



Python CET Course

Day 2

By Tan Kok Cheng & Seow Khee Wei / Republic Polytechnic

Programme Day Two

Morning

- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Processing CSV files
- Generating PDF

Afternoon

- Image processing: loading, scaling, watermark
- Creating charts
- Connecting to the Web
- Sending emails
- Telegram bot

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

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*.

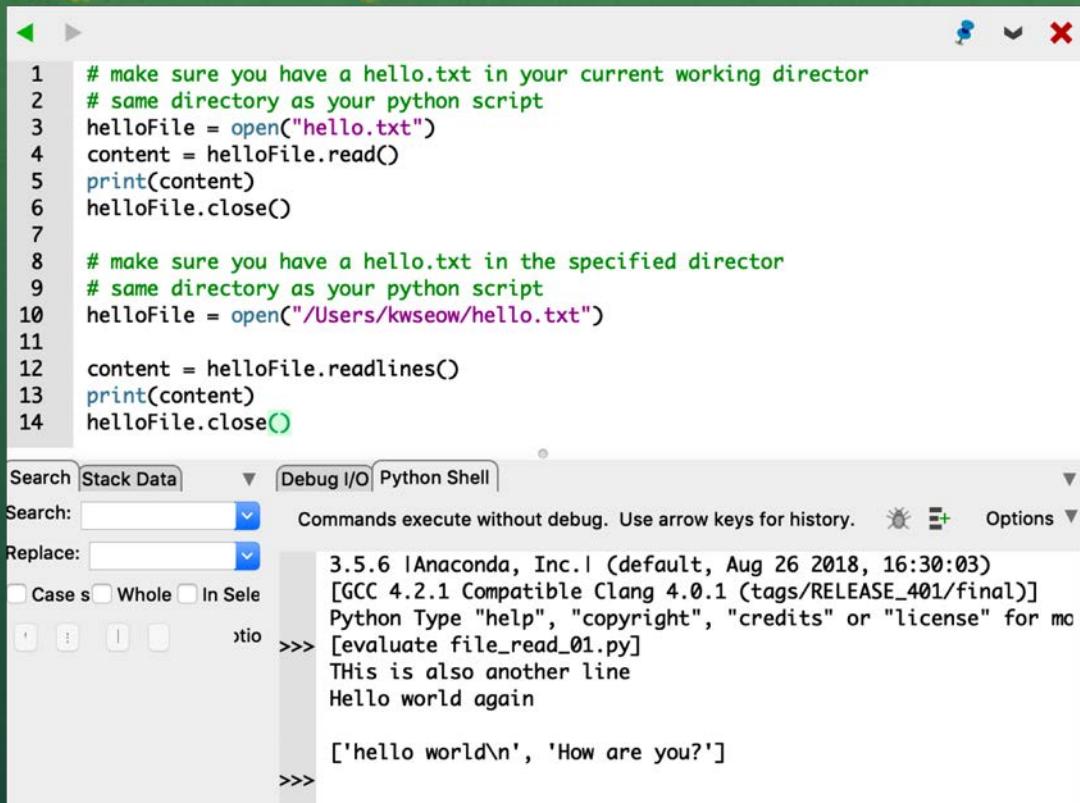
Relative file paths are notated by a lack of a leading forward slash.

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



The screenshot shows a Python IDE interface with a code editor and a Python Shell tab. The code in the editor is:

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

The Python Shell tab shows the output of running the script:

```
3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" for more
information
>>> [evaluate file_read_01.py]
THis is also another line
Hello world again

['hello world\n', 'How are you?']
```

- 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.

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 char_written = helloFile.write("THis is also another line\n")
5 print("No of character written:",char_written)
6 helloFile.close()
7
8 # reopen to display content
9 helloFile = open("hello.txt")
10 print(hellofile.read())
11 helloFile.close()
12
13 # open the file for adding next text
14 helloFile = open("hello.txt", "a")
15 char_written = helloFile.write("Hello world again\n")
16 print("No of character written:",char_written)
17 helloFile.close()
18
19 # reopen to display content
20 helloFile = open("hello.txt")
21 print(hellofile.read())
22 helloFile.close()
```

Search Stack Data Debug I/O Python Shell

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

Replace: Options

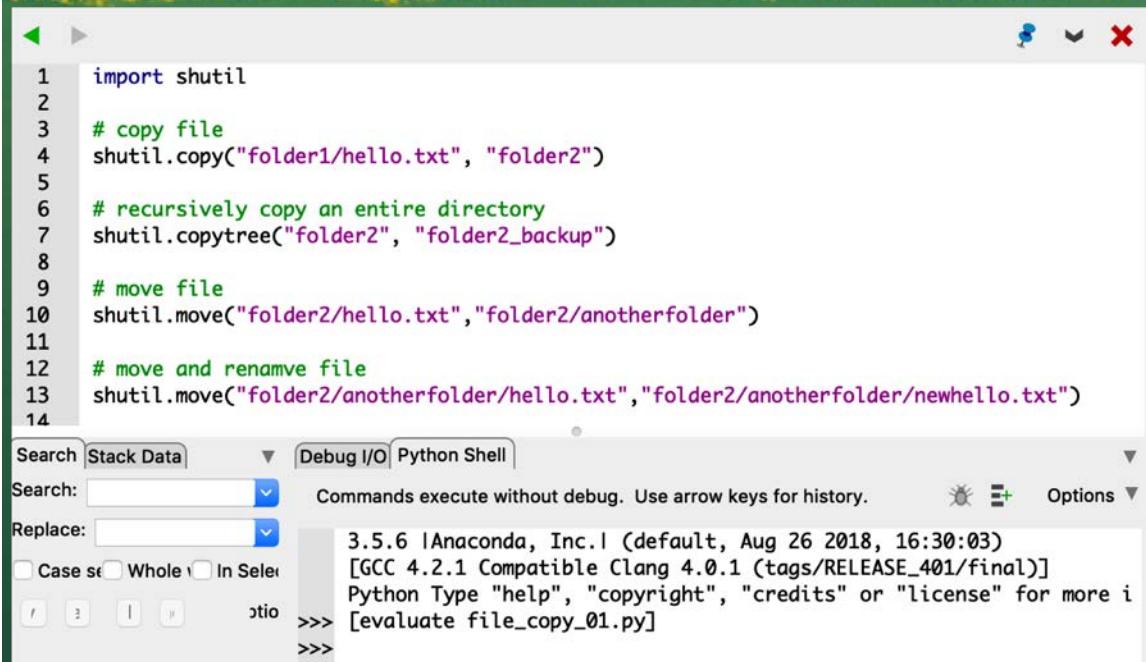
Case s Whole In Sele

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

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

- 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.

Copy and moving files



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

The screenshot shows a Python code editor window with a script named `file_copy_01.py`. The code demonstrates the use of the `shutil` module for file and directory operations. It includes examples for copying files, recursively copying entire directories, moving files, and renaming files.

Below the code editor is a Python Shell interface. The shell shows the following environment information:

3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Python Type "help", "copyright", "credits" or "license" for more information

The shell history shows the command `>>> [evaluate file_copy_01.py]` being run.

- `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*).

Deleting files

The screenshot shows a Python development environment. On the left, a code editor displays a script named `file_delete_01.py` with the following content:

