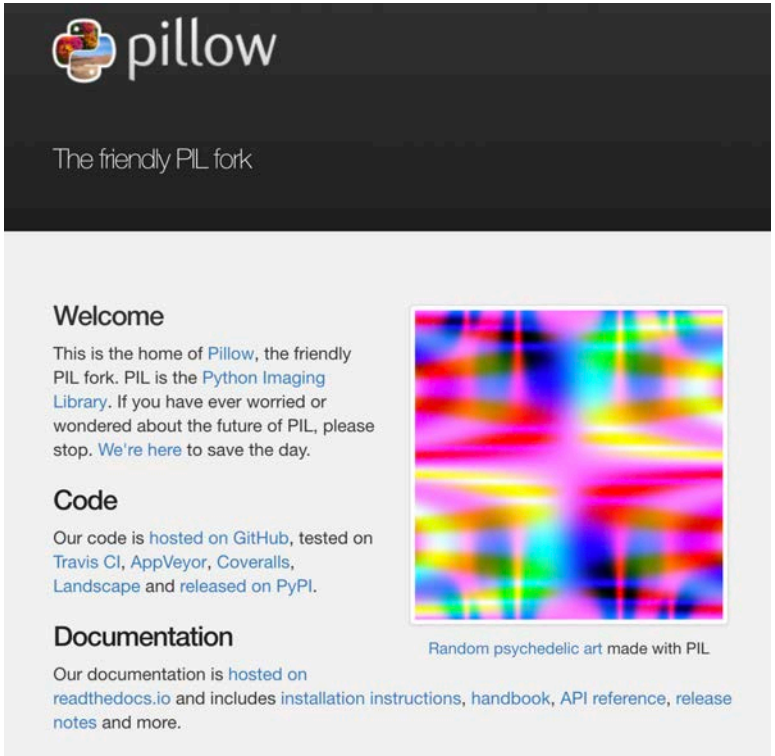
6) Close the file (Remove the link)



Exercise

- Write a script to read an Excel file and output every even rows to a csv file.

Image Processing



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

Install using the following command:
pip install Pillow

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

SearchByExtension

Before we start processing images,
Let us build a utility function to help in our work.

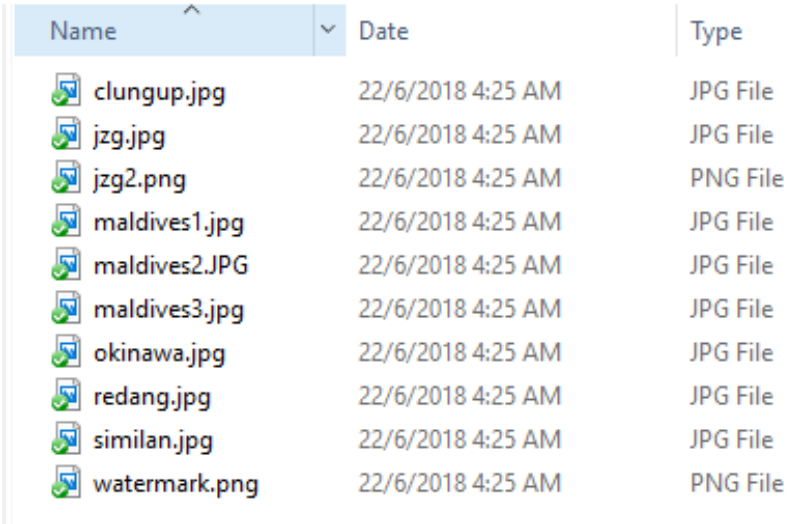
First, we need to be able to loop through all the images.

* We discussed the for loop when we talked about walking a directory.

```
1   import os
2
3   where = "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")
```

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.

SearchByExtension



Name	Date	Type
clungup.jpg	22/6/2018 4:25 AM	JPG File
jzg.jpg	22/6/2018 4:25 AM	JPG File
jzg2.png	22/6/2018 4:25 AM	PNG File
maldives1.jpg	22/6/2018 4:25 AM	JPG File
maldives2.JPG	22/6/2018 4:25 AM	JPG File
maldives3.jpg	22/6/2018 4:25 AM	JPG File
okinawa.jpg	22/6/2018 4:25 AM	JPG File
redang.jpg	22/6/2018 4:25 AM	JPG File
similan.jpg	22/6/2018 4:25 AM	JPG File
watermark.png	22/6/2018 4:25 AM	PNG File

```
3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.
Python Type "help", "copyright", "credits" or "license" for m
[evaluate image_processing.py]
 1 img\clungup.jpg
 2 img\jzg.jpg
 3 img\maldives1.jpg
 4 img\maldives3.jpg
 5 img\okinawa.jpg
 6 img\redang.jpg
 7 img\similan.jpg
```

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(...)

processAllImage()

```
1   import os
2
3   where = "img"
4
5   def processAllImage():
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  processAllImage()
```

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

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

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

You should now see a list of 10 images.

Note:
.

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.

processAllImage()

```
1  import os
2
3  where = "img"
4
5  def processAllImage(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 processAllImage(True)
```

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

Image Processing

```
1   import os
2   from PIL import Image
3
4   where = "img"
5
6   def processAllImage(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" % (c, fullname))
17                  im.show()
18                  c += 1
19                  if (onlyFirst):
20                      return
21
22  processAllImage(True)
```

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

Image Processing

```
1   import os
2   from PIL import Image
3
4   where = "img"
5
6   def processAllImage(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  processAllImage(True)
```

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

Image Processing

```
1  import os
2  from PIL import Image, ImageFilter, ImageOps
3
4  where = "img"
5
6  def processAllImage(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)" \
17                        % (c, fullname, im.size, im.mode))
18                 out = im.filter(ImageFilter.BLUR)
19
20                 im.show()
21                 out.show()
22                 c += 1
23                 if (onlyFirst):
24                     return
25
26 processAllImage(True)
```

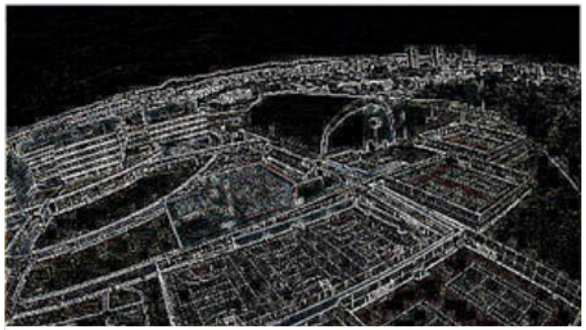
Now that we can load and understand the image, it is time to try and modify it.

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!

Image processing - filters



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

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

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

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

⚠ * Remember to include ImageOps in your import statement

Image Processing - Rotating

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)
```

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

Image Processing - Writing

```
1  import os
2  from PIL import Image, ImageFilter, ImageOps
3
4  where = "img"
5  outFolder = "out"
6
7  def processAllImage(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 %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 processAllImage(True)
```

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

Let's agree we store the output images in the folder **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)

Utility function – cleanOutput()

```
 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")
```

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

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.

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

Image processing – Converting

```
>>> 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
>>>
```

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 ?

Image processing – Converting

```
1  import os
2  from PIL import Image
3
4  where = "img"
5  outFolder = "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()
```

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.

You can create the output filename by :
outFileName =

os.path.join(outFolder,os.path.splitext(file)[0]+".jpg")

Image processing – Watermark

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

```
mark = Image.open("img\\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 like:

```
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 \\img\\watermark.png and place it in each image on the bottom right.

Image processing – Watermark

```
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   im = Image.open("img\\clungup.jpg")
9   mark = Image.open("img\\watermark.png")
10  mark = mark.resize((100,100))
11
12  out = watermark(im, mark, (0,50))
13  out.show()
14
```

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:

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

Connecting to the Web

