

# Introductory Programming Using Python

---

## Day 2

By Jason Lim

Republic Polytechnic



# Administrative matters

---



The instructor is **busy** taking  
attendance ☺

Perform software installation (if haven't done so):

- Python 3.8.x
- Wing IDE

Course Material Link: <https://bit.ly/py-78oct2021>



# Trainers

---

## Day 1

**Tay Mei Lan**

**tay\_mei\_lan@myrp.edu.sg**



## Day 2

**Jason Lim**

**jason\_lim@rp.edu.sg**





# Programme Day Two

---

Morning	Afternoon
<ul style="list-style-type: none"><li>• Read and writing files</li><li>• Copying, moving and deleting files and folders</li><li>• Working with Excel</li><li>• Image Processing</li></ul>	<ul style="list-style-type: none"><li>• Connecting to the Web</li><li>• Sending emails</li><li>• Generating PDF</li></ul>



# Outline for the day

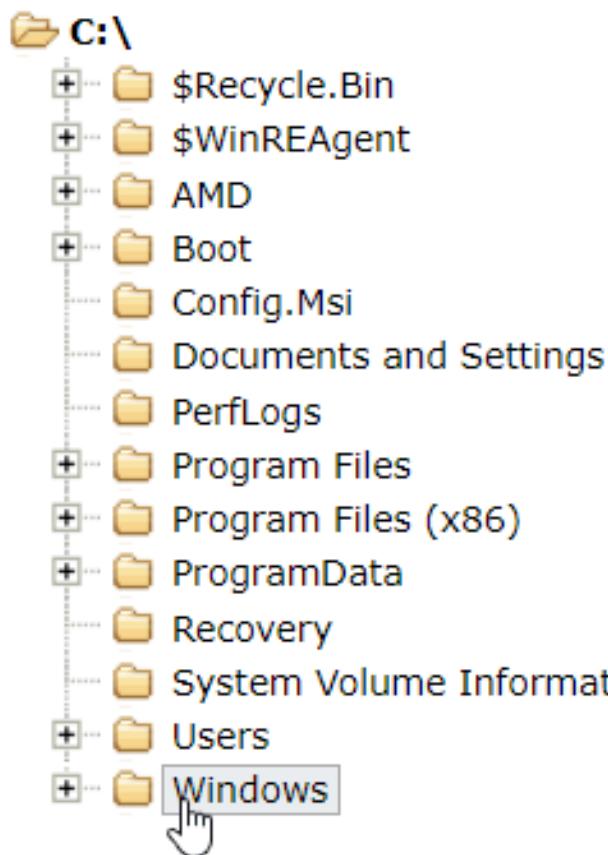
---

Time	Agenda
9.00am	Welcome and admin matters
9.15am – 10.30am	<ul style="list-style-type: none"><li>• Read and writing files</li><li>• Copying, moving and deleting files and folders</li></ul>
10.30am – 10.45am	Break
10.45am – 12.30pm	<ul style="list-style-type: none"><li>• Working with Excel</li><li>• Image Processing</li></ul>
12.30pm – 1.30pm	Lunch
1.30pm – 3.15pm	<ul style="list-style-type: none"><li>• Connecting to the Web</li><li>• Sending emails</li></ul>
3.15pm – 3.30pm	Break
3.30pm – 4.30pm	<ul style="list-style-type: none"><li>• Creating Chart</li><li>• Generating PDF</li></ul>
4.45pm – 5.00pm	Wrap up, Q&A

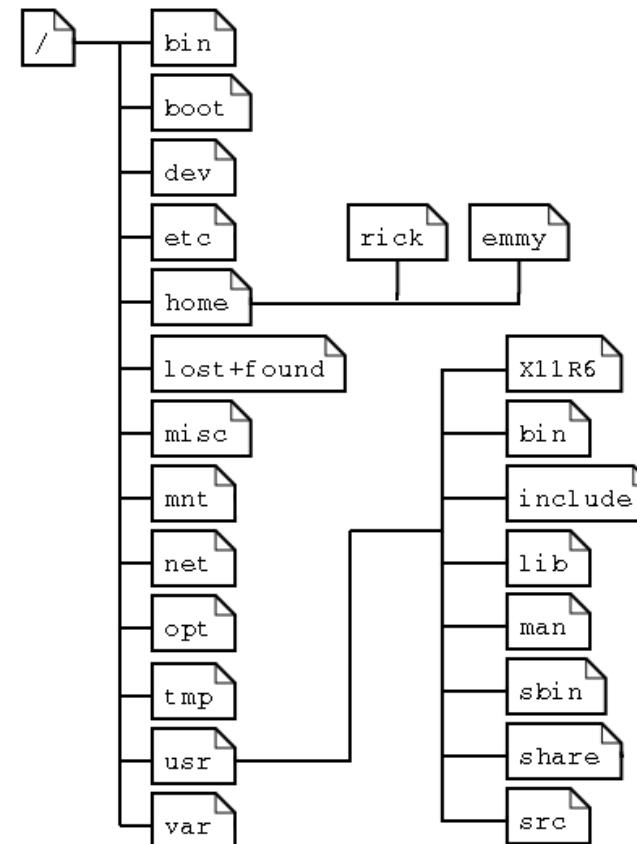


# File Tree

- Windows



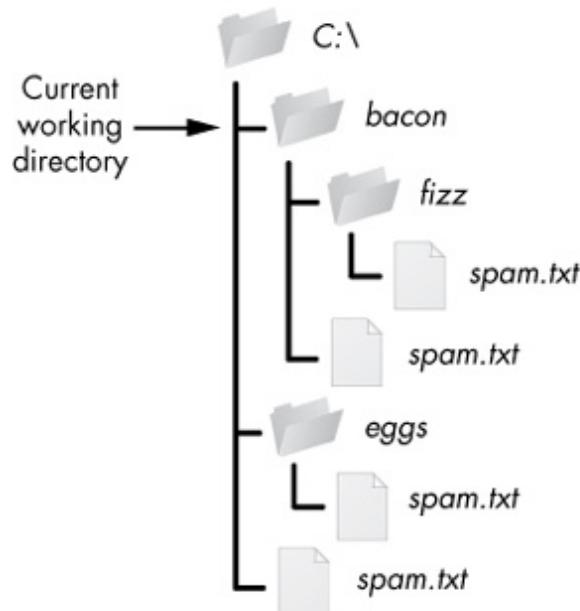
- Linux





# File Paths

An absolute file path describes how to access a given file or directory, starting from the root of the file system.



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

A relative file path is interpreted from the perspective your current working directory.

Relative Paths	Absolute Paths
..\\	C:\\
.\\	C:\\bacon
.\\fizz	C:\\bacon\\fizz
.\\fizz\\spam.txt	C:\\bacon\\fizz\\spam.txt
.\\spam.txt	C:\\bacon\\spam.txt
..\\eggs	C:\\eggs
..\\eggs\\spam.txt	C:\\eggs\\spam.txt
..\\spam.txt	C:\\spam.txt

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



# 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 readlines() to read the contents of a file, line by line

The screenshot shows a Python IDE interface with two tabs: 'Search' and 'Stack Data' on the left, and 'Debug I/O' and 'Python Shell' on the right. The 'Python Shell' tab is active, displaying the following session:

```

Search Stack Data
Search: Replace: Case sensitive Whole words In Selection
Prev Next eplac place Option: >option:
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_read_01.py]
THis is also another line
Hello world again
['THis is also another line\n', 'Hello worl

```



# 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()
```

Search Stack Data 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

file\_write.py

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

---

```
import shutil

# copy file, destination must exist
shutil.copy("hello.txt", "folder1")

# recursively copy an entire directory
# error if the destination folder already exist
shutil.copytree("folder1", "folder_to_delete")
shutil.copytree("folder1", "folder_to_delete2")

# move file, destination must exist
shutil.move("folder1/hello.txt","folder2")

# move and rename file
shutil.move("folder_to_delete/hello.txt","folder_to_delete2/newhello.txt")
```

- `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

---