```
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")
```

On the right, a terminal window titled "Python Shell" shows the output of running the script:

```
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
```

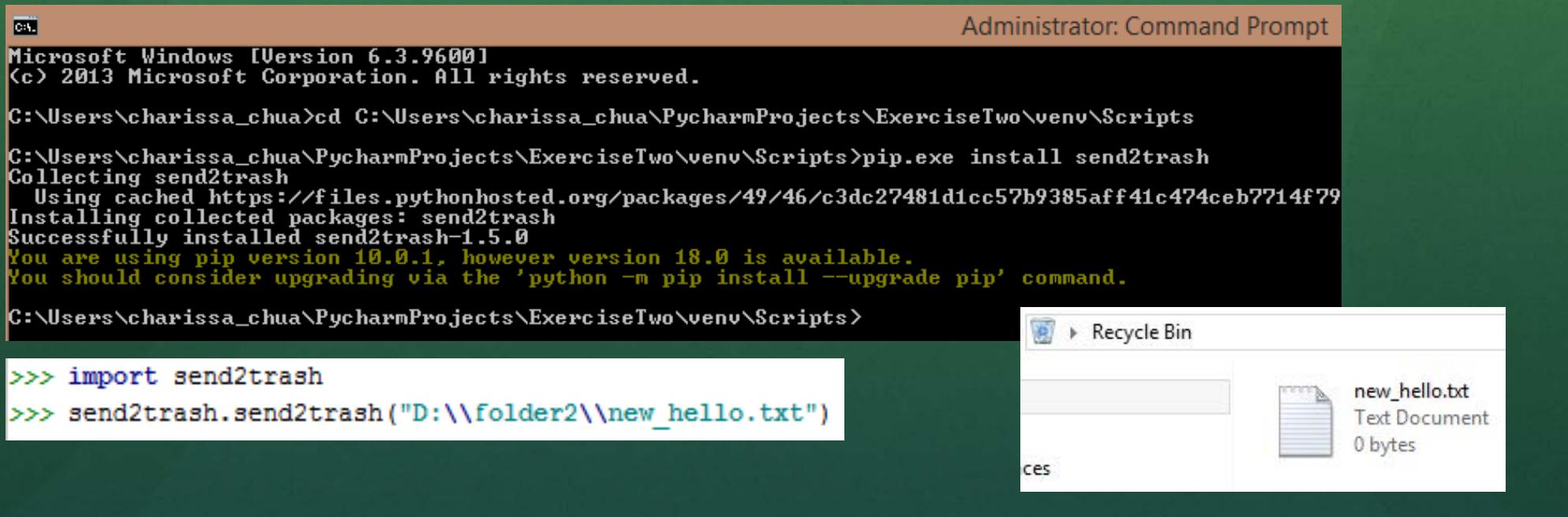
Below the terminal window, another code editor window titled "RemoveFiles.py" shows the following code:

```
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)
```

- `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

send2trash module

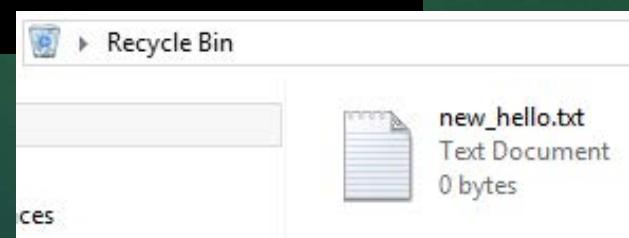
- Install send2trash module using pip.exe
- send2trash.send2trash() will send a file or folder to the recycling bin



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

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

C:\Users\charissa_chua\PycharmProjects\ExerciseTwo\venv\Scripts>
>>> import send2trash
>>> send2trash.send2trash("D:\\\\folder2\\\\new_hello.txt")
```



Walk a Directory to perform some tasks

```
FileUtils.py x

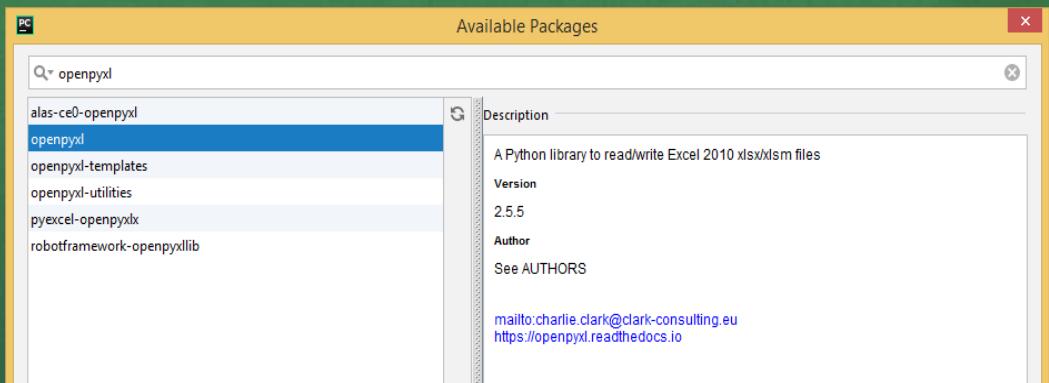
1 import os
2 import shutil
3
4 for folderName, subfolders, filenames in os.walk("C:\\\\Users\\\\charissa_chua\\\\Downloads"):
5     print("The folder is " + folderName)
6     print("The subfolders in " + folderName + " are: " + str(subfolders))
7     print("The filenames in " + folderName + " are: "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

Remove all folders with “c235” in them

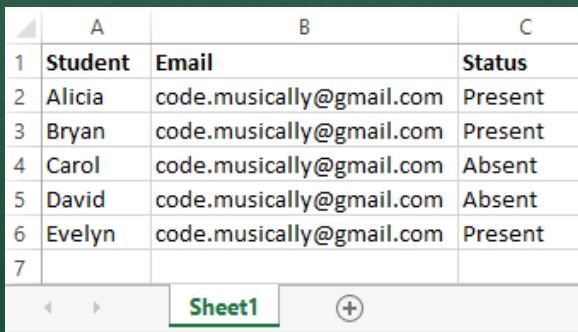
Back up all jpeg files with file extension
“.backup”

Working with Excel

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



The screenshot shows the PyCharm interface with the "Available Packages" dialog box open. The search bar at the top contains the text "openpyxl". Below the search bar, a list of packages is displayed, with "openpyxl" highlighted. To the right of the list, detailed information about the selected package is shown, including its description as "A Python library to read/write Excel 2010 xlsx/xlsm files", version 2.5.5, author Charlie Clark, and links to the AUTHORS file and documentation.



The screenshot shows an Excel spreadsheet titled "Sheet1". The data is organized in a table with columns labeled "A", "B", and "C". The rows contain the following data:

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

Working with Excel - Reading