```
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
```

- 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

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

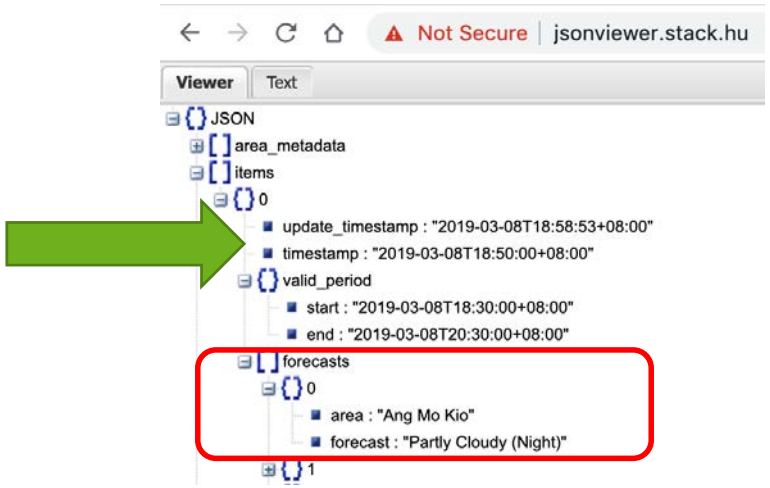
```
1   {"area_metadata":[{"name":"Ang Mo
    Kio","label_location":{"latitude":1.375,"
    longitude":103.839}},{"name":"Bedok","lab
    el_location":{"latitude":1.321,"longitude
    ":103.924}},{"name":"Bishan","label_locat
```

Connecting to the Web

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

```
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",
```

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)
```

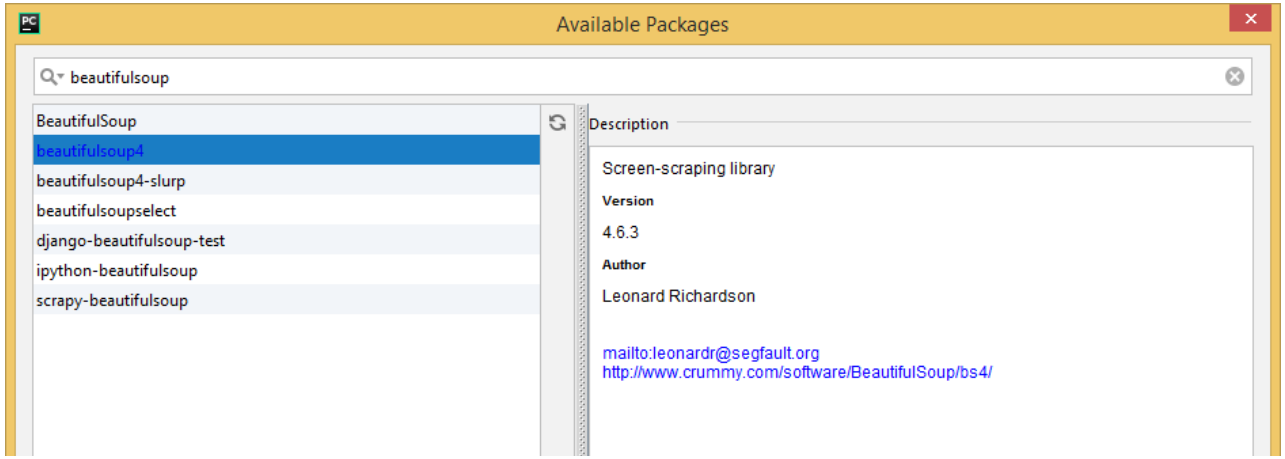
```
C:\Users\denise_quek\AppData\Local\Programs\Python\Py
Ang Mo Kio: Thundery Showers
Bedok: Thundery Showers
Bishan: Heavy Thundery Showers with Gusty Winds
Boon Lay: Heavy Thundery Showers with Gusty Winds
Bukit Batok: Heavy Thundery Showers with Gusty Winds
Bukit Merah: Heavy Thundery Showers with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

Connecting to the Web

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

Web Scraping – download and process Web content

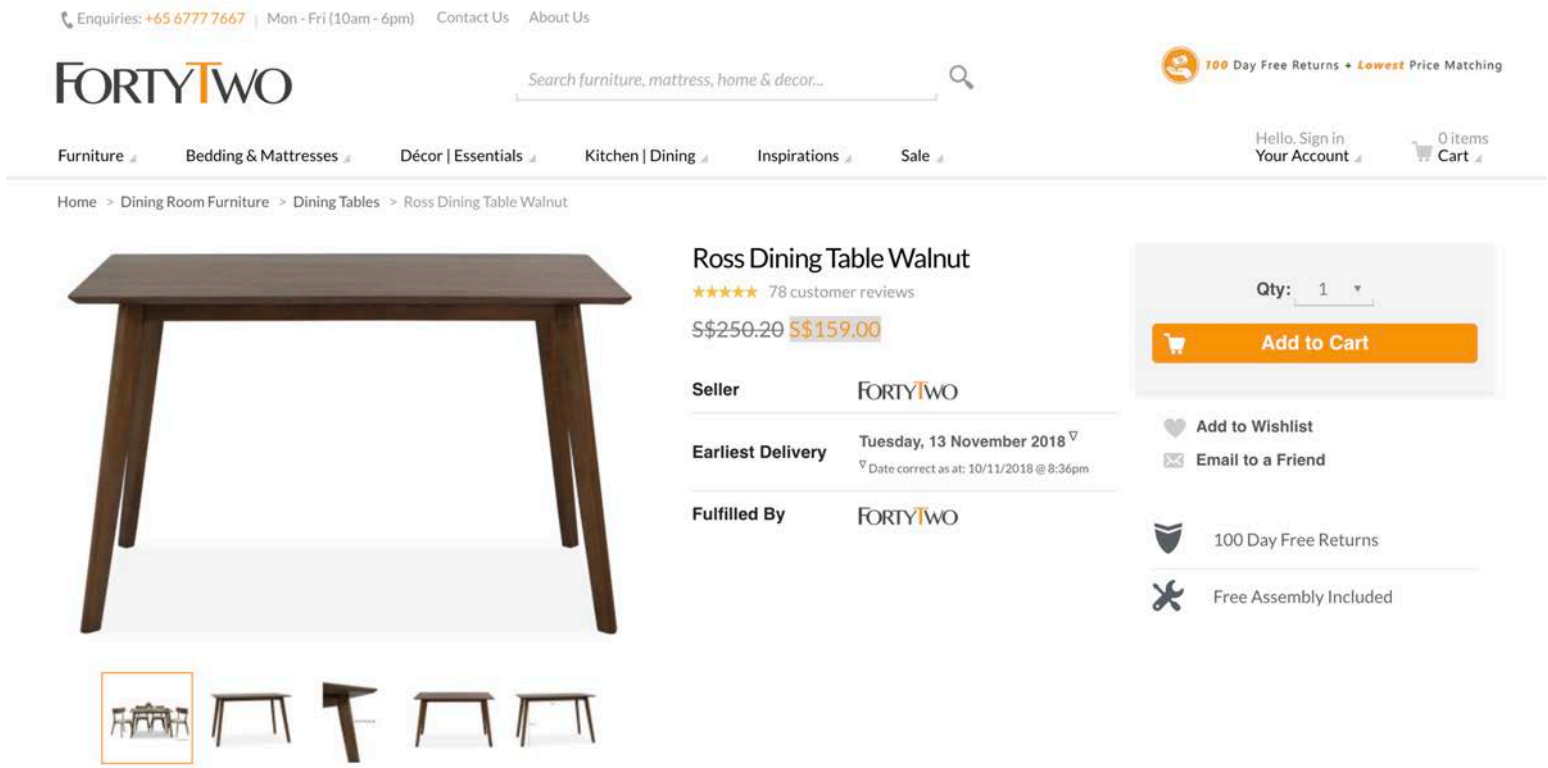
- Install Beautiful Soup 4 -



Connecting to the Web

- What's the URL?

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

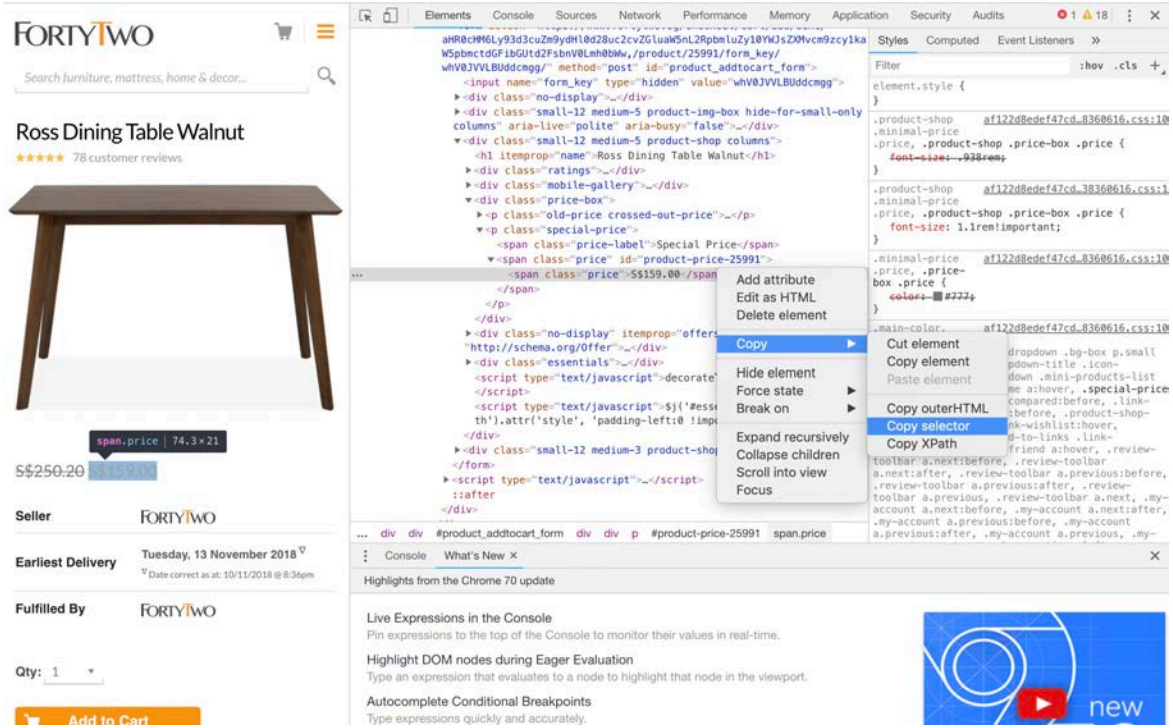


Connecting to the Web

- Get the url

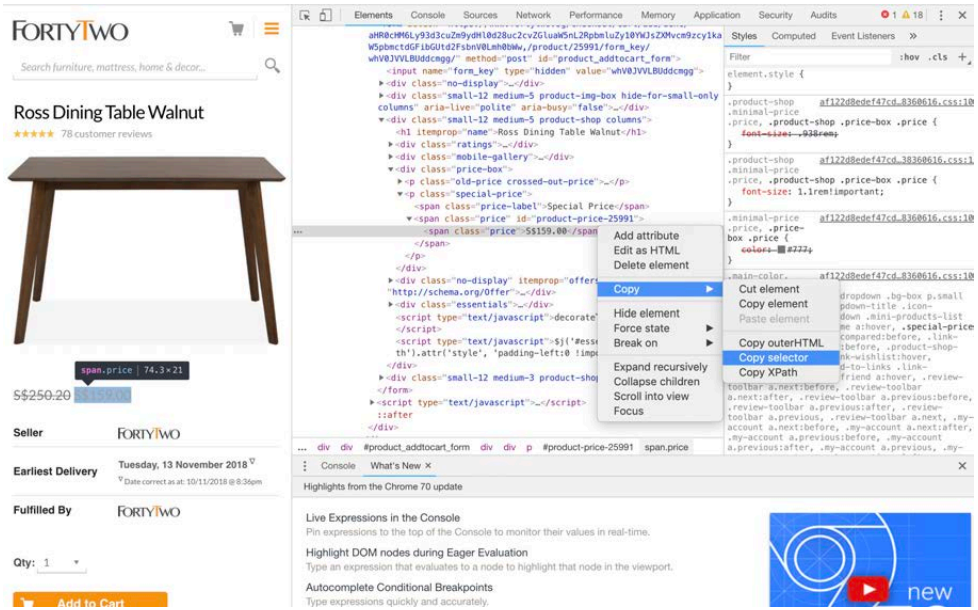
https://www.fortytwo.sg/dining/dining-tables/ross-dining-table-walnut.html

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



Connecting to the Web

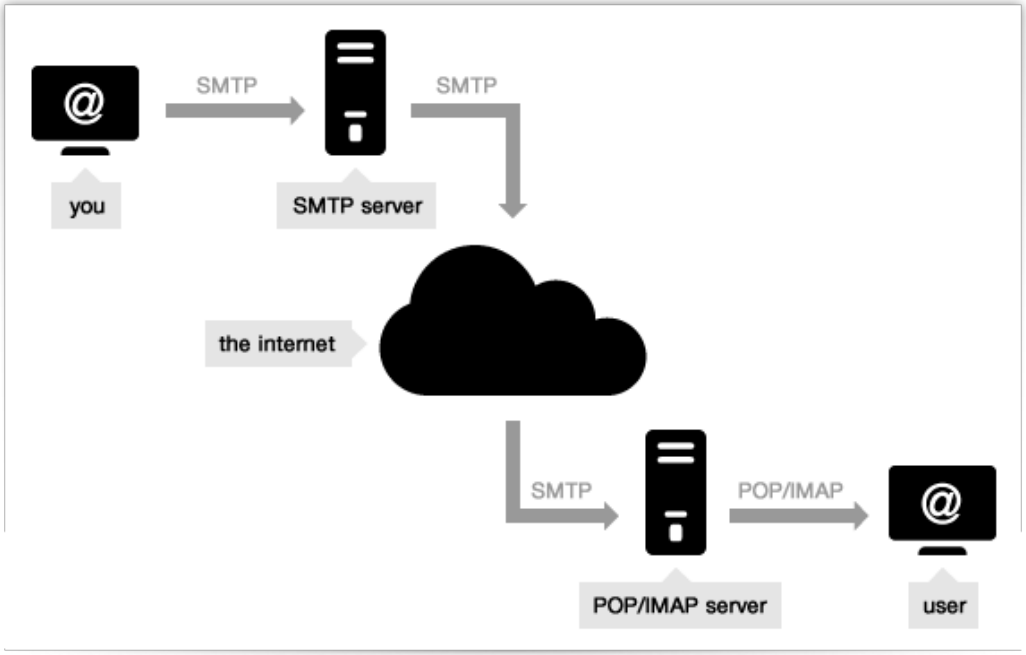
- Get the url
- Select the element to extract, right-click "Inspect"
- Right-click "Copy" → "Copy selector"