```
import os

# error if file do not exist
os.unlink("folder2/hello.txt")

# get current working directory
print(os.getcwd())

# delete directory (can only delete empty folder)
os.rmdir("folder_to_delete")

import shutil
# delete directory (with content)
# error if folder is not found
shutil.rmtree("folder_to_delete2")
```

e.g. To delete all .docx file in the current folder

```
import os

for filename in os.listdir():
    if filename.endswith(".docx"):
        print(filename)
        os.unlink(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

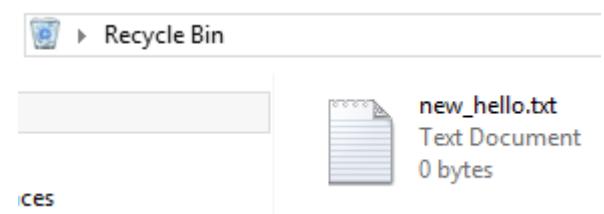
- Install send2trash module using pip.exe  
-> **pip install send2trash**
- send2trash.send2trash() will send a file or folder to the recycling bin

```
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")
```





# Use Case Sharing

- Organizing students' submissions into separate folder
  - Class of 25 students

25 folders,  
one for  
each  
student

Student 120101	Student 122431
Student 120897	Student 122868
Student 120904	Student 123295
Student 121104	Student 123525
Student 121243	Student 123534
Student 121550	Student 123673
Student 121804	Student 123864
Student 121938	Student 123900
Student 122061	Student 124059
Student 122084	Student 124133
Student 122152	Student 124990
Student 122263	Student 128079



Team 1
Team 2
Team 3
Team 4
Team 5

Student  
submission  
sorted by  
teams



# Other Use Cases

---

- System administrators can use these commands to
  - Copy and backup files to other hard-disks
  - Delete folders/ files at fixed schedules
    - End of financial year?
    - End of semester?
- Others use
  - Check timestamp of files, and delete all files created before a certain date



# Exercise

---

- **Write code to achieve the following:**

1. Create a file named: “myfile.txt”.
2. Write the following line of text into the file:
  - Programming is fun!
3. Close the file
4. Create a folder called “myfolder”
  - Use os.mkdir() command
5. Copy myfile.txt to myfolder



# Python Package Index

---

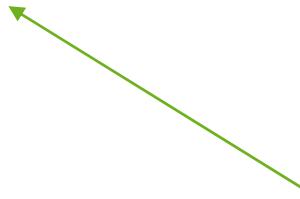
- <https://pypi.org/>
- A repository of software for the Python Programming Language
- Python Installation provides the core libraries needed for the common tasks
  - Additional packages can be found at the website and installed as extension
    - E.g. send2trash, openpyxl, pillow etc
- Installation is easy done with the following command
  - **pip install <software\_package>**
- Installed packages can be found at:
  - C:\python38\Lib\site-packages



# Using pip install

---

- **For all windows users by default**
  - Open command prompt
  - pip install <package\_name>
- **For Mac User**
  - Open terminal
  - **pip3** install <package\_name>
- **For RP staff using RP issued laptop**
  - Open command prompt
  - pip install --user <package\_name>



Double-Dash



---

# Excel Spreadsheet Manipulation with Python



# Working with Excel

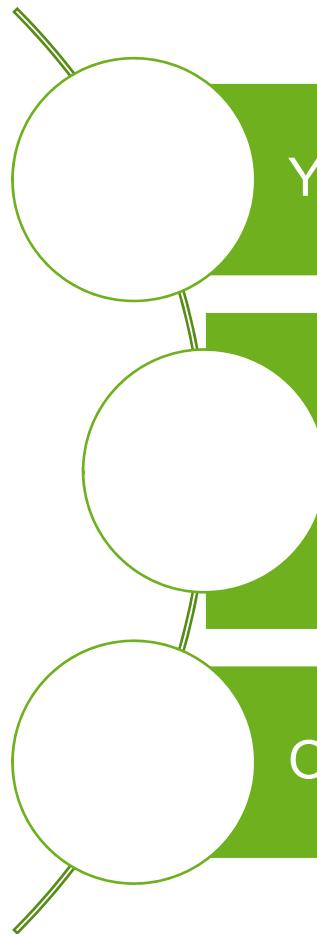
- Install openpyxl module using  
“pip install openpyxl”
- Full openpyxl documentation:  
<https://openpyxl.readthedocs.io/en/stable/index.html>

The screenshot shows the PyPI project page for openpyxl 3.0.3. At the top, there's a blue header with the version number "openpyxl 3.0.3" and a "Latest version" button. Below the header, a message says "A Python library to read/write Excel 2010 xlsx/xlsm files". On the left, there's a sidebar with "Navigation" links: "Project description" (which is selected and highlighted in blue), "Release history", and "Download files". Below that is a "Project links" section with links to "Homepage", "Tracker", "Source", and "Documentation". On the right, under "Project description", it says "coverage 95%" and has sections for "Introduction", "openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xlb/xltm files. It was born from lack of existing library to read/write natively from Python the Office Open XML format. All kudos to the PHPExcel team as openpyxl was initially based on PHPExcel.", and "Security" (with a note about quadratic blowup attacks).



# Typical Workflow for Excel Automation

---



You are given some data in a spreadsheet

You want to do some or all of the following

- Analyze the data
- Manipulate the data
- Create visualization (Charts, Pivot Table etc)

Output the processed data in another spreadsheet



# Reading Excel file

1) Import openpyxl

```
import openpyxl
```

2) Load Excel content into  
“workbook” object by  
specifying the file name

```
workbook = openpyxl.load_workbook("bmi.xlsx")  
sheet=workbook["Sheet1"]
```

3) Get the worksheet named  
"Sheet1"

```
name = sheet.cell(row=2, column=1).value  
weight = sheet.cell(row=2, column=2).value  
height = sheet.cell(row=2, column=3).value
```

4) Get the value of each cell  
by specifying the row and  
column

```
print("name:%s \tweight: %d \theight: %f " % (name, weight, height))
```

5) Display the retrieved  
values, only for a row



# Reading Excel file

The typical workflow for reading excel file is to use a **for** loop to go through each row to read the data

```
import openpyxl

workbook = openpyxl.load_workbook("bmi.xlsx")
sheet=workbook["Sheet1"]

max_row = sheet.max_row # get number of rows
#loop through every row
for i in range(2, max_row + 1):

    #read cell
    name = sheet.cell(row=i, column=1).value
    weight = sheet.cell(row=i, column=2).value
    height = sheet.cell(row=i, column=3).value

    print("name:%s \tweight: %d \theight: %f " % (name, weight, height))
```

1) Get the number of rows and columns

2) Use For loop to go through every row

3) Display the retrieved values, for all rows



# Update Excel file

```
import openpyxl

workbook = openpyxl.load_workbook("bmi.xlsx")
sheet=workbook["Sheet1"]

max_row = sheet.max_row # get number of rows

# add a column header for bmi
sheet.cell(row=1, column=4).value = "bmi"

#loop through every row
for i in range(2, max_row + 1):

    #read cell
    name = sheet.cell(row=i, column=1).value
    weight = sheet.cell(row=i, column=2).value
    height = sheet.cell(row=i, column=3).value

    bmi = weight / (height * height)

    sheet.cell(row=i, column=4).value = bmi

print("name:%s \tBMI: %f" % (name, bmi))

#save the file
workbook.save("bmi_update.xlsx")
```

1) Load file into memory & get the sheet

2) Adds a header at the 4<sup>th</sup> column

3) Perform calculation with values taken from the excel files

4) Add calculated value to cell

5) Save the spreadsheet



# Create Excel file

If you have data in nested python list, you can write the data into an excel file.

```
import openpyxl

workbook = openpyxl.Workbook()

#get the default sheet
sheet=workbook["Sheet"]

#create a list of tuples as data source
data = [
    [225.7, 'Gone with the Wind', 'Victor Fleming'],
    [194.4, 'Star Wars', 'George Lucas'],
    [161.0, 'ET: The Extraterrestrial', 'Steven Spielberg']
]

for row in data:
    sheet.append(row)

#save the spreadsheet
workbook.save("movies.xlsx")
```