```
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")
```

- 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

Working with Excel - Update

```
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")
```

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

Working with Excel - Create

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

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

```

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

```

Working with Excel - Format

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

Working with CSV File

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

What is CSV format?

- The same data when viewed with Excel ...

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

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

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

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

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

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

Reading from a CSV File

1) Load CSV library

```
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()
```

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

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

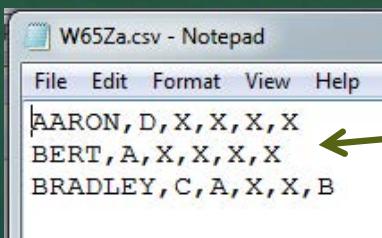
5) Close the file (remove the link)

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)

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

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

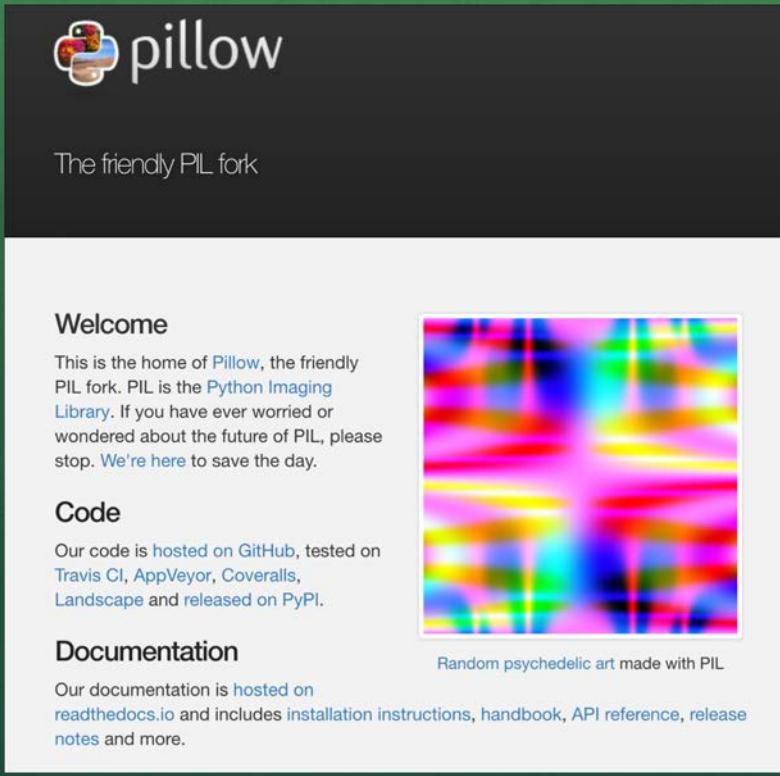
Writing to a CSV File

```
1) Load CSV library  
import csv  
with open("W65z_new.csv", "w", newline='') as writerFileHandle:  
    writer1 = csv.writer(writerFileHandle)  
  
    row1 = ['AARON', 'D', 'X', 'X', 'X', 'X']  
    row2 = ['BERT', 'A', 'X', 'X', 'X', 'X']  
    row3 = ['BRADLEY', 'C', 'A', 'X', 'X', 'B']  
    rowlist = [row1, row2, row3]  
  
    for row in rowlist:  
        writer1.writerow(row)  
  
writerFileHandle.close()  
  
2) Create (if new) & Open the file named "W65z_new.csv" for writing  
(indicated by 'w' argument). writerFileHandle links the file to the  
program.  
  
3) writer1 stores content to be  
written to the file in CSV format  
  
4) rowlist stores the content to  
be written to the CSV file.  
(rowlist is a list containing lists  
as items)  
  
5) For-loop retrieves each item  
from rowlist into loop variable  
row → row (a list) is written as 1  
csv formatted line into the file.  
  
6) Close the file (Remove the  
link)  
  


|   | AARON   | D | X | X | X | X |
|---|---------|---|---|---|---|---|
| 1 | BERT    | A | X | X | X | X |
| 2 | BRADLEY | C | A | X | X | B |


```

Image processing



The screenshot shows the official website for Pillow, the Python Imaging Library fork. The header features the Pillow logo (a stylized 'P' made of colorful pixels) and the text "pillow" in white. Below the logo is the tagline "The friendly PIL fork". The main content area has a dark background with white text. It includes sections for "Welcome", "Code", and "Documentation". The "Welcome" section contains a paragraph about the project's history and a link to "We're here". The "Code" section links to GitHub and Travis CI. The "Documentation" section links to readthedocs.io. To the right of the text is a small, colorful abstract image titled "Random psychedelic art made with PIL".

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

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

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

Random psychedelic art made with PIL

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

You can install this package directly in PyCharm.

Alternatively, run the command:
`pip install Pillow`

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

Image processing

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

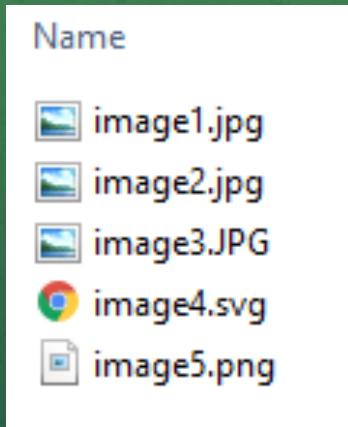
Before we start processing images, we need to be able to loop through all the images.

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

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 on the left

Image processing



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

Process finished with exit code 0
```

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(...)`

Image processing

```
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()
```

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.