```
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
```

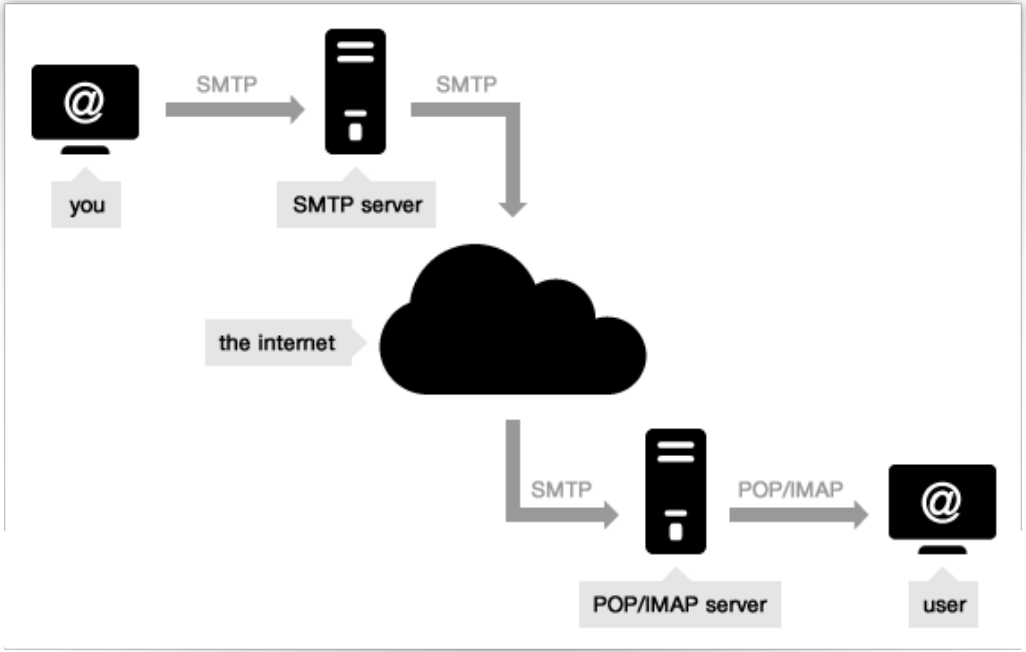
Send Email



- 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.

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

Send 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.

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

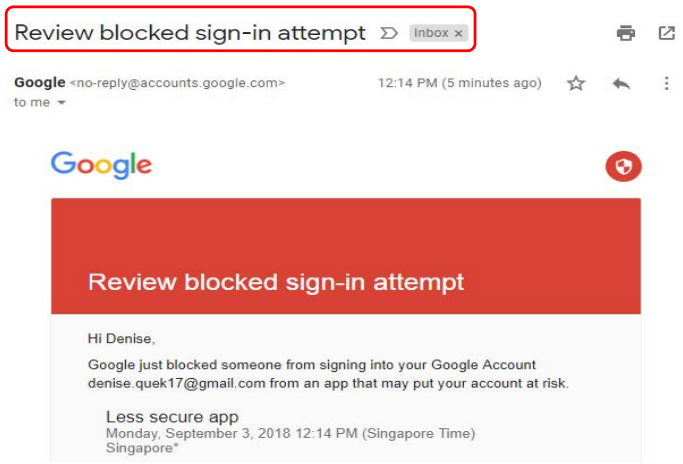
```
import smtplib

