

# Introductory Programming Using Python

---

## Day 2

By Wong Hau Shian and Lynn Teo

Republic Polytechnic



# Trainers

---

## Day 2

**Wong Hau Shian**  
**(wong\_hau\_shian@rp.edu.sg)**



**Lynn Teo**  
**(lynn\_teo@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>• Creating Chart</li><li>• Generating PDF</li></ul>



# Outline for the day

---

Time	Agenda
9.00am	Welcome and admin matters
9.15am – 10.30am	
10.30am – 10.45am	Break
10.45am – 12.30pm	
12.30pm – 1.30pm	Lunch
1.30pm – 3.15pm	
3.15pm – 3.30pm	Break
3.30pm – 4.30pm	
4.45pm – 5.00pm	Wrap up, Q&A



# 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.

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

For example,

```
example_directory.
```

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



# Read files

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

Open() will return a file object which has reading and writing related methods

Pass 'r' (or nothing) to open() to open the file in read mode.

Call read() to read the contents of a file

Call close() when you are done with the file.

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

Search Stack Data

Search:

Replace:

☐ Case sensitive ☐ Whole words ☐ In Selection

Previ Ne: eplac place Option:

Debug I/O Python Shell

Commands execute without debug. Use arrow keys for history.

Options

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

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



# Write files

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

Search Stack Data

Search:

Replace:

☐ Case sensitive ☐ Whole words ☐ In Selection

Previ Ne: eplac place

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_write.py]
This is also another line

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
```

Search Stack Data Debug I/O Python Shell

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

Replace:

☐ Case sensitive ☐ Whole word ☐ In Selection

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 i  
[evaluate file\_copy\_01.py]  
>>>  
>>>

- `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("hello.txt")

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

# delete directory (can only delete empty folder)
os.rmdir("folder3")

import shutil
# delete directory (with content)
# error if folder is not found
shutil.rmtree("folder3")
```