Image processing

```
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)
```

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

```
C:\Users\denise_quek\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/denise_quek/Desktop/Day2/day2.py
1 .\Day2Resource\img\image1.jpg
Process finished with exit code 0
```

Image processing

```
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)
```

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 = ".\\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)
```

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.

```
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

```
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)
```

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:

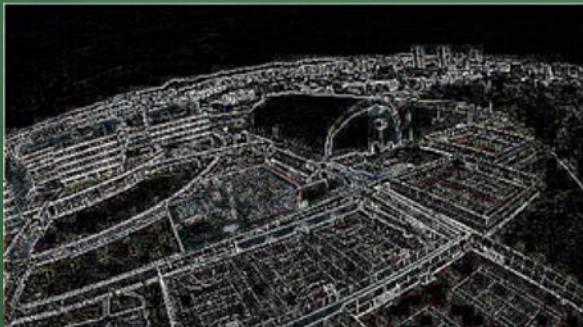
from PIL import Image, ImageFilter

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

* Remember to include
ImageOps in your import statement

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



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
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)
```

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.

Image processing - writing

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

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.

Image processing - writing

```
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)
```

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.

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 holiday.gif to holiday.jpg ?

Let's open the real time Interpreter

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
>>>
```

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

Change the outFileNmae 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.

Image processing - converting

```
1 import os
2 from PIL import Image
3
4 where = ".\\Day2Resource\\img\\"
5
6 def convertImages():
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 file.lower().endswith("bmp")):
11                im = Image.open(fullname)
12                f, e = os.path.splitext(file)
13                fname2 = f + ".jpg"
14                outfilename = os.path.join(where, fname2)
15                print(outfilename)
16                im.save(outfilename, "jpeg")
17
18 convertImages()
```

You can shorten those 3 lines into one by :

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

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

Image processing - watermark

Create the mark image

You can reduce the size to 100,100

Create a new function called

`def watermark(im, mark, position):`

....

It takes the original image, the watermark image and the desired position that we want the watermark to appear.

The function will return the result.

We can use this function like:

`watermark(im, mark, (0, 50)).show()`

or

`imOut = watermark(im, mark, (0,50))
imOut.save(fileOut)`

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

Copyright
@RP

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

```
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()
```

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:

Charting

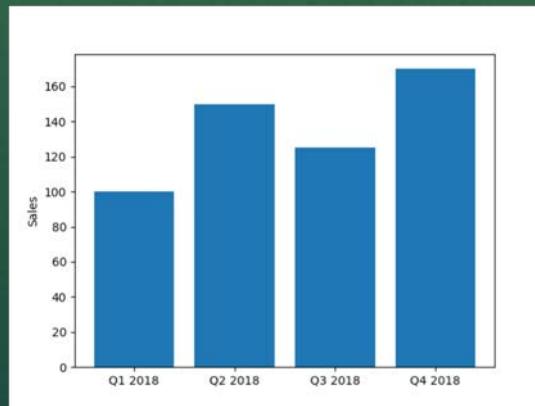
The screenshot shows the matplotlib.org/gallery page. At the top left is the matplotlib logo with "Version 3.0.2". Below it is a navigation bar with links: home | examples | tutorials | API | docs. On the right side of the header is a "Fork me on GitHub" button. The main content area is titled "Gallery" and contains a brief description: "This gallery contains examples of the many things you can do with Matplotlib. Click on any image to see the full image and source code." It also links to the "tutorials page", "external resources", and "FAQ" in the user guide. Below this, there are two rows of four chart examples each. The first row includes: "Arctest", "Stacked Bar Graph", "Barchart", and "Horizontal bar chart". The second row includes: "Broken Barh", "Plotting categorical variables", "Plotting the coherence of two signals", and "CSD Demo". Each example has a small thumbnail image and a caption below it. At the bottom of the page is a footer with the URL "https://matplotlib.org/index.html" and a note: "Small changes by better approaches".

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

```

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

```



Charting

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

Charting

- Multiple charts

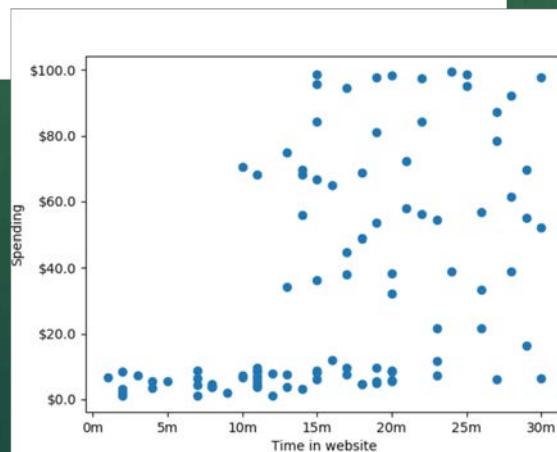


https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html

```
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()
```

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



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

PDF

The screenshot shows the GitHub project page for PyFPDF. The header includes the repository name "PyFPDF" and a search bar. The main content area has a title "FPDF for Python" and a brief description: "PyFPDF is a library for PDF document generation under Python, ported from PHP (see FPDF: "Free"-PDF, a well-known PDFlib-extension replacement with many examples, scripts and derivatives)." Below this is a "Main features" section with a bulleted list:

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

Further down, there's a note about the repository being a fork of the original port by Max Pat, followed by a list of enhancements:

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

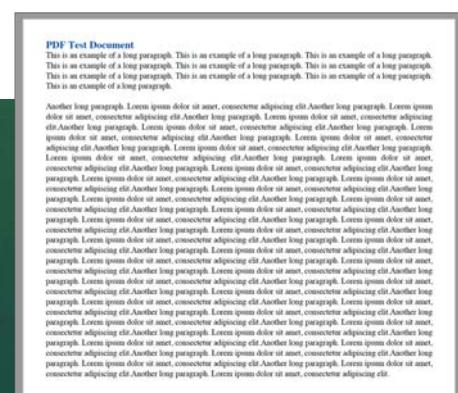
At the bottom, it lists "FPDF original features:" which include: accept_page_break, add_font, add_link, add_page, alias_nb_pages, cell, close, and dashed_line.