1) Some data in nested list

2) Using for loop to add each row  
of data into the excel sheet

3) Save the spreadsheet



# Format Excel

```
import openpyxl  
from openpyxl.styles import Font, PatternFill, Border, Side  
  
workbook = openpyxl.load_workbook("bmi.xlsx")  
sheet=workbook["Sheet1"]
```

```
#define the colors to use for styling  
BLUE = "0033CC"  
LIGHT_BLUE = "E6ECFF"  
WHITE = "FFFFFF"  
  
#define styling  
header_font = Font(name="Tahoma", size=14, color=WHITE)  
header_fill = PatternFill("solid", fgColor=BLUE)
```

```
# format header  
for row in sheet["A1:c1"]:  
    for cell in row:  
        cell.font = header_font  
        cell.fill = header_fill  
  
#define styling  
white_side = Side(border_style="thin", color=WHITE)  
blue_side = Side(border_style="thin", color=BLUE)  
alternate_fill = PatternFill("solid", fgColor=LIGHT_BLUE)  
border = Border(bottom=blue_side, left=white_side, right=white_side)
```

```
# format rows  
for row_index, row in enumerate(sheet["A2:C3"]):  
    for cell in row:  
        cell.border = border  
        if row_index %2 :  
            cell.fill = alternate_fill
```

```
workbook.save("bmi_format.xlsx")
```

1) Import necessary functions

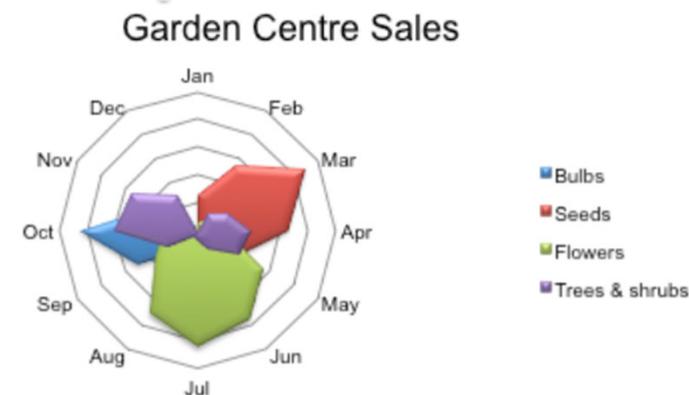
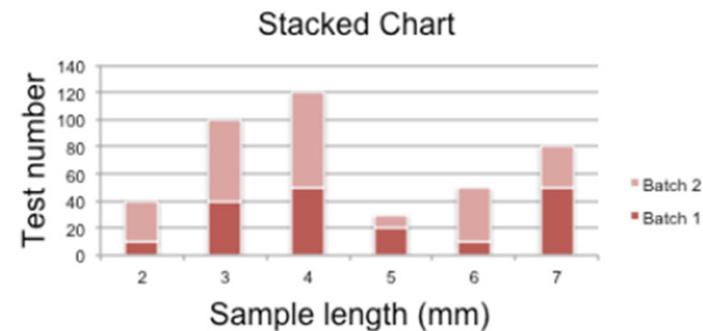
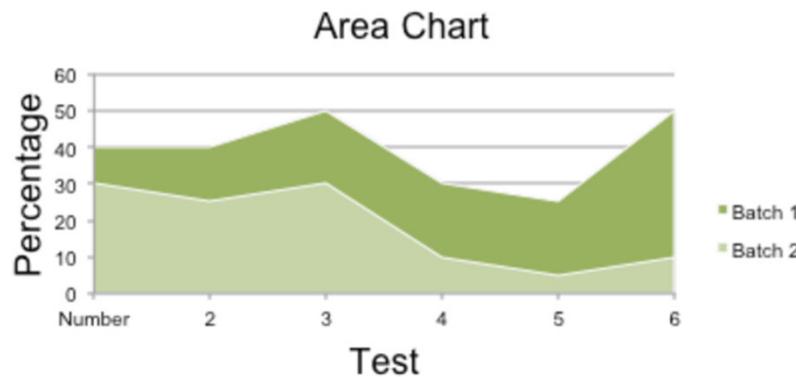
2) Setup colors and styles

3) Loop through cell and set properties



# Excel Charts

- Openpyxl supports the creation of many types of charts
  - Area Charts
  - Bar and Column Charts
  - Bubble Charts
  - Line Charts
  - Scatter Charts
  - etc





# Create Excel Chart

```
import openpyxl
from openpyxl.chart import BarChart, Reference, Series

workbook = openpyxl.load_workbook("bmi.xlsx")
sheet=workbook["Sheet1"]

chart = BarChart()

labels = Reference(sheet, min_col=1, min_row=2, max_row=3)
data = Reference(sheet, min_col=3, min_row=1, max_row=3)

chart.add_data(data, titles_from_data=True)
chart.set_categories(labels)
chart.title = "Height"

sheet.add_chart(chart, 'E1')
workbook.save('bmi_chart.xlsx')
```

1) Import necessary functions

2) Load the data from excel file

3) Specify the label and the data range

4) Add the chart to the sheet, and save the file in another excel file.



# Create Excel Chart

```
import openpyxl
from openpyxl.chart import BarChart, Reference, Series ← 1) Import necessary
functions

workbook = openpyxl.load_workbook("bmi.xlsx")
sheet=workbook["Sheet1"] ← 2) Load the data from excel
file

chart = BarChart()

# first column is used as label, starting from row 2
labels = Reference(sheet, min_col=1, min_row=2, max_row=3) ← 3) Specify the label and
the data range

# first row is used for header, that is why min_row is 1
data = Reference(sheet, min_col=3, min_row=1, max_row=3)

chart.add_data(data, titles_from_data=True)
chart.set_categories(labels)
chart.title = "Bar Chart" ← 4) Specify values for title x-axis
and y-axis for the chart
chart.x_axis.title = "Name"
chart.y_axis.title = "Height"
chart.series[0].SeriesLabel = "height"

sheet.add_chart(chart, 'E1')
workbook.save('bmi_chart.xlsx') ← 5) Add the chart to the
sheet, and save the file
in another excel file.
```



# More Explanation on Chart Reference

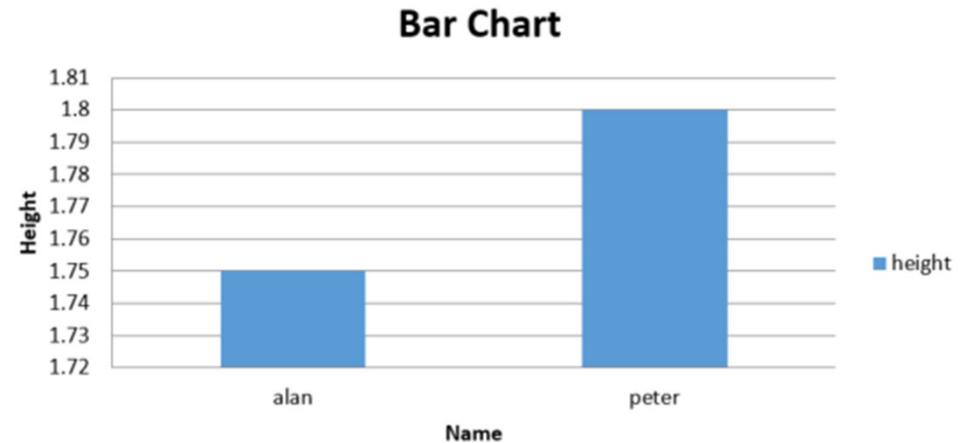
	A	B	C
1	name	weight	height
2	alan	65	1.75
3	peter	70	1.8