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
```

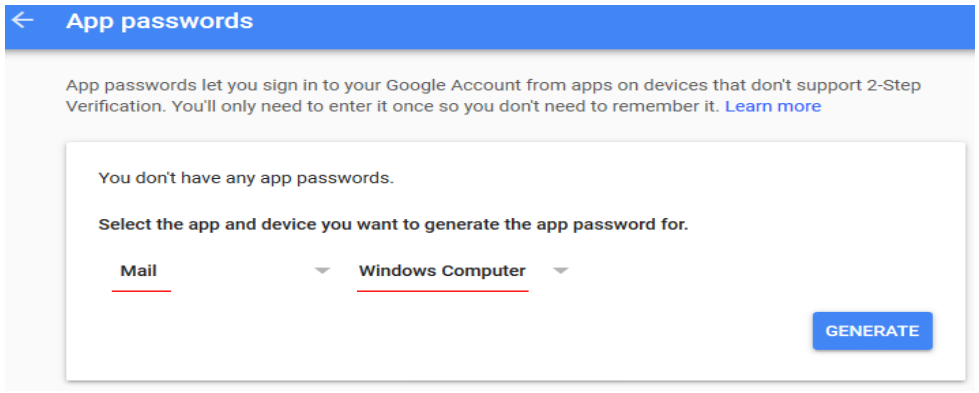
Send Email using Gmail

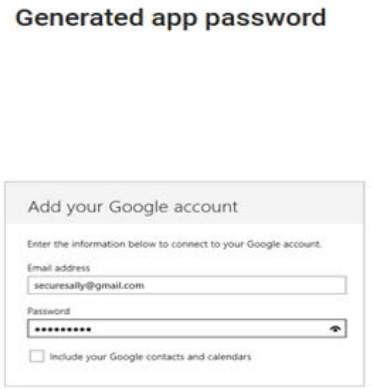
Steps To Create Google App Password

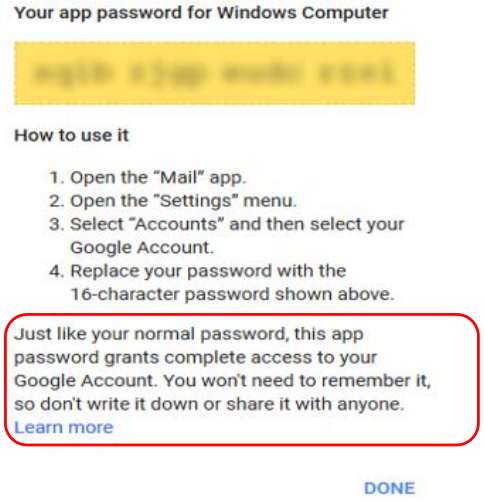
Step 1: Login to Gmail. Go to Account → Signing in to Google

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

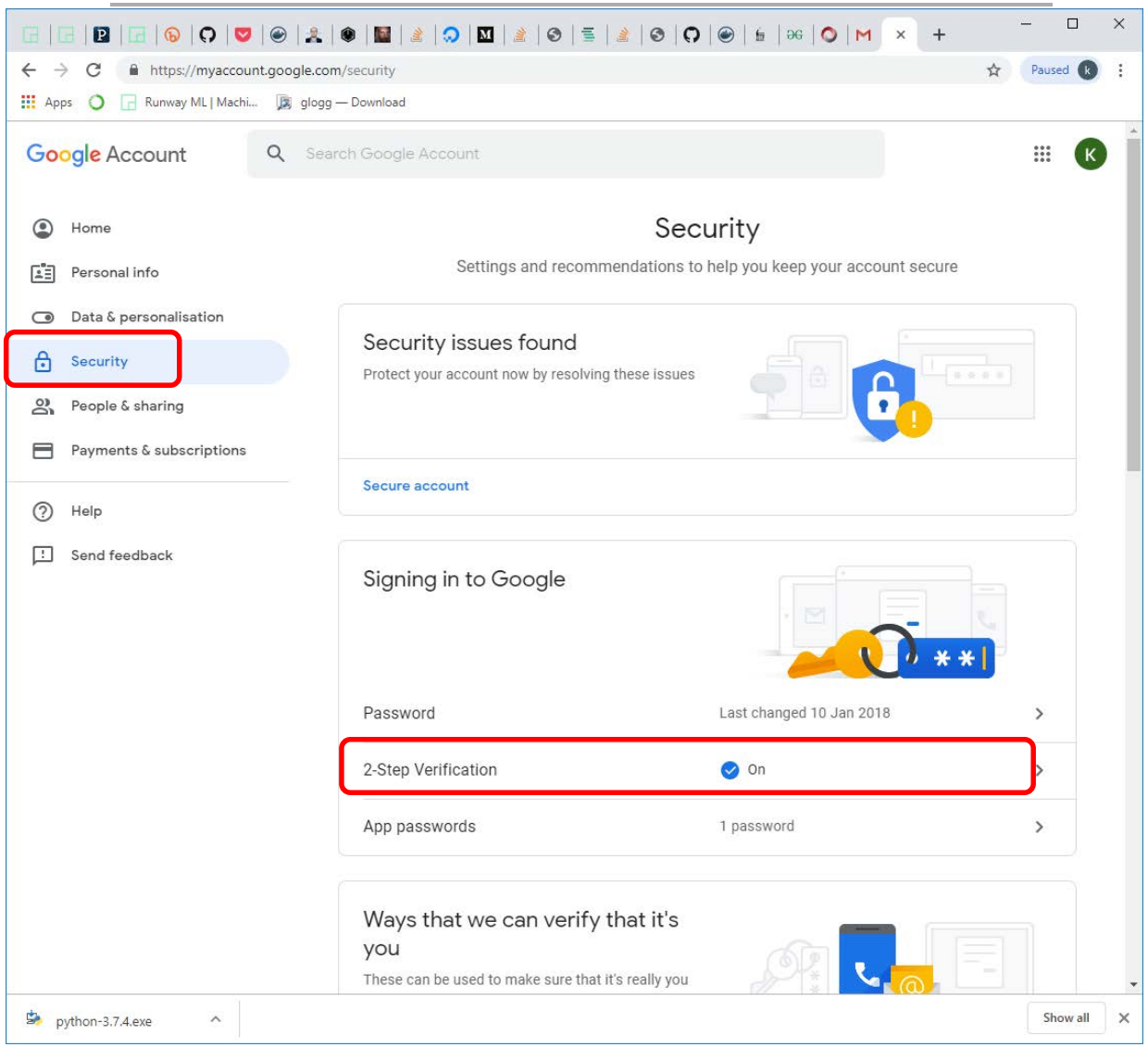
Step 3: Create an App password







Send Email using Gmail



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.mu	
6	Evelyn	code.mu	
7			

Shee

```python
#! 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

```
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
```