- `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

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)
```

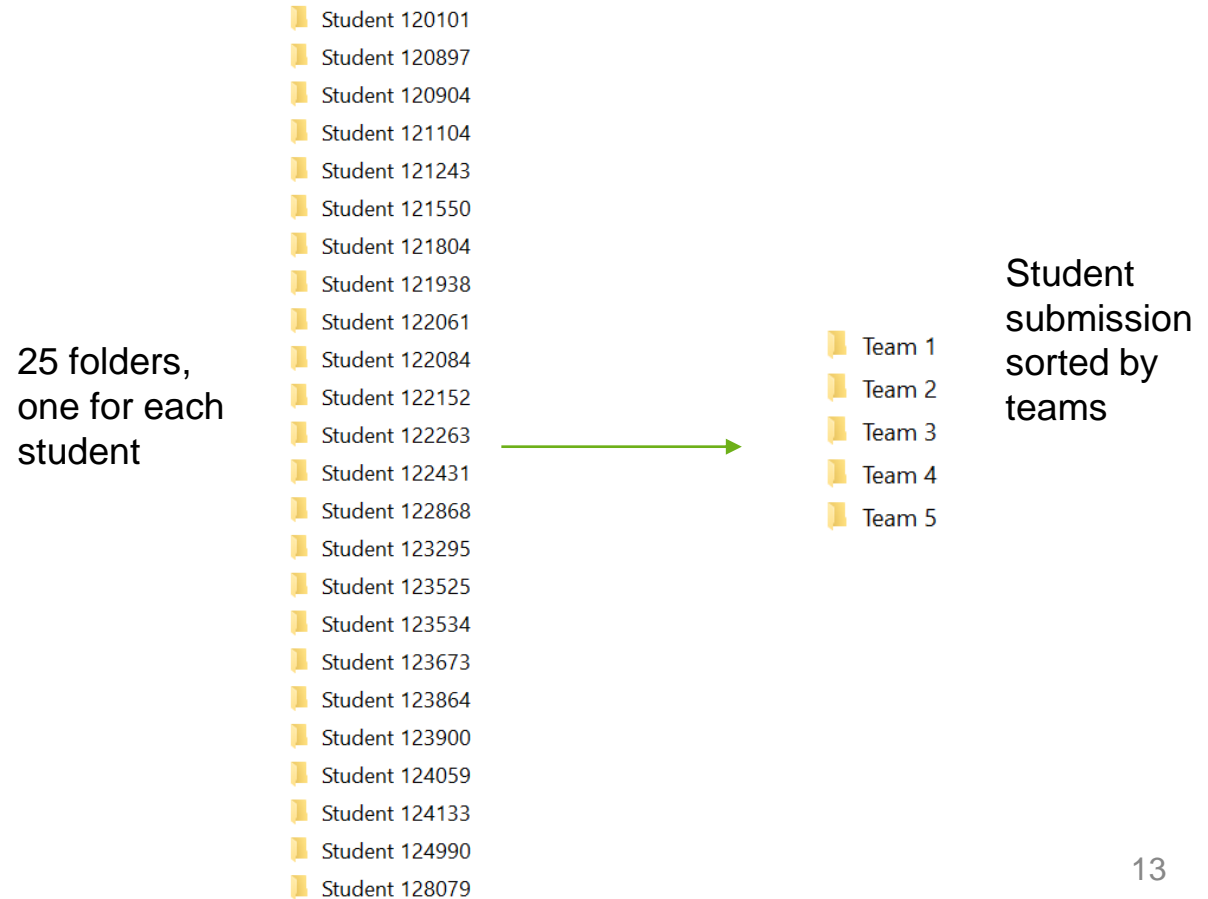


Deleting can be dangerous, so do a dry run first



# Use Case Sharing

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





# 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 staff using company issued laptop with no Admin rights**
  - Open command prompt
  - `pip install --user <package_name>`

Double-Dash



Break

# 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 page for openpyxl 3.0.3. The header is blue with the text 'openpyxl 3.0.3' and a 'pip install openpyxl' button. A green badge indicates it is the 'Latest version' and 'Released: Jan 11, 2020'. Below the header, a grey bar states 'A Python library to read/write Excel 2010 xlsx/xlsm files'. The main content area is divided into two columns. The left column, titled 'Navigation', contains links for 'Project description' (highlighted), 'Release history', and 'Download files'. The right column, titled 'Project description', includes a 'coverage 95%' badge, an 'Introduction' section, and a 'Security' section. The 'Introduction' section describes openpyxl as a Python library for reading and writing Excel 2010 xlsx/xlsm/xlt/xltx files, noting its origin from the lack of existing libraries and its basis on PHPExcel. The 'Security' section mentions that openpyxl does not guard against quadratic blowup or billion laughs XML attacks by default, but these can be guarded against by installing defusedxml.

**openpyxl 3.0.3** ✓ Latest version  
`pip install openpyxl`

Released: Jan 11, 2020

A Python library to read/write Excel 2010 xlsx/xlsm files

**Navigation**

- Project description
- Release history
- Download files

**Project links**

- Homepage
- Tracker
- Source
- Documentation

**Project description**

coverage 95%

**Introduction**

openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xlt/xltx 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.

**Security**

By default openpyxl does not guard against quadratic blowup or billion laughs xml attacks. To guard against these attacks install defusedxml.



# Typical Workflow for Excel Automation

---

1

You are given some data in a spreadsheet

2

You want to do some or all of the following

- Analyze the data
- Manipulate the data

3

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



# 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"]
```

1) Get the number of rows and columns

```
max_row = sheet.max_row # get number of rows
```

```
#loop through every row
```

```
for i in range(2, max_row + 1):
```

2) Use For loop to go through every row

```
    #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))
```

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



# 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")
```

2) Load file into memory & get the sheet

1) Perform calculation with values taken from the excel files

2) Add comments 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']  
]
```

1) Some data in nested list

```
for row in data:  
    sheet.append(row)
```

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

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

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



# Typical Workflow for Excel Automation

---

1

You are given some data in a spreadsheet

2

You want to do some or all of the following

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

3

Output the processed data in another spreadsheet





# Use Case Sharing

- Use RPA (Robotic Process Automation) to send WhatsApp messages to students individually
- Write script to split my contact list for students based on intake year
  - Added country code: 65 to all the numbers as well

	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

A1						
						ReqTerm
	A	B	C	D	E	F
1	ReqTerm	StudentID	Name	Handphone		
2	1710	16041191	PETER	6587226030		
3	1710	17011180	MARY	6592266192		
4						
5						
6						
7						
	Sheet	1610	1710	1810	1910	2010

Create worksheets for  
students of different intake

# Image Processing with Python



# Image Processing