`data = Reference(sheet, min_col=3,  
min_row=1, max_row=3)`

`min_col = 3 ← height`

`Min_col = 2 ← weight`

`labels = Reference(sheet, min_col=1  
min_row=2, max_row=3)`





# Exercise

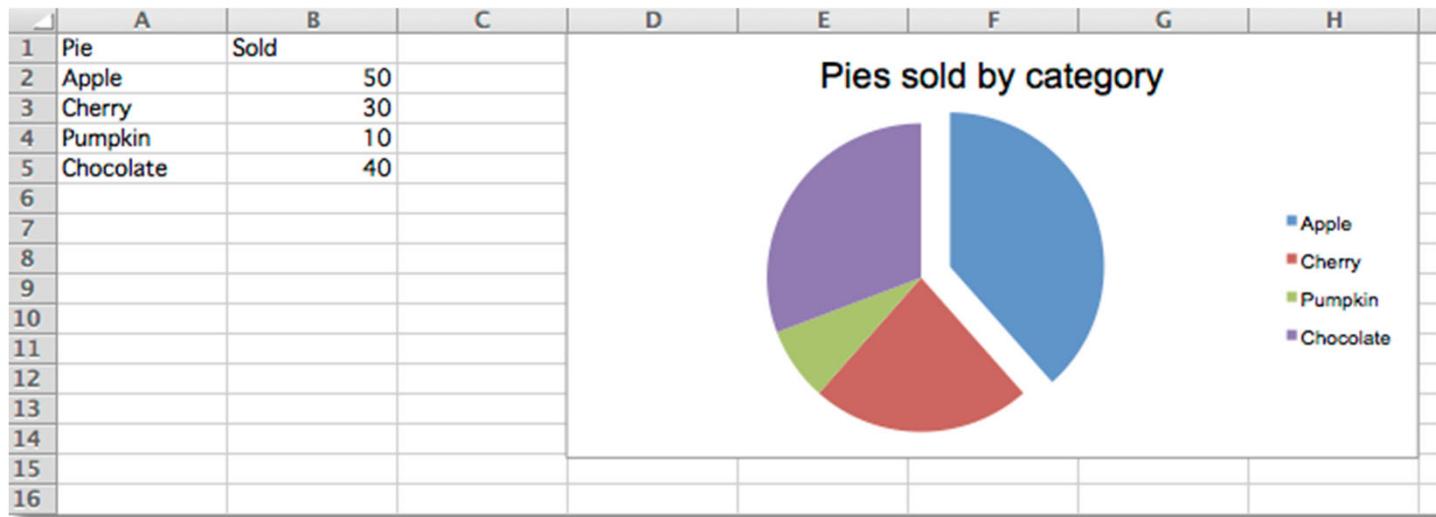
---

- Write code to achieve the following:

1. Refer to the documentation at

<https://openpyxl.readthedocs.io/en/stable/charts/pie.html#id1>

2. Follow the instruction to create the following pie chart.





# Use Case Sharing

- RPA (Robotic Process Automation) can be used to send WhatsApp messages to students individually
- Write script
  - to split contact list for students based on intake year
  - to add country code: 65 to all the numbers

	A	B	C	D
1	ReqTerm	StudentID	Name	Handphone
2	1610	16011054	MAX TAN	87733828
3	1710	16041191	PETER	87226030
4	1710	17011180	MARY	92266192
5	1810	18034448	ALAN	92336601
6	1910	19043330	JOSEPH	92468837
7	1910	19045104	ERIC	86511160
8	1910	19045784	JAVIER	97937779
9	1910	19047541	JUNE	97277250
10	1910	19011433	APRIL	97661277
11	2010	20041418	MAY	91766557

2 students  
in 2017  
intake

	A	B	C	D	E	F
1	ReqTerm	StudentID	Name	Handphone		ReqTerm
2	1710	16041191	PETER	6587226030		
3	1710	17011180	MARY	6592266192		
4						
5						
6						
7						

Create worksheets for  
students of different intake



---

# Image Processing with Python



# Image Processing

The screenshot shows the official website for Pillow, a Python Imaging Library fork. The header features the Pillow logo (a stylized 'P' inside a blue square) 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 purpose and a link to "We're here". The "Code" section links to GitHub and other CI services. The "Documentation" section links to the ReadTheDocs site. To the right of the text, there is a small image of a colorful, abstract pattern labeled "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 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:

<https://pillow.readthedocs.io/en/stable/handbook/overview.html>



# Image Processing

```
import os
from PIL import Image

filename = "img/clungup.jpg"

im = Image.open(filename)
print ("%s - %s" % (im.size, im.mode))

# show the image
im.show()

# close the file
im.close()
```

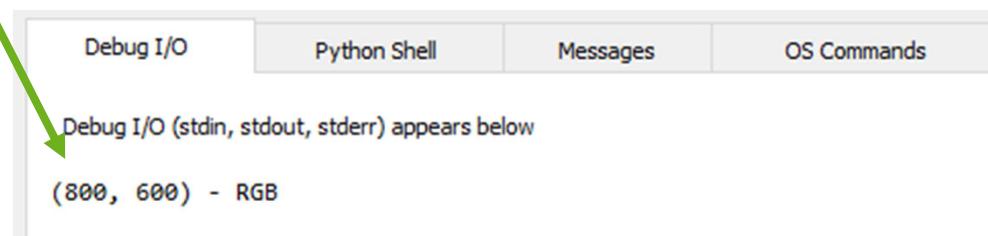


As a start we need to import it:  
`import Image`

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

Then we can display the image using  
`im.show()`

Print some info about the image using  
`im.size` and `im.mode`



Size: 800 x 600, Mode: RGB



# Image Processing

```
import os
from PIL import Image, ImageFilter

filename = "img/clungup.jpg"

im = Image.open(filename)

out = im.filter(ImageFilter.BLUR)

im.show()
out.show()
```



Pillow has many conversion and filters, 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 other different filters!



# Image processing - filters



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



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

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



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



\* Remember to include  
**ImageOps** in your import statement



# Image processing - filters

```
import os
from PIL import Image, ImageFilter, ImageOps

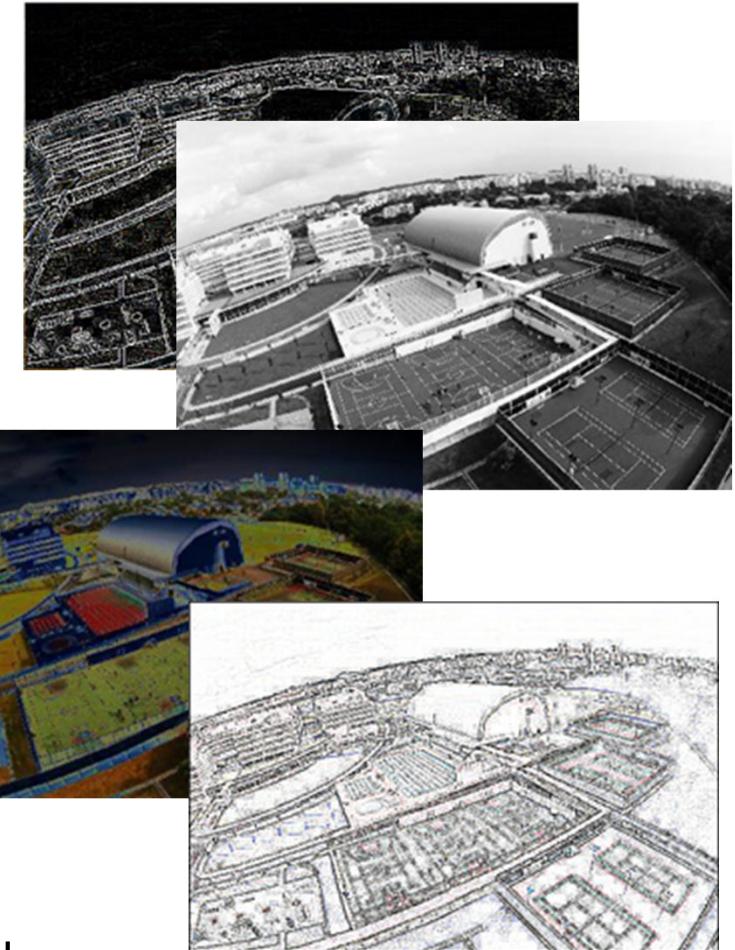
filename = "img/clungup.jpg"

im = Image.open(filename)

# Filter
#out = im.filter(ImageFilter.BLUR)
#out = im.filter(ImageFilter.FIND_EDGES)
#out = im.filter(ImageFilter.CONTOUR)

# ImageOps
out = ImageOps.grayscale(im)
#out = ImageOps.solarize(im)

im.show()
out.show()
```