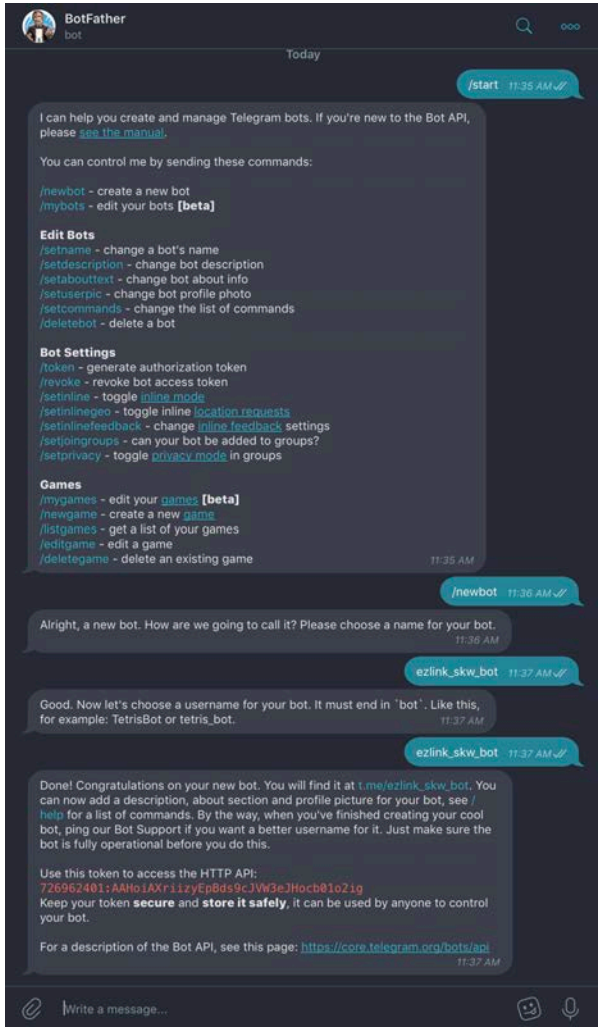
Send Email using Yahoo

MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP).

```
1   import smtplib
2   from email.mime.text import MIMEText
3   SMTP_SERVER = "smtp.mail.yahoo.com"
4   SMTP_PORT = 587
5
6   sender_yahoo_account ="seow_khee_wei@yahoo.com.sg"
7   sender_yahoo_password = "your_yahoo_account_passowrd"
8   sender_email_address ="seow_khee_wei@yahoo.com.sg"
9   receiver_email_address = "kwseow@gmail.com"
10  email_msg = "This is a test mail.\n\nRegards"
11
12  msg = MIMEText(email_msg)
13  msg['Subject'] = "Service at appointmentTime"
14  msg['From'] = sender_email_address
15  msg['To'] = receiver_email_address
16
17  print("Trying to connect o yahoo SMTP server")
18  smtpObj = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
19  smtpObj.set_debuglevel(True)
20  smtpObj.starttls()
21
22  print("Connected.  Logging in...")
23  smtpObj.login(sender_yahoo_account, sender_yahoo_password)
24  smtpObj.sendmail(sender_email_address, receiver_email_address, msg.as_string())
25  smtpObj.quit()
26
27  print("Email sent successfully...")
```

Use MIMEText to format message body

Telegram Bot



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

Telegram Bot

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

```python
import sys
import time
import telepot
from telepot.loop import MessageLoop

TOKEN = '<YOUR TOKEN>'

# Define the information to return per command
def get_help():
    msg = '''
    Use one of the following commands:
        help: To show this help
        offers: To see this week offers
        events: To see this week events
    '''
    return msg

def get_offers():
    msg = '''
    This week enjoy these amazing offers!
        20% discount in beach products
        15% discount if you spend more than $50
    '''
    return msg

def get_events():
    msg = '''
    Join us for an incredible party the Thursday in our Marina Bay Sands shop!
    '''
    return msg

COMMANDS = {
    'help': get_help,
    'offers': get_offers,
    'events': get_events,
}
```