- Install fpdf
 - pip install fpdf

PDF

- Install fpdf
 - pip install fpdf

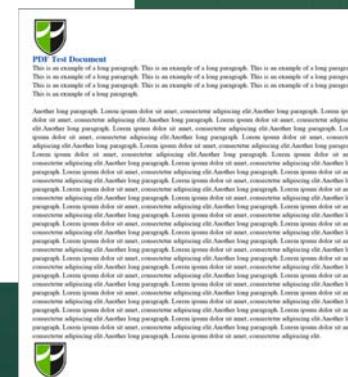
```
1 import fpdf
2
3 #create a new pdf
4 document = fpdf.FPDF()
5
6 #define font and color for title and add the first page
7 document.set_font("Times", "B", 14)
8 document.set_text_color(19,83,173)
9 document.add_page()
10
11 #write the title of the document
12 document.cell(0,5,"PDF Test Document")
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")
```



PDF – adding images

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

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



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

PDF – adding password

- pip install PyPDF2

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

<https://pythonhosted.org/PyPDF2/>

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
```



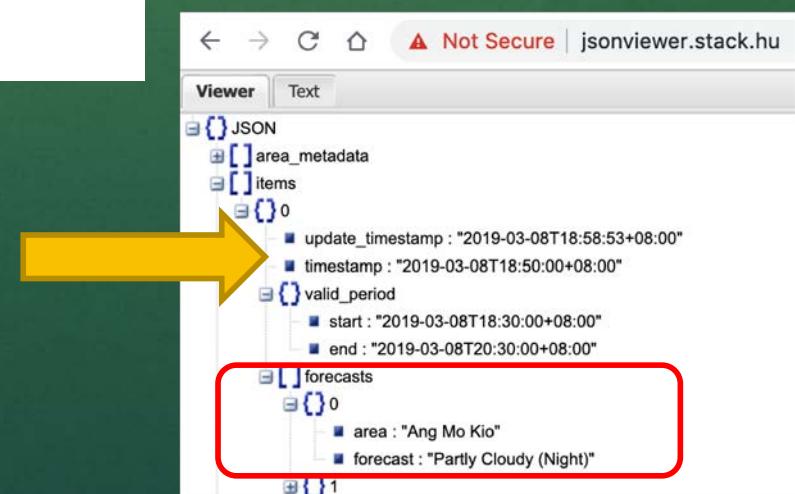
```
downloadedFile.txt
1 {"area_metadata": [{"name": "Ang Mo Kio", "label_location": {"latitude": 1.375, "longitude": 103.839}}, {"name": "Bedok", "label_location": {"latitude": 1.321, "longitude": 103.924}}, {"name": "Bishan", "label_locat
```

Connecting to the Web

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

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

```
{"area_metadata": [{"name": "Ang Mo Kio",  
"label_location": {"latitude": 1.375, "longitude":  
103.839}}, {"name": "Bedok", "label_location": {  
"latitude": 1.321, "longitude": 103.924}}, {"name":  
"Bishan", "label_location": {"latitude": 1.350772,  
"longitude": 103.839}}, {"name": "Boon Lay",  
"label_location": {"latitude": 1.304, "longitude":  
103.701}}, {"name": "Bukit Batok",  
"label_location": {"latitude": 1.31, "longitude":  
103.839}}], "valid_period": {  
"start": "2019-03-08T18:30:00+08:00",  
"end": "2019-03-08T20:30:00+08:00"}, "forecasts": [  
{"area": "Ang Mo Kio", "forecast": "Partly Cloudy (Night)"}]}
```



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)
```