\* 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)
```

} Flip images

```
#Rotating the image  
out = im.transpose(Image.ROTATE_90)  
out = im.transpose(Image.ROTATE_180)  
out = im.transpose(Image.ROTATE_270)
```

} Rotate images

```
#Contrast  
#First add ImageEnhance to our imports:  
from PIL import Image, ImageFilter, ImageEnhance
```

#Then:

```
enh = ImageEnhance.Contrast(im)  
out = enh.enhance(1.3)
```



make image brighter by  
changing the contrast



# Image Processing - Writing

---

```
import os
from PIL import Image, ImageFilter, ImageOps

filename = "clungup.jpg"

src_folder = "img/"
out_folder = "out/"

im = Image.open(src_folder + filename) # img/clungup.jpg
out = im.filter(ImageFilter.BLUR)

outFilename = out_folder + filename # out/clungup.jpg

out.save(outFilename)
```

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

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



# Image processing – Converting

Sometimes, we may want to keep all your photos in the same format.

Or, we obtained some gif files but want to have bmp or png type.

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

fname1		fname2
holiday.gif	->	holiday.jpg

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

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



# Image processing – Converting

---

```
import os
from PIL import Image, ImageFilter, ImageOps

filename = "clungup.jpg"

src_folder = "img/"
out_folder = "out/"

im = Image.open(src_folder + filename) # img/clungup.jpg
out = im.filter(ImageFilter.BLUR)

# split the filename and the extension
f, e = os.path.splitext(filename)

# add the gif extension to the filename
fname2 = f + ".gif"

outFilename = out_folder + fname2 # out/clungup.gif

out.save(outFilename)
```

`os.path.splitext(file)` returns a list.  
We are only interested in `f` which is  
the first item in the list.



# 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)
```

There could be cases where 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.

Copyright  
@RP



# Image processing – Watermark

```
from PIL import Image

def watermark(im, mark, position):
    layer = Image.new("RGBA", im.size, (0,0,0,0))
    layer.paste(mark, position)
    return Image.composite(layer, im, layer)

im = Image.open("img\\clungup.jpg")
mark = Image.open("img\\watermark.png")
mark = mark.resize((100,100))
mark.putalpha(128)

out = watermark(im, mark, (0,0))
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 shown here



# Use Case I: Batch Resize

---

1. Find all the files in “img” folder with “.jpg” extension
2. Resize all the file to 60 x 90.
3. Save all the files to the resized folder

```
import os
from PIL import Image, ImageFilter, ImageOps

files = os.listdir('img')
size = 60, 90

for file in files:
    if file.lower().endswith('.jpg'):
        im = Image.open("img/" + file)
        im.thumbnail(size, Image.ANTIALIAS)
        im.save("resized/" + file, "JPEG")
```



# Use Case II: Batch Rename

---

1. Find all the files in “img” folder with “.jpg” extension
2. Copy all the files to the renamed folder
3. Rename all the files with the “s-” prefix.

```
import os
import shutil

files = os.listdir('img')

for file in files:
    if file.lower().endswith('.jpg'):
        shutil.copyfile("img/" + file, "renamed/s-" + file[:-4] + ".jpg")
```



---

# Web Automation with Python



# Connecting to the Web

- requests – download files and web pages from the Web

pip install requests

```
import requests

url="https://api.data.gov.sg/v1/environment/24-hour-weather-forecast"
req=requests.get(url)
print(req.text)
```

Get the required information from  
the given URL





# Connecting to the Web

---

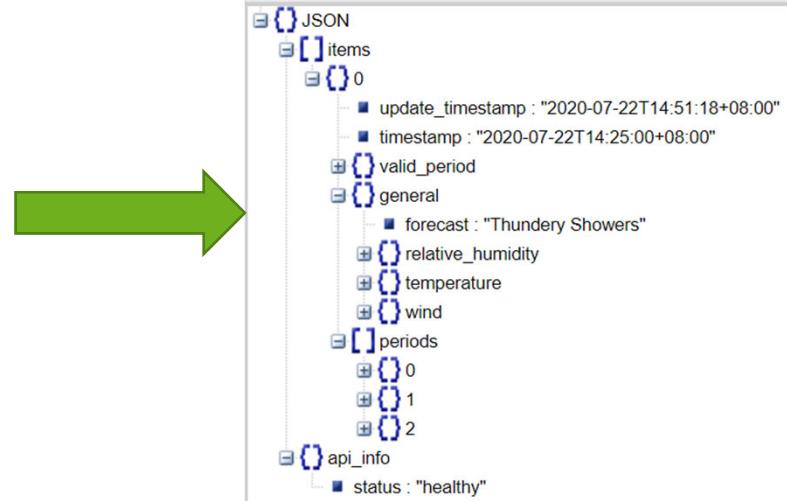
- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

<http://jsonviewer.stack.hu/>

```
import requests

url="https://api.data.gov.sg/v1/environment/24-hour-weather-forecast"
req=requests.get(url)
print(req.text)

{
  "items": [
    {
      "update_timestamp": "2020-07-22T14:51:18+08:00",
      "timestamp": "2020-07-22T14:25:00+08:00",
      "valid_period": {
        "start": "2020-07-22T14:25:00+08:00",
        "end": "2020-07-23T12:00:00+08:00"
      },
      "general": {
        "forecast": "Thunder Showers",
        "relative_humidity": {
          "low": 70,
          "high": 95
        },
        "temperature": {
          "low": 22,
          "high": 28
        },
        "wind": {
          "speed": {
            "low": 10,
            "high": 20
          },
          "direction": "ESE"
        }
      },
      "periods": [
        {
          "start": "2020-07-22T12:00:00+08:00",
          "end": "2020-07-22T18:00:00+08:00",
          "regions": [
            {
              "west": "Moderate Rain",
              "east": "Moderate Rain",
              "central": "Light Rain",
              "north": "Light Rain"
            }
          ],
          "time": {
            "start": "2020-07-22T14:25:00+08:00",
            "end": "2020-07-23T06:00:00+08:00"
          }
        },
        {
          "start": "2020-07-22T18:00:00+08:00",
          "end": "2020-07-23T06:00:00+08:00",
          "regions": [
            {
              "west": "Partly Cloudy (Night)",
              "east": "Partly Cloudy (Night)",
              "central": "Partly Cloudy (Night)",
              "north": "Partly Cloudy (Night)"
            }
          ],
          "time": {
            "start": "2020-07-22T20:00:00+08:00",
            "end": "2020-07-23T06:00:00+08:00"
          }
        }
      ]
    }
  ],
  "api_info": {
    "status": "healthy"
  }
}
```





# 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

```
import json
import requests

url="https://api.data.gov.sg/v1/environment/24-hour-weather-forecast"
req=requests.get(url)

data = json.loads(req.text)

# print update timestamp
update_time = data["items"][0]["update_timestamp"]
print("Update time: " + update_time)                                Update time: 2020-07-22T14:51:18+08:00
                                                                    Forecast: Thunder Showers