response

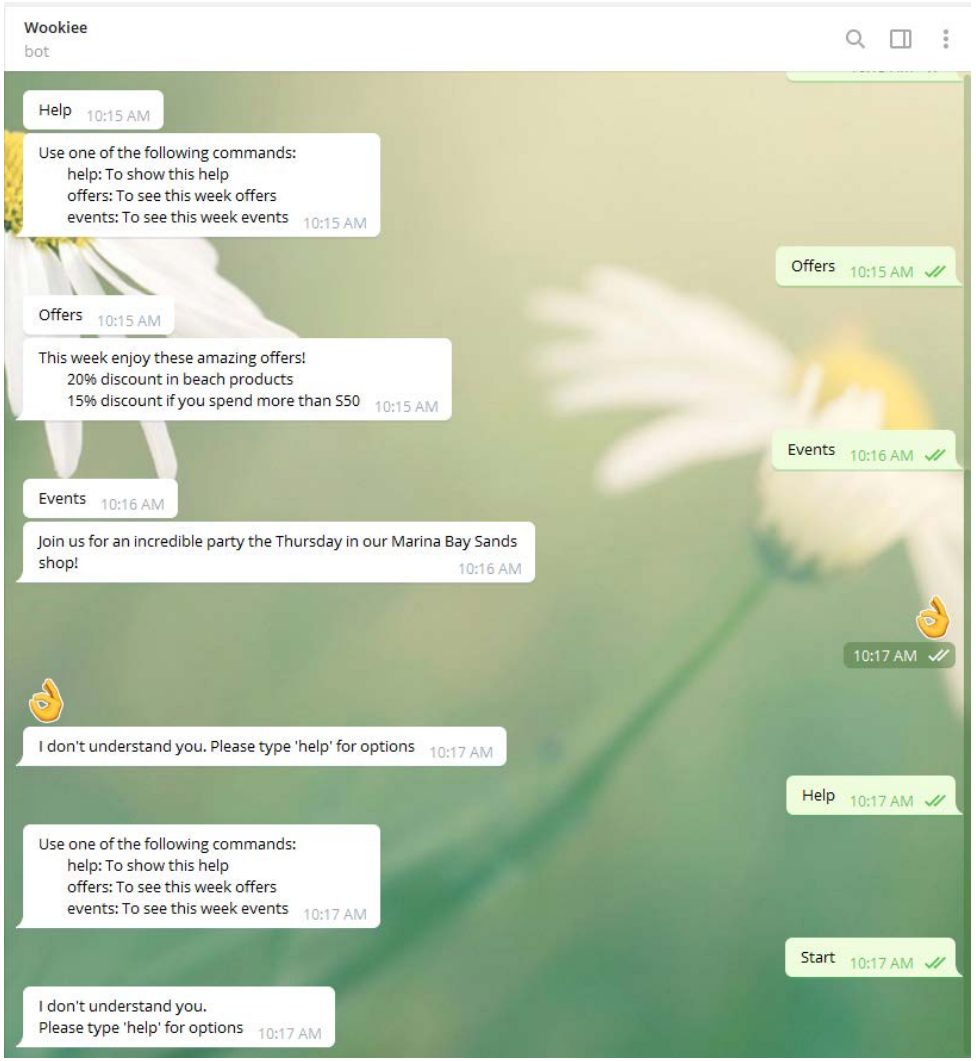
intents

Telegram Bot

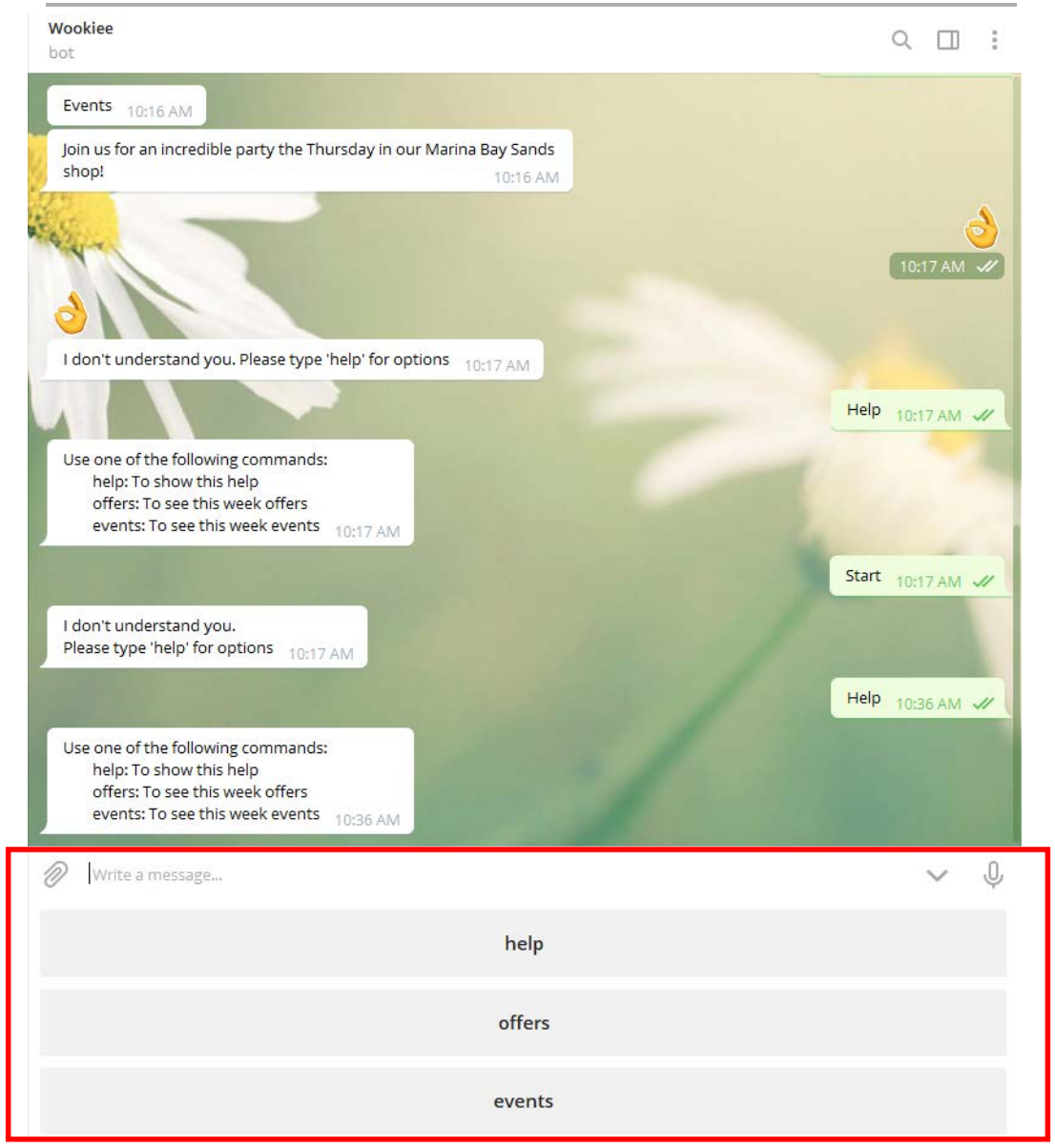
```
38  def handle(msg):
39      content_type, chat_type, chat_id = telepot.glance(msg)
40      print(content_type, chat_type, chat_id)
41
42      if content_type != 'text':
43          bot.sendMessage(chat_id, "I don't understand you.\nPlease type 'help' for options")
44          return
45
46      elif content_type == 'text':
47          # Make the commands case insensitive
48          command = msg['text'].lower()
49          if command not in COMMANDS:
50              bot.sendMessage(chat_id,"I don't understand you.\nPlease type 'help' for options")
51              return
52          message = COMMANDS[command]()
53          bot.sendMessage(chat_id,message)
54
55  bot = telepot.Bot(TOKEN)
56  MessageLoop(bot, handle).run_as_thread()
57  print ('Listening ...')
58
59  # Keep the program running.
60  while 1:
61      time.sleep(10)
```

Process the message

Telegram Bot



Telegram – Custom Keyboard



Telegram – Custom Keyboard

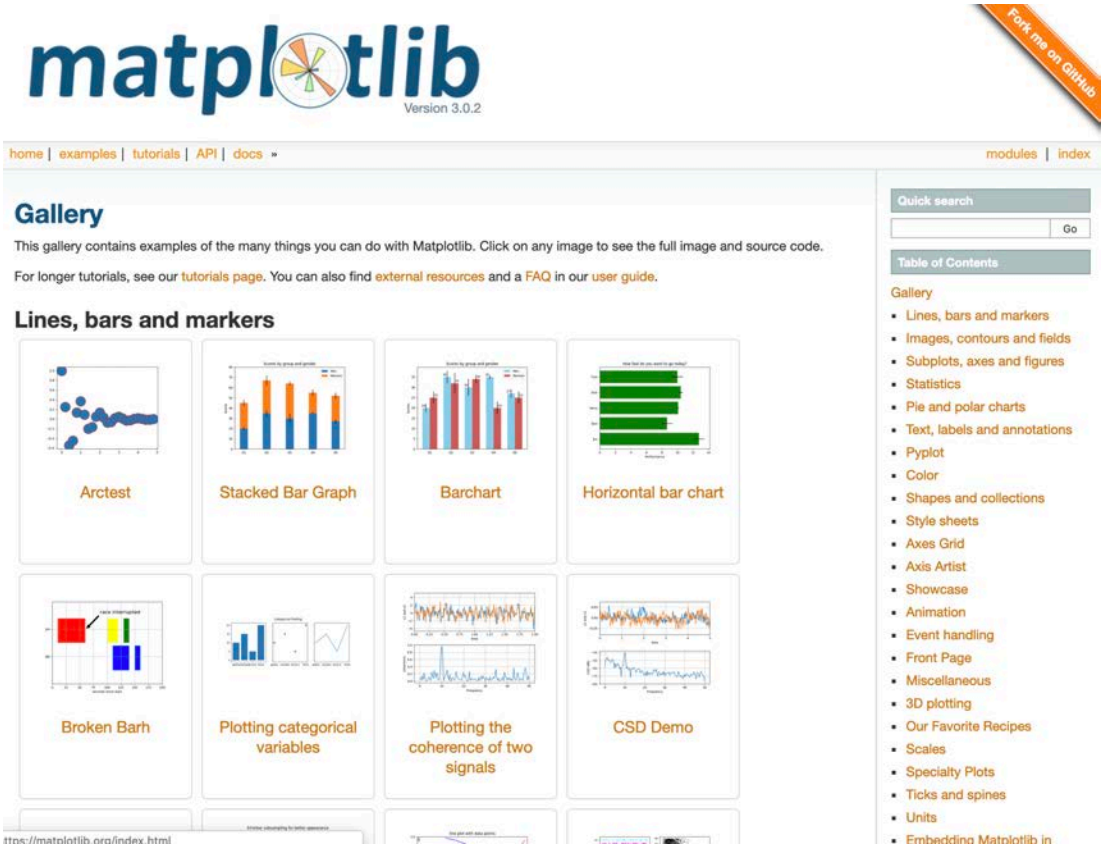
```python
import sys
import time
import telepot
from telepot.loop import MessageLoop
from telepot.namedtuple import ReplyKeyboardMarkup, KeyboardButton

TOKEN = '<YOUR TOKEN>'

# Create a custom keyboard with only the valid responses
keys = [[KeyboardButton(text=text)] for text in COMMANDS]
KEYBOARD = ReplyKeyboardMarkup(keyboard=keys)

# Define the information to return per command
def get_help():
    msg = '''
    Use one of the following commands:
        help: To show this help
        offers: To see this week offers
        events: To see this week events
    '''
    return msg

def get_offers():
    msg = '''
    This week enjoy these amazing offers!
        20% discount in beach products
        15% discount if you spend more than $50
    '''
    return msg

def get_events():
    msg = '''
    Join us for an incredible party the Thursday in our Marina Bay Sands shop!
    '''
    return msg
```

Telegram – Custom Keyboard

- Set the reply_markup parameter to the custom keyboard in your call to sendMessage()

```
43  def handle(msg):
44      content_type, chat_type, chat_id = telepot.glance(msg)
45      print(content_type, chat_type, chat_id)
46
47      if content_type != 'text':
48          bot.sendMessage(chat_id, "I don't understand you.\nPlease type 'help' for options",reply_markup=KEYBOARD)
49          return
50
51      elif content_type == 'text':
52          # Make the commands case insensitive
53          command = msg['text'].lower()
54          if command not in COMMANDS:
55              bot.sendMessage(chat_id,"I don't understand you.\nPlease type 'help' for options",reply_markup=KEYBOARD)
56              return
57          message = COMMANDS[command]()
58          bot.sendMessage(chat_id,message,reply_markup=KEYBOARD)
59
```