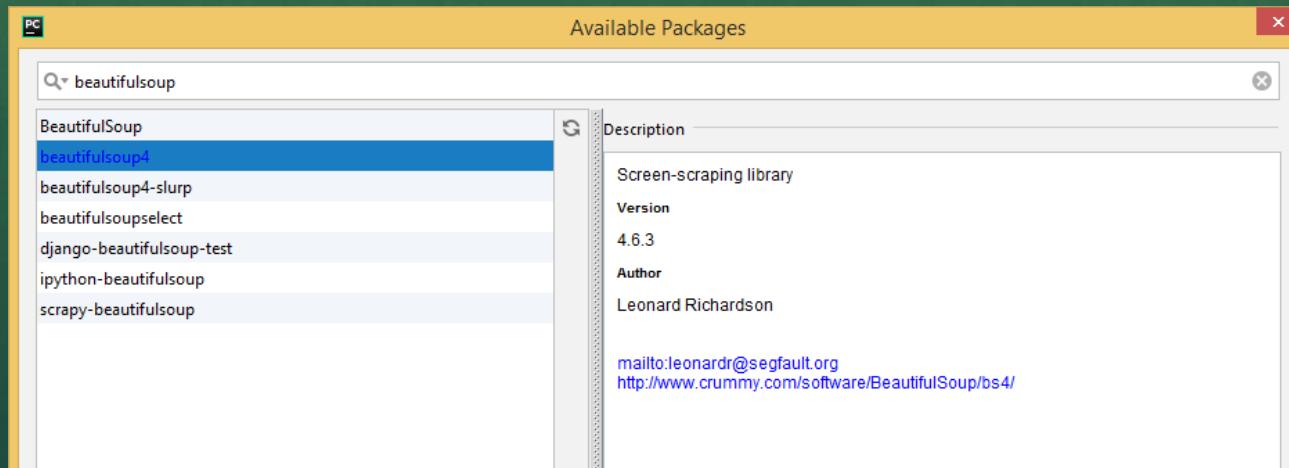
```
C:\Users\denise_quek\AppData\Local\Programs\Python\Py
Ang Mo Kio: Thundery Showers
Bedok: Thundery Showers
Bishan: Heavy Thundery Showers with Gusty Winds
Boon Lay: Heavy Thundery Showers with Gusty Winds
Bukit Batok: Heavy Thundery Showers with Gusty Winds
Bukit Merah: Heavy Thundery Showers with 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 - pip install beautifulsoup4



Connecting to the Web

- What's the URL?

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

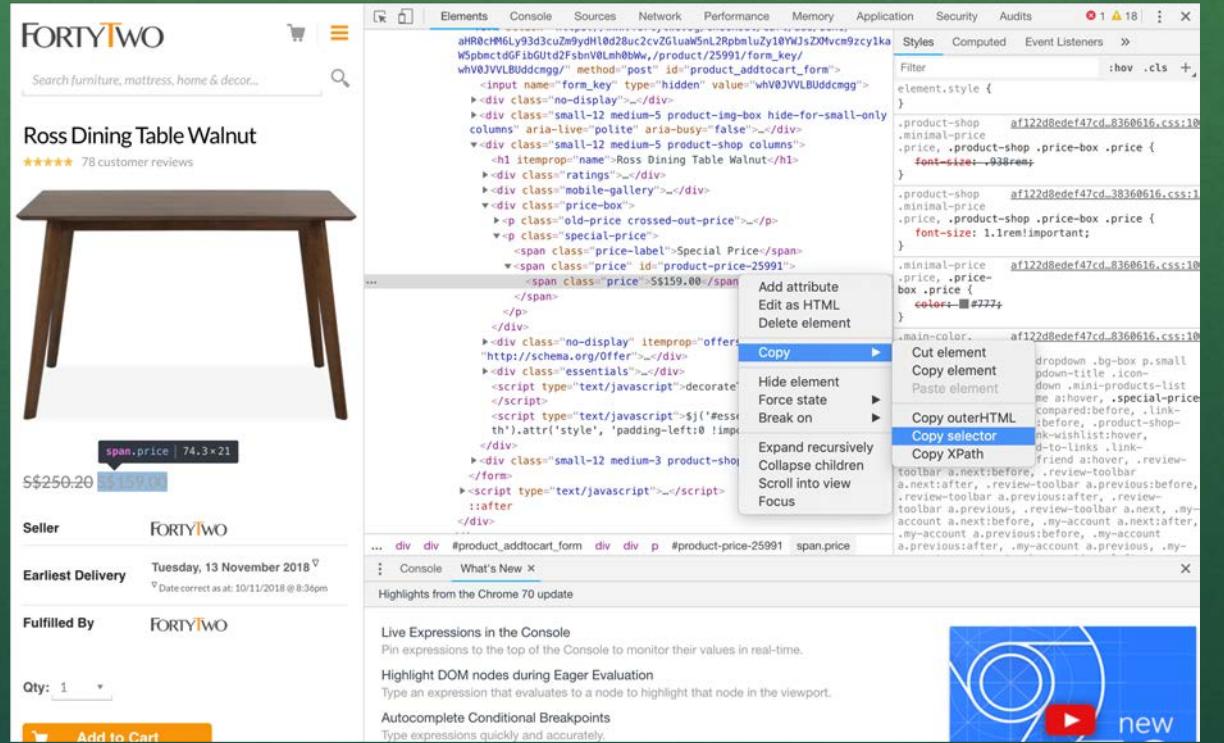
The screenshot shows the FortyTwo website's product page for the Ross Dining Table Walnut. At the top, there is a navigation bar with links for Enquiries, Mon-Fri (10am-6pm), Contact Us, About Us, Furniture, Bedding & Mattresses, Décor | Essentials, Kitchen | Dining, Inspirations, Sale, Hello, Sign in Your Account, and a Cart icon. The main content area features a large image of the dark wood dining table with a rectangular top and four tapered legs. Below the image, there is a smaller image showing five different dining table models. The product title "Ross Dining Table Walnut" is displayed, along with a star rating of 4.5 stars from 78 customer reviews and a price of S\$250.20 reduced to S\$159.00. The seller is listed as FORTYTWO, with an earliest delivery date of Tuesday, 13 November 2018. The item is fulfilled by FORTYTWO. On the right side, there is a "Qty: 1" dropdown, an "Add to Cart" button, an "Add to Wishlist" link, an "Email to a Friend" link, and icons for 100 Day Free Returns and Free Assembly Included.

Connecting to the Web

- Get the url

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

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



Connecting to the Web

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

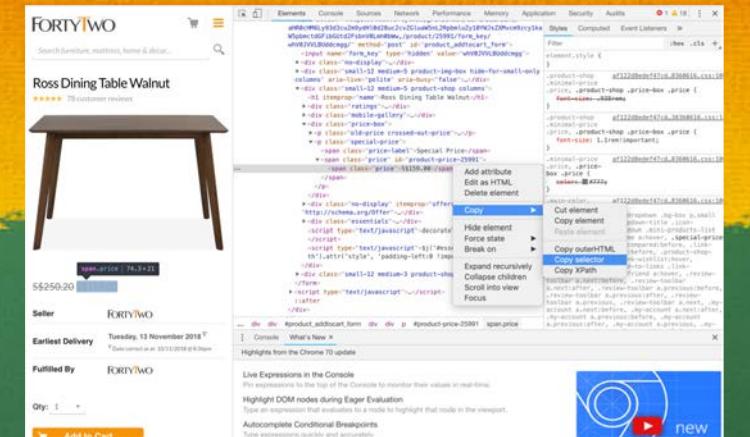
```
import bs4
import requests

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)
```

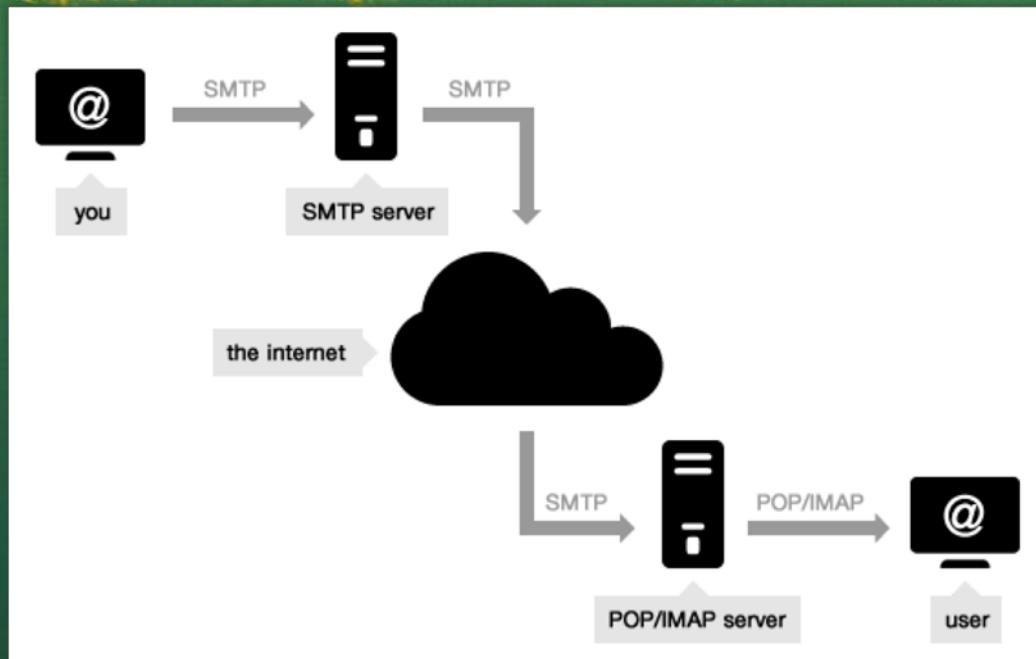