# print forecast
forecast = data["items"][0]["general"]["forecast"]
print("Forecast: " + forecast)
```

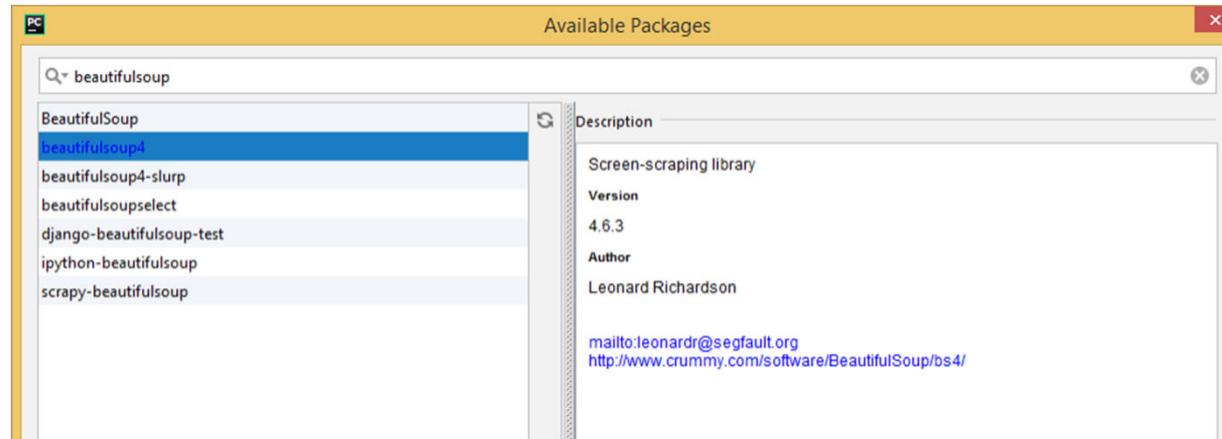


# Connecting to the Web

- 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/landon-regular-dining-table-coffee.html>

Enquiries: [+65 6777 7667](tel:+6567777667) | Mon - Fri (10am - 6pm)

**FORTYTWO**


🔍

---

New   Furniture ▾   Bedding & Mattresses ▾   Décor | Essentials ▾   Kitchen | Dining ▾   Lightings | Fans ▾   Sale ▾

Home > Dining Room Furniture > Dining Tables > Landon Regular Dining Table Coffee



Landon Regular Dining Table  
Coffee

★★★★★ 6 customer reviews

S\$129.90  
**S\$69.90**

Warranty: 1 Year

---

Standard Delivery

42EXPR

100 Day Free Returns

Free Assembly Included

Qty:  ▼

**Add to Cart**

♥ Add to Wishlist  
✉ Email to a Friend



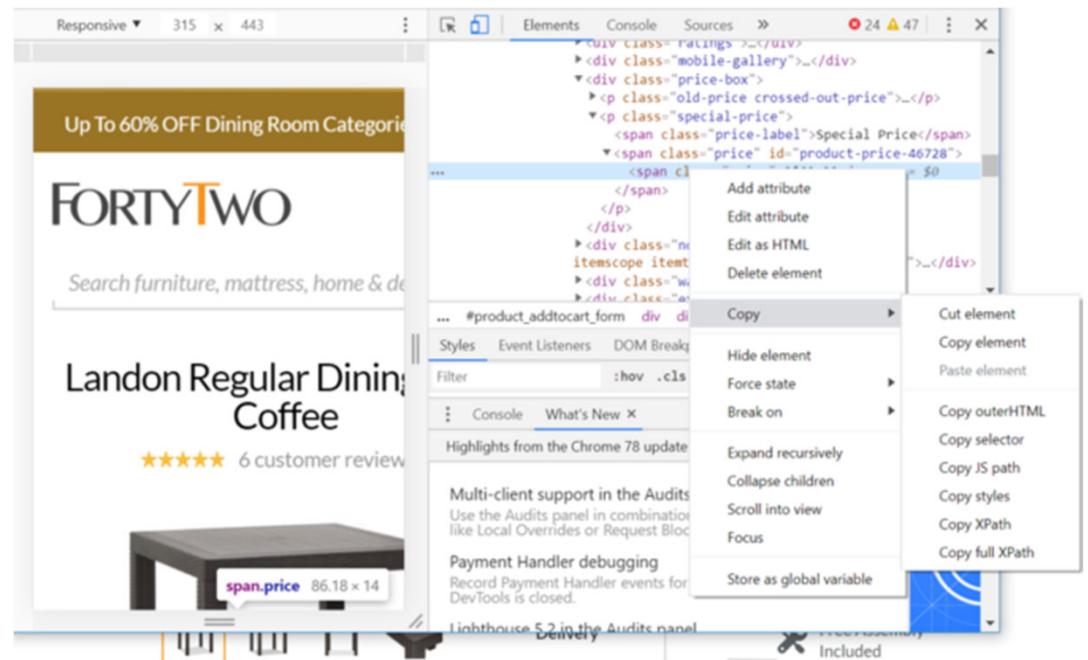
# Connecting to the Web

---

- Get the url

<https://www.fortytwo.sg/dining/dining-tables/landon-regular-dining-table-coffee.html>

- Select the element to extract
  - right-click -> "Inspect"
  - hover to "Copy"
  - click on "Copy selector"





# Connecting to the Web

- Get the url
- Select the element to extract
  - right-click -> "Inspect"
  - hover to "Copy"
  - click on "Copy selector"

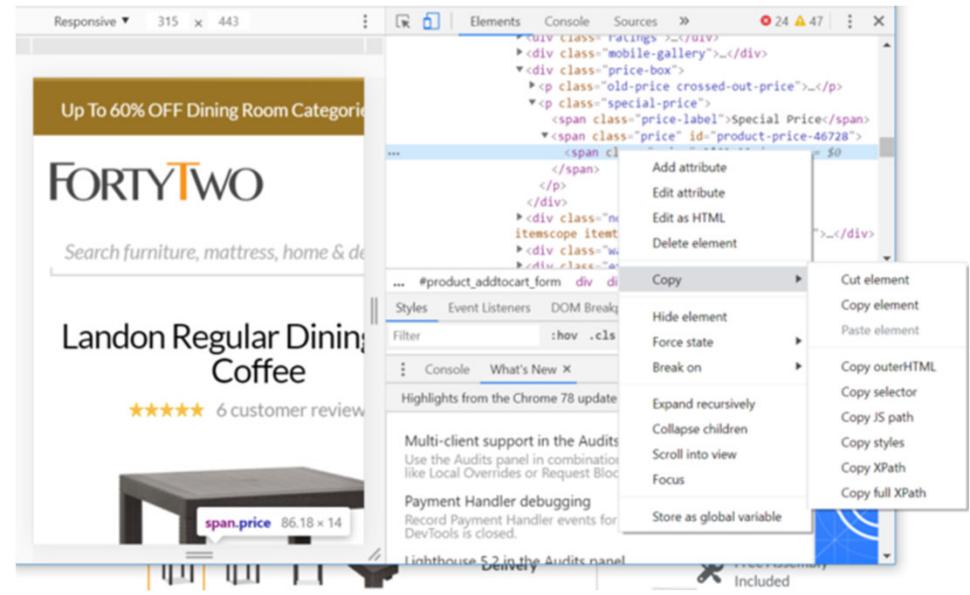
```
import bs4
import requests

requestObj = requests.get("https://www.fortytwo.sg/dining/dining-tables/landon-regular-dining-table-coffee.html")
requestObj.raise_for_status()
soup = bs4.BeautifulSoup(requestObj.text, 'html.parser')

elements = soup.select("#product-price-46728") # $69.90
print("Current Price: " + elements[0].text)

elements = soup.select("#old-price-46728") # $129.90
print("\nOld Price: " + elements[0].text)