Charting

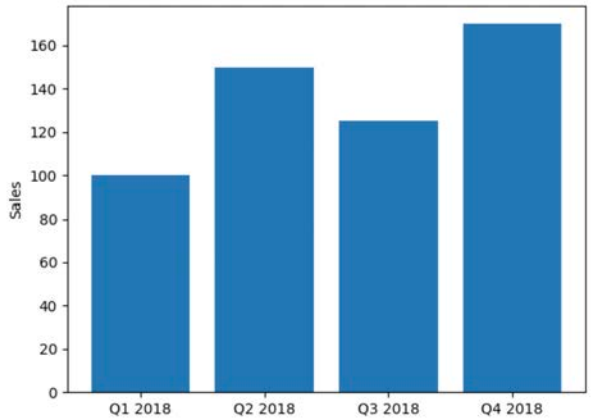


Install matplotlib

Full documentation:
https://matplotlib.org/

Charting

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



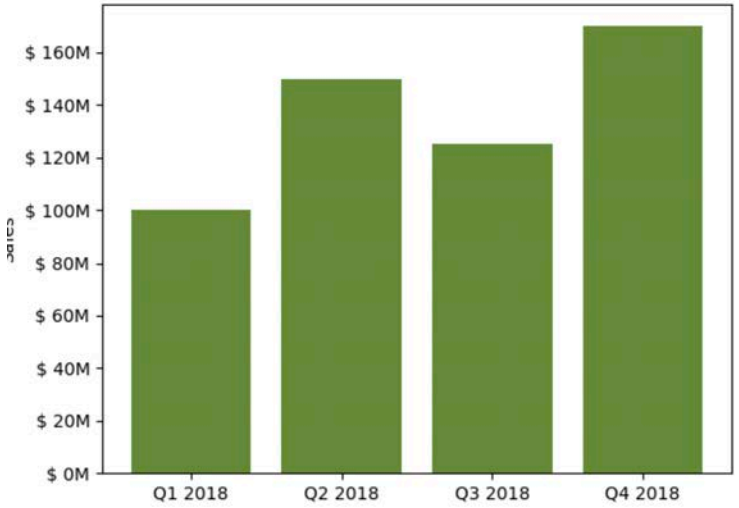
- Install matplotlib
- Prepare data
- Create bar graph
- Display the chart

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

Charting

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

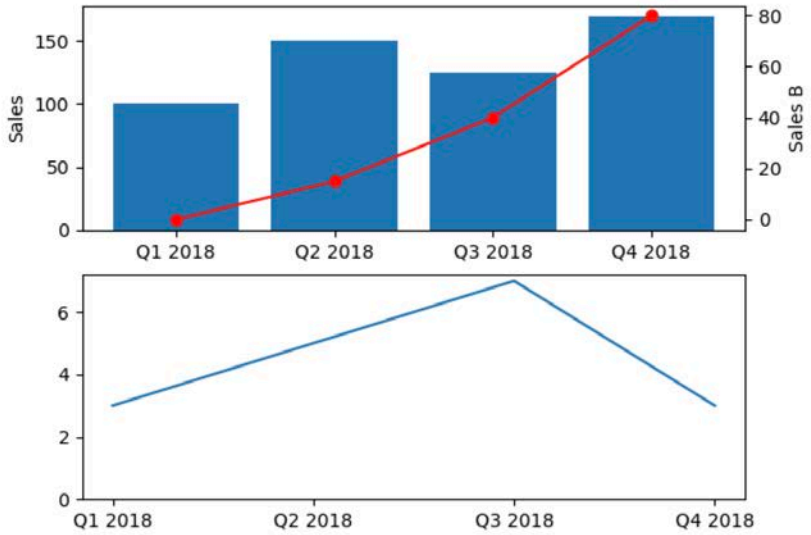
- Install matplotlib
- Prepare data
- Customise graph options
- Create bar graph
- Display the chart



Charting

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

- Multiple charts

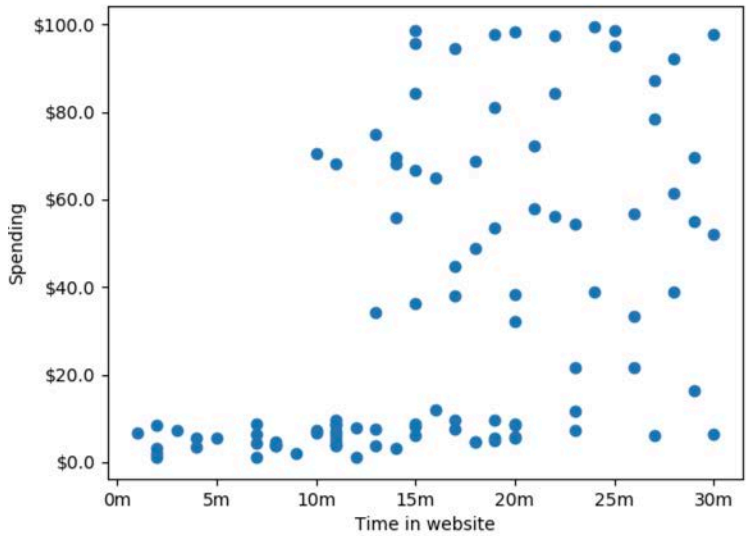


https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html

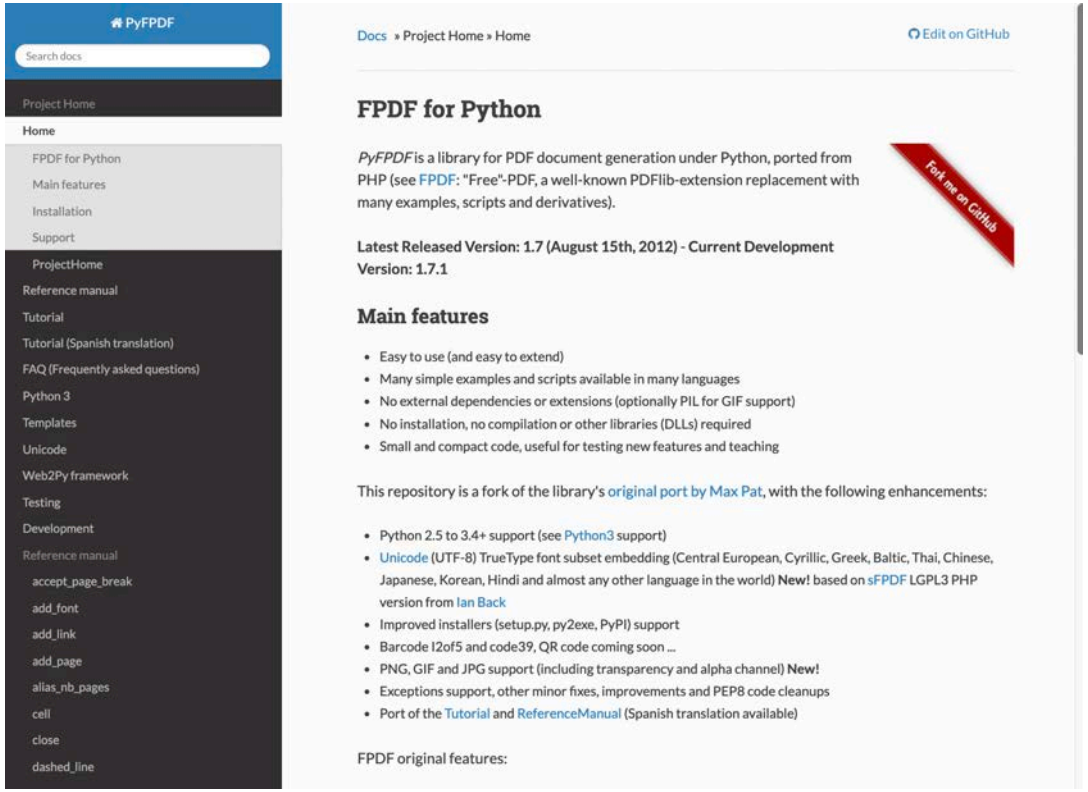
Charting – Scatter Plot

```
1   import csv
2   import matplotlib.pyplot as plt
3   from matplotlib.ticker import FuncFormatter
4
5   def format_minutes(value, pos):
6           return '{}m'.format(int(value))
7
8   def format_dollars(value, pos):
9           return '${}'.format(value)
10
11  # read data from csv
12  fp = open("scatter.csv","r", newline='')
13  reader = csv.reader(fp)
14  data = list(reader)
15
16  data_x=[]
17  data_y=[]
18  for x, y in data:
19          data_x.append(float(x))
20          data_y.append(float(y))
21
22  plt.scatter(data_x, data_y)
23
24  plt.gca().xaxis.set_major_formatter(FuncFormatter(format_minutes))
25  plt.xlabel('Time in website')
26  plt.gca().yaxis.set_major_formatter(FuncFormatter(format_dollars))
27  plt.ylabel('Spending')
28
29  plt.show()
```