```
C:\Users\kwseow\PycharmProjects\PSA
S$159.00

Process finished with exit code 0
```



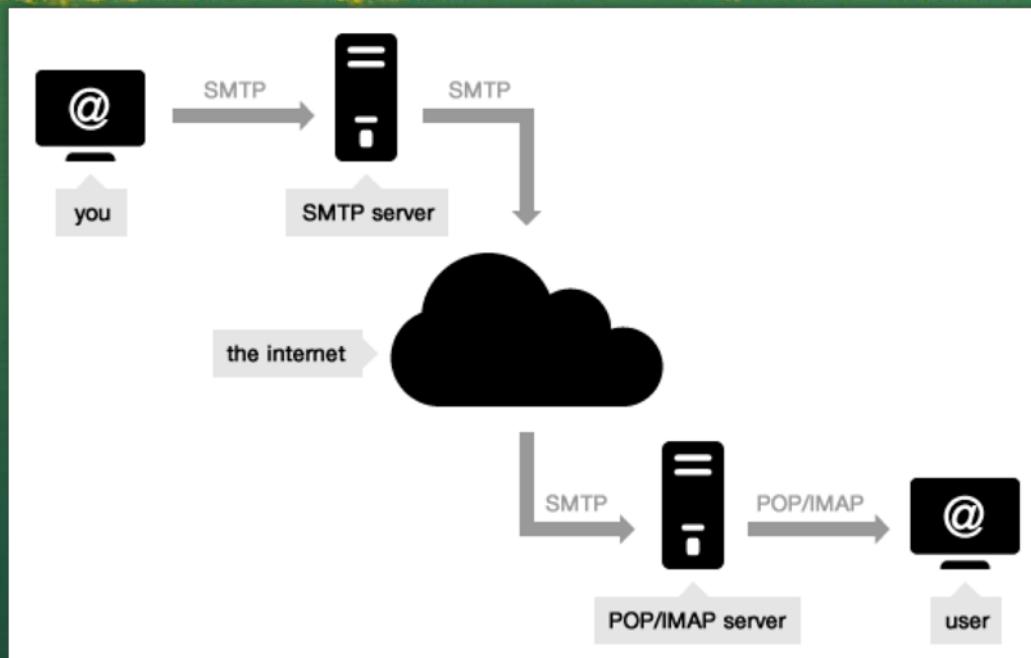
Send Email



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

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

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

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

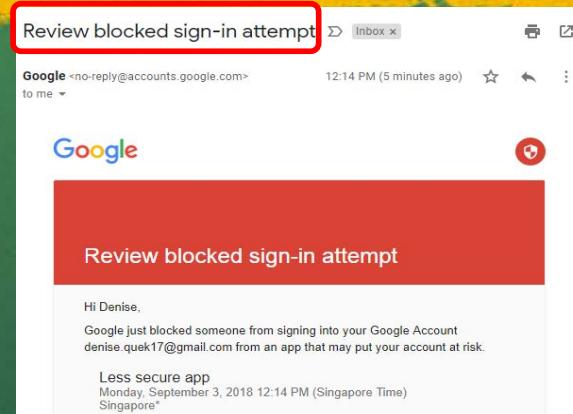
print("Connected. Logging in...")
smtpObj.login(sender_email_address, sender_email_password)

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

smtpObj.quit()
```

Send Email using Gmail

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



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

Process finished with exit code 1
```

A screenshot of a terminal window showing Python code execution. The command run was "C:\Users\denise_quek\AppData\Local\Programs\Python\Python37\python.exe D:/CET_Python/Denise/TestEmail.py". The output shows the program connecting to the Gmail SMTP server and attempting to log in. It then displays a traceback for a "SMTPAuthenticationError" at line 13 of TestEmail.py. The error message is "(534, b'5.7.9 Application-specific password required. Learn more at\n5.7.9')". A red box highlights the error message.

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

App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Mail Windows Computer

GENERATE

Generated app password

Your app password for Windows Computer

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

[Learn more](#)

DONE

How to use it

1. Open the "Mail" app.
2. Open the "Settings" menu.
3. Select "Accounts" and then select your Google Account.
4. Replace your password with the 16-character password shown above.

Send Email using Gmail

- Replace your actual password with the App password