elements = soup.select("div.earliest-delivery-date") # Earliest by Sunday, 31 May 2020
print("\nDelivery Date: " + elements[0].text)
```



Debug I/O	Python Shell	Messages	OS C
Debug I/O (stdin, stdout, stderr) appears below			
Current Price: S\$69.90			
Old Price: S\$129.90			
Delivery Date: Earliest by Sunday, 31 May 2020			

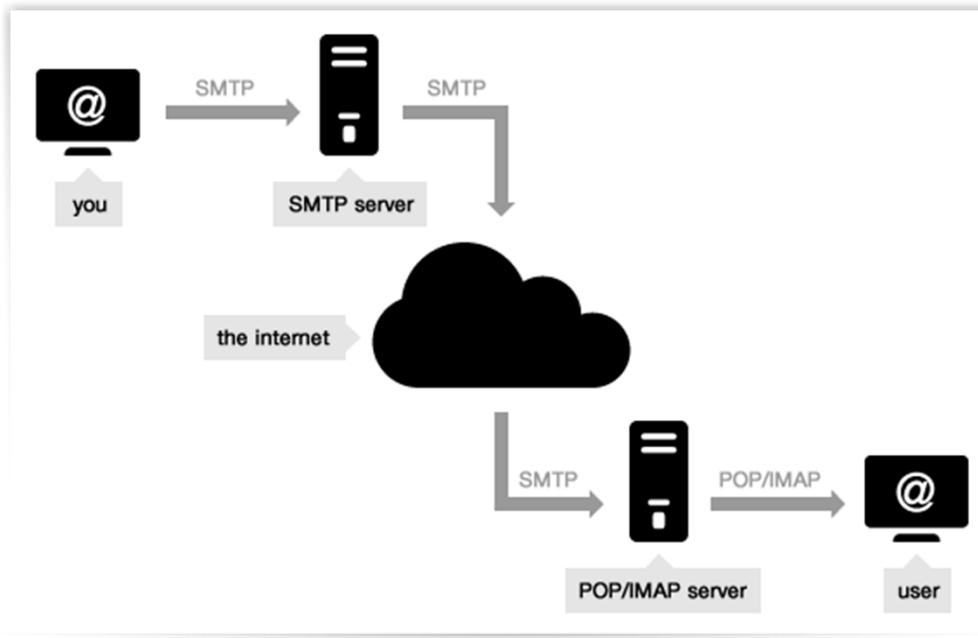


---

# Email Automation with Python



# Send Email



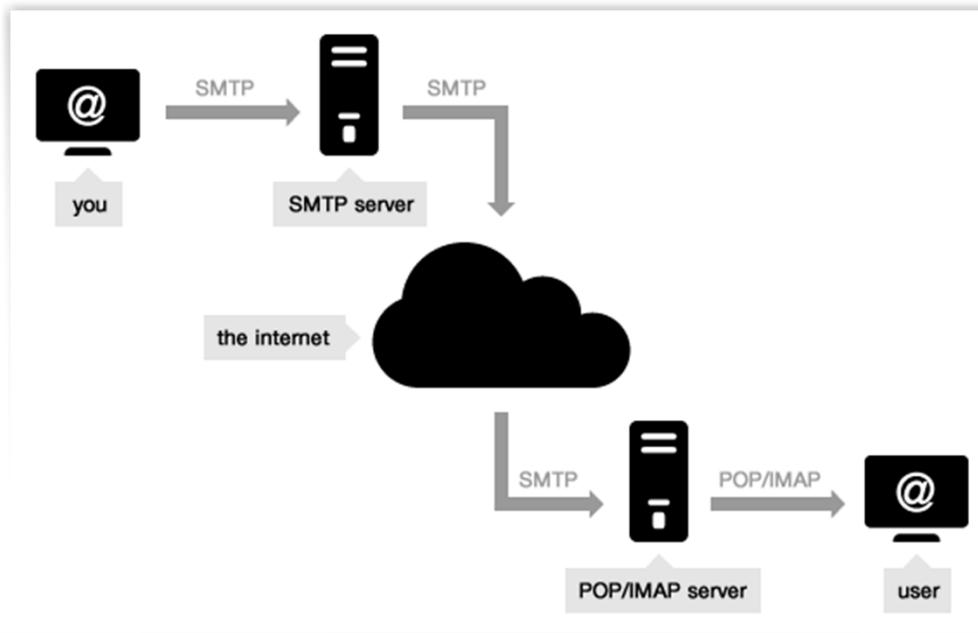
- SMTP (Simple Mail Transfer Protocol) is used for sending and delivering from a client to a server via port 25, 465 or 587: it's the **outgoing server**.
- IMAP and POP are two methods to access email. IMAP is the recommended method when you need to check your emails from several different devices, such as a phone, laptop, and tablet.

<https://www.mailgun.com/blog/which-smtp-port-understanding-ports-25-465-587/>

<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

Note: If you are using your office network, most port numbers, including 587, may be blocked.



# 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."

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...")

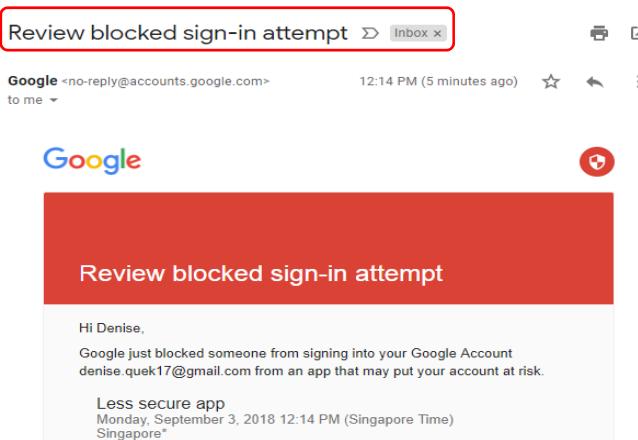
smtpObj.quit()
```

➤ 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.



# 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

[← 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

**[REDACTED]**

Add your Google account

Enter the information below to connect to your Google account.

Email address: `securesally@gmail.com`

Password: `*****`

Include your Google contacts and calendars

**GENERATE**

### Your app password for Windows Computer

#### 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.

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**



# Send Email using Gmail

The screenshot shows the Google Account Security settings page. The left sidebar has a red box around the 'Security' tab. The main area shows 'Security issues found' with a lock icon and a yellow exclamation mark. Below it is a 'Secure account' section. The 'Signing in to Google' section has a red box around the '2-Step Verification' setting, which is turned 'On'. Other options include 'Password' (last changed 10 Jan 2018), 'App passwords' (1 password), and 'Ways that we can verify that it's you' (with icons for key, phone, and email).

Google Account

Security

Settings and recommendations to help you keep your account secure

Security issues found

Protect your account now by resolving these issues

Secure account

Signing in to Google

2-Step Verification  On

App passwords 1 password

Ways that we can verify that it's you

These can be used to make sure that it's really you



# 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
```



# Use Case: Send emails to students

---

- Send email to students who were absent

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

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



# Use Case: Send Emails to Students

---

- Open an Excel file
- 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
```



# Sharing other Use Cases

---

- Sending Emails using Outlook
- Create Appointment using Outlook



---

# Generate PDF Report with Python



# PDF

PyPDF

Search docs

Project Home

---

Home

- FPDF for Python
- Main features
- Installation
- Support
- ProjectHome**
- Reference manual
- Tutorial
- Tutorial (Spanish translation)
- FAQ (Frequently asked questions)
- Python 3
- Templates
- Unicode
- Web2Py framework
- Testing
- Development
- Reference manual
- accept\_page\_break
- add\_font
- add\_link
- add\_page
- alias\_nb\_pages
- cell
- close
- dashed\_line

Docs » Project Home » Home

Edit on GitHub

## FPDF for Python

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

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

### Main features

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

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

- Python 2.5 to 3.4+ support (see Python3 support)
- Unicode (UTF-8) TrueType font subset embedding (Central European, Cyrillic, Greek, Baltic, Thai, Chinese, Japanese, Korean, Hindi and almost any other language in the world) New! based on sPDF LGPL3 PHP version from Ian Back
- Improved installers (setup.py, py2exe, PyPI) support
- Barcode (2D) and code39, QR code coming soon ...
- PNG, GIF and JPG support (including transparency and alpha channel) New!
- Exceptions support, other minor fixes, improvements and PEP8 code cleanups
- Port of the Tutorial and ReferenceManual (Spanish translation available)

FPDF original features:

- Install fpdf
- pip install fpdf

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



# PDF – Basic document

```
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()

#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(0,10, "This is an example of a long paragraph. \n" * 10)
document.ln()

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

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

**PDF Test Document**  
This is an example of a long paragraph.  
This is an example of a long paragraph.



# PDF – adding images

```

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=5, w=23)

document.set_y(40);

#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()

#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.