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



PDF



- Install fpdf
  - pip install fpdf

PDF – Basic document

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

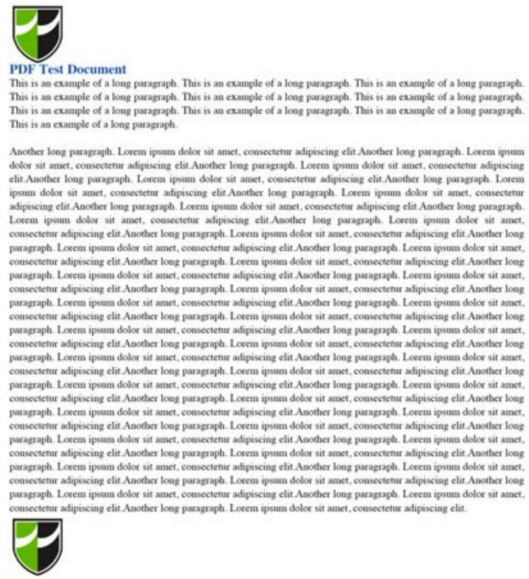
- Import fpdf
- Create a new pdf document
- Add page
- Add text
- Save file

https://pyfpdf.readthedocs.io/en/latest/reference/image/index.html

PDF – adding images

```python
import fpdf

#create a new pdf
document = fpdf.FPDF()

#define font and color for title and add the first page
document.set_font("Times","B", 14)
document.set_text_color(19,83,173)
document.add_page()

#add a image
document.image("rp_logo.png", x=10, y=8, w=23)
document.set_y(30);

#write the title of the document
document.cell(0,5,"PDF Test Document")
document.ln()

#write a long paragraph
document.set_font("Times", "", 11)
document.set_text_color(0)
document.multi_cell(0,5, "This is an example of a long paragraph. " * 10)
document.ln()

#write another long paragrahp
document.multi_cell(0,5, "Another long paragraph. \
Lorem ipsum dolor sit amet, consectetur adipiscing elit." * 40)

#add another image
document.image("rp_logo.png", w=23)

#save the document
document.output("pdf_report.pdf")
```

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


PDF Test Document

This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph. This is an example of a long paragraph.

Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. ... Another long paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

PDF – Adding password

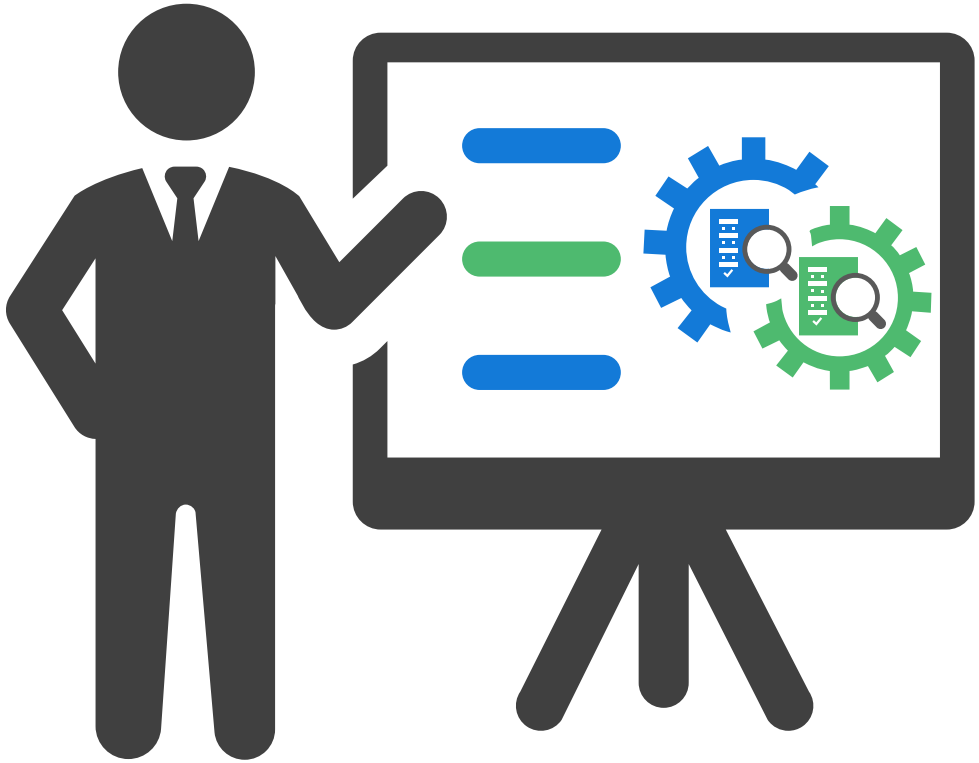
```
1  import fpdf
2  import PyPDF2
3
4  #create a new pdf
5  document = fpdf.FPDF()
6
7  #define font and color for title and add the first page
8  document.set_font("Times","B", 14)
9  document.set_text_color(19,83,173)
10 document.add_page()
11
12 #add a image
13 document.image("rp_logo.png", x=10,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

- pip install PyPDF2

https://pythonhosted.org/PyPDF2/

PSA UNIVERSITY

Day 2 Summary



- ✓ *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*
- ✓ *Telegram bot*

Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: http://bit.ly/2Xl57hB

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

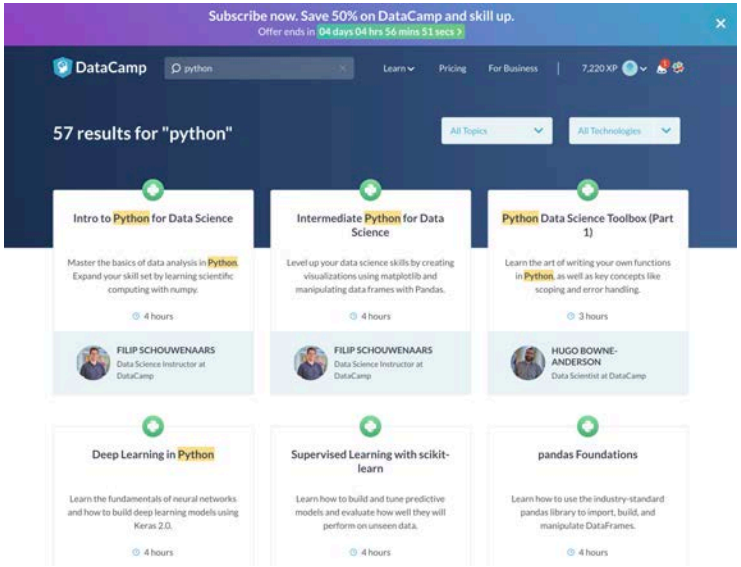
Advanced examples:
http://www.diveintopython.org/toc/index.html

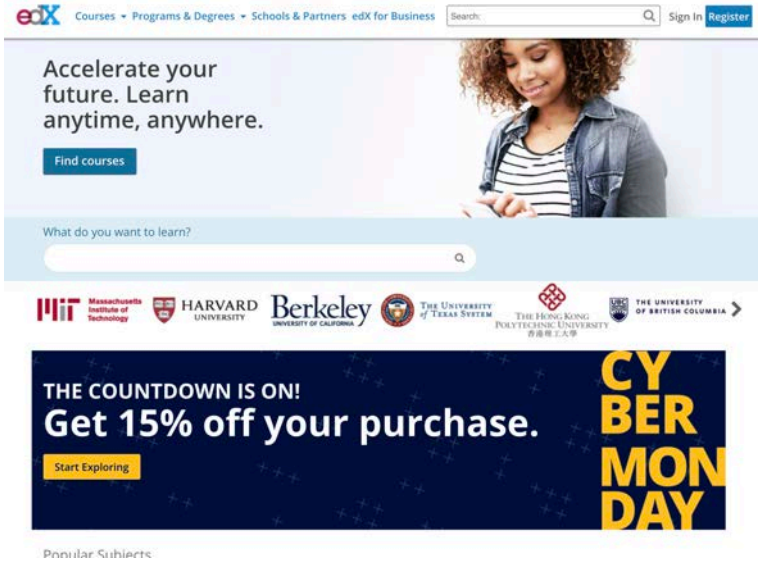
Where to go from here?

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

Edx
https://www.edx.org/

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





Lifelong Learning

Scan me

- https://www.rp.edu.sg/soi/lifelong-learning

Short Courses

SOI offers an extensive variety of short, industry-relevant courses for ICT skills upgrading and skills acquisition. Our courses are categorized under different areas, ranging from Artificial Intelligence (AI), Business Intelligence/ Business Analytics (BI/BA), Business Processes (BP), Unmanned Aerial Vehicle (UAV), IT Security, New/ Digital Media, Software Development to the Internet of Things (IoT). To view our short course offerings, click on the relevant tab below.

AI | Data Analytics | IT Security | DevOps | Software Development | New/ Digital Media | UAV | RPA

+ Artificial Intelligence for Everyone - A Practical Experience (1 day Beginner)

+ Artificial Intelligence for Techies - A Hands-On Approach (1 day Beginner)

+ An Introduction to Code-Free Machine Learning (1 day Beginner)

Thank you