```
import smtplib

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

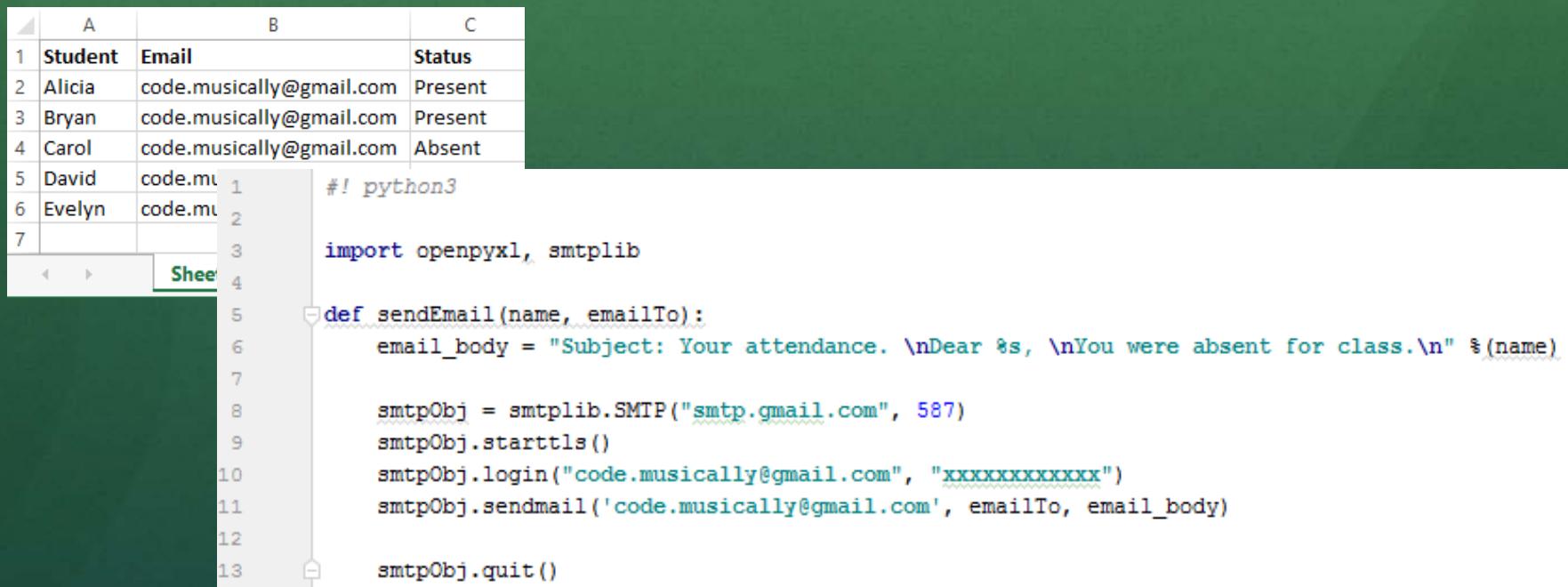
- Run your email program

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

Process finished with exit code 0
```

Send Email using Gmail

- Send email to students who were absent



The image shows a screenshot of a spreadsheet and a Python script. The spreadsheet has columns for Student, Email, and Status. The data is as follows:

	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			

The Python script is as follows:

```
#! 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", "xxxxxxxxxxxx")
    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_password"
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

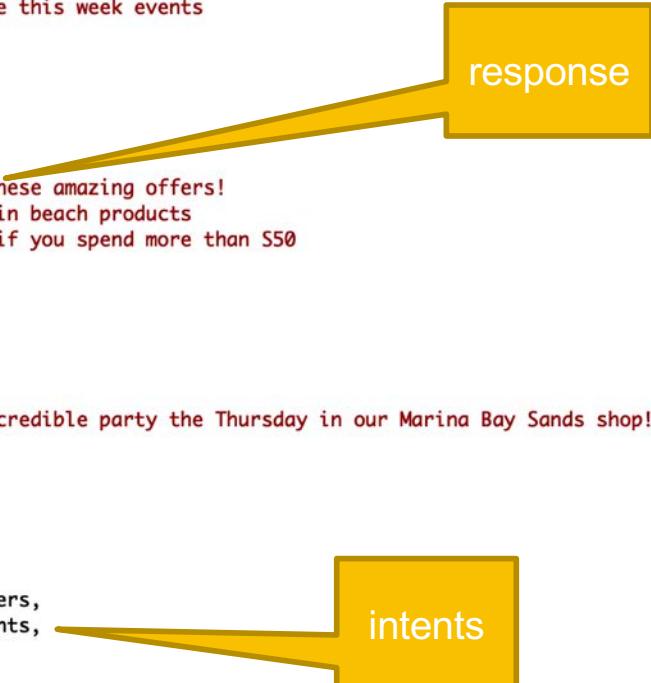


- Create a new bot using BotFather: <https://telegram.me/botfather>
- Run /start to start the interface and then create a new bot with /newbot
- The interface will ask you the name of the bot and a username, which should be unique
- The Telegram channel of your bot—<https://t.me/<yourusername>>
- A token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

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



Telegram Bot

Process the message

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

Telegram Bot



Telegram Bot – Custom Keyboard



Telegram Bot – Custom Keyboard

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

Telegram Bot – Custom Keyboard

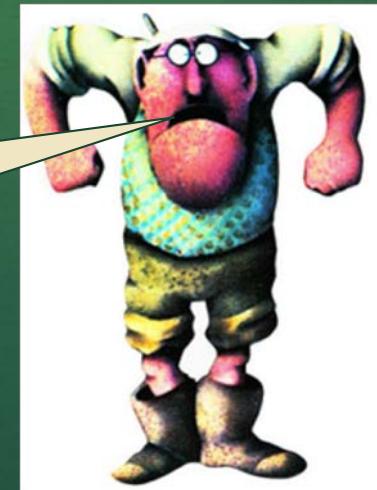
```
45 class MarketingBot(telepot.helper.ChatHandler):
46
47     def open(self, initial_msg, seed):
48         self.sender.sendMessage(get_help(), reply_markup=KEYBOARD)
49         # prevent on_message() from being called on the initial message
50         return True
51
52     def on_chat_message(self,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

End of Day 2

This concludes the Introduction to Python,
I hope you enjoyed it.

Thank you !

QUESTIONS ?



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:

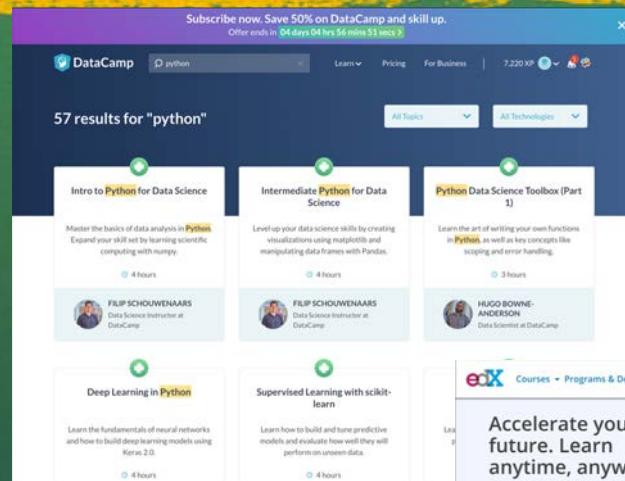
<http://docs.python.org/library/index.html>

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>

Where to go from here ?



Think Python is an introduction to Python programming for beginners. It starts with basic concepts of programming, and is carefully designed to define all terms when they are first used and to develop each new concept in a logical progression. Larger pieces, like recursion and object-oriented programming are divided into a sequence of smaller steps and introduced over the course of several chapters.

Think Python is a Free Book. It is available under the [Creative Commons Attribution-NonCommercial 3.0 Unported License](#), which means that you are free to copy, distribute, and modify it, as long as you attribute the work and don't use it for commercial purposes.

<http://greenteapress.com/thinkpython/thinkpython.pdf>

Contact

tan_kok_cheng@rp.edu.sg

[seow khee wei@rp.edu.sg](mailto:seow_khee_wei@rp.edu.sg)

@kwseow

Github: <http://bit.ly/2HwB1m4>

What's next?

**Software Bots
for
Conversation
(3 days)**



**Introduction to
AI-based
Recommender
Systems using
Python and
TensorFlow
(3 days)**



**Data Science
with Python
(2 days)**



Online Evaluation
(SF) Introductory Programming using Python
(21 - 22 Mar 2019)



<https://bit.ly/2TDpxnf>