# PDF – Adding password

```
import fpdf
import PyPDF2

#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()

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

#save the document
document.output("pdf_report_before_pw.pdf")

#save the document into a new password protected/encrypted pdf
pdffile = open(r"pdf_report_before_pw.pdf", "rb")
pdfReader = PyPDF2.PdfFileReader(pdffile)
pdfWriter = PyPDF2.PdfFileWriter()
for pageNum in range(pdfReader.numPages):
    pdfWriter.addPage(pdfReader.getPage(pageNum))

pdfWriter.encrypt('123') ←
resultPDF = open(r"pdf_report_after_pw.pdf", "wb")
pdfWriter.write(resultPDF)
resultPDF.close()
pdffile.close()
```

- pip install PyPDF2

Password is 123

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



# Use Cases

---

- Automation:
  - Generation of reports with data from spreadsheet or database
  - Generation of Course Certificates in PDF format
  - Password protection of banking statement in PDF file



---

# Charting/Visualisation with Python



# Charting



Version 3.0.2

[home](#) | [examples](#) | [tutorials](#) | [API](#) | [docs](#) »

[Fork me on GitHub](#)
[modules](#) | [index](#)
[Quick search](#)
 Go

[Table of Contents](#)
**Gallery**

- Lines, bars and markers
- Images, contours and fields
- Subplots, axes and figures
- Statistics
- Pie and polar charts
- Text, labels and annotations
- Pyplot
- Color
- Shapes and collections
- Style sheets
- Axes Grid
- Axis Artist
- Showcase
- Animation
- Event handling
- Front Page
- Miscellaneous
- 3D plotting
- Our Favorite Recipes
- Scales
- Specialty Plots
- Ticks and spines
- Units
- Embedding Matplotlib in

## Gallery

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.

For longer tutorials, see our [tutorials page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

### Lines, bars and markers


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

**pip install matplotlib**

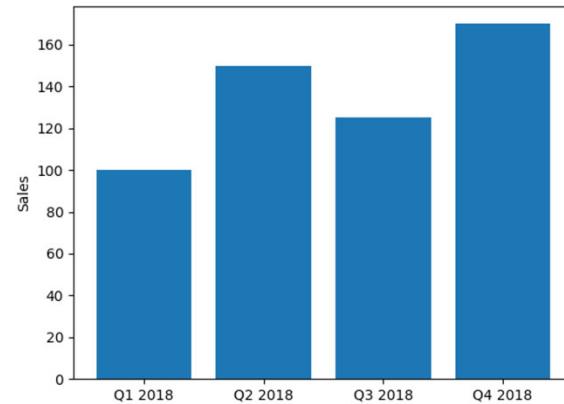
Full documentation:  
<https://matplotlib.org/>



# 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](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html)



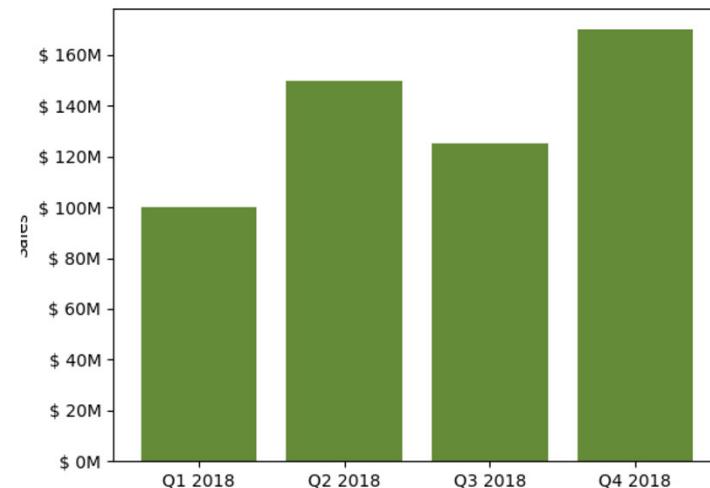
# Charting

```

1 import matplotlib.pyplot as plt
2 from matplotlib.ticker import FuncFormatter
3
4 def value_format(value, position):
5     return '$ {}'.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





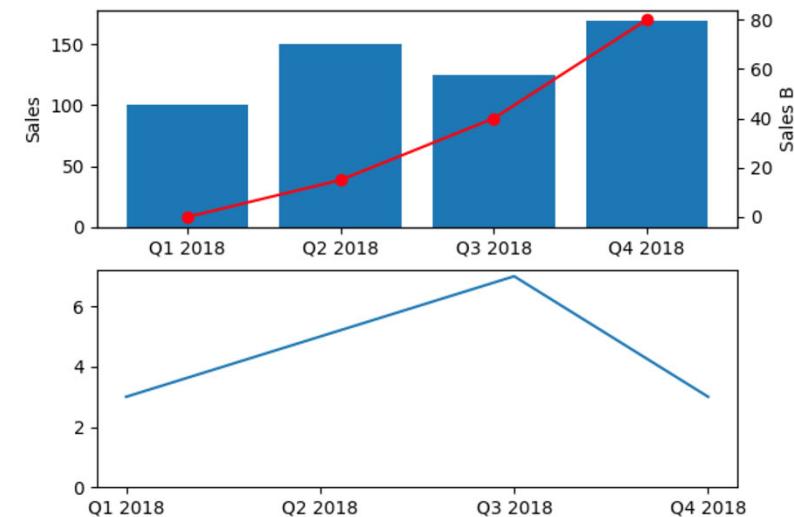
# Charting

```

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 #create a bar graph with information 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](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html)



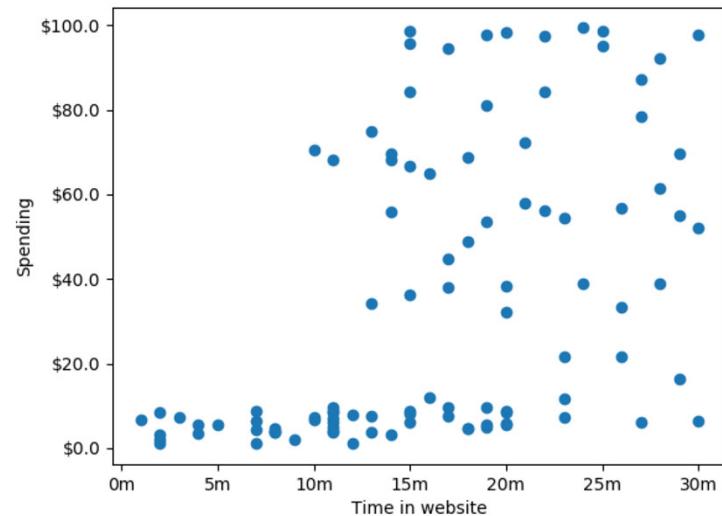
# Charting – Scatter Plot

```

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





# End of Day 2

---

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

**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>

The image shows two screenshots of online learning platforms. On the left, the DataCamp website displays a search results page for "python" with five course cards visible. On the right, the edX homepage features a banner for Cyber Monday with logos of partner institutions like MIT, Harvard, and Berkeley.

**DataCamp Screenshot:**

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

**edX Homepage Screenshot:**

Accelerate your future. Learn anytime, anywhere.

Find courses

What do you want to learn?

MIT HARVARD BERKELEY THE UNIVERSITY OF TEXAS SYSTEM THE HONG KONG POLYTECHNIC UNIVERSITY THE UNIVERSITY OF BRITISH COLUMBIA

THE COUNTDOWN IS ON!  
Get 15% off your purchase.  
Start Exploring

CYBER MONDAY



# 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>

# Lifelong Learning



Scan me



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

## Short Courses



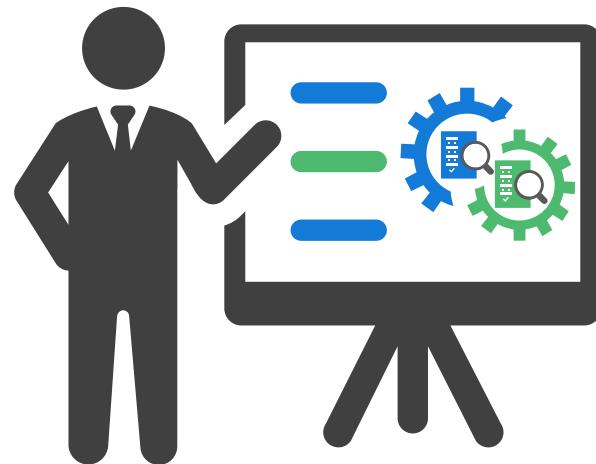
SOI offers an extensive variety of short, industry-relevant courses for ICT skills upgrading and skills acquisition. Our courses are categorized under different areas, ranging from Artificial Intelligence (AI), Business Intelligence/Business Analytics (BI/BA), Business Processes (BP), Unmanned Aerial 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



Email:

tay\_mei\_lan@myrp.edu.sg  
jason\_lim@rp.edu.sg

Learning material & source code:  
<https://bit.ly/py-78oct2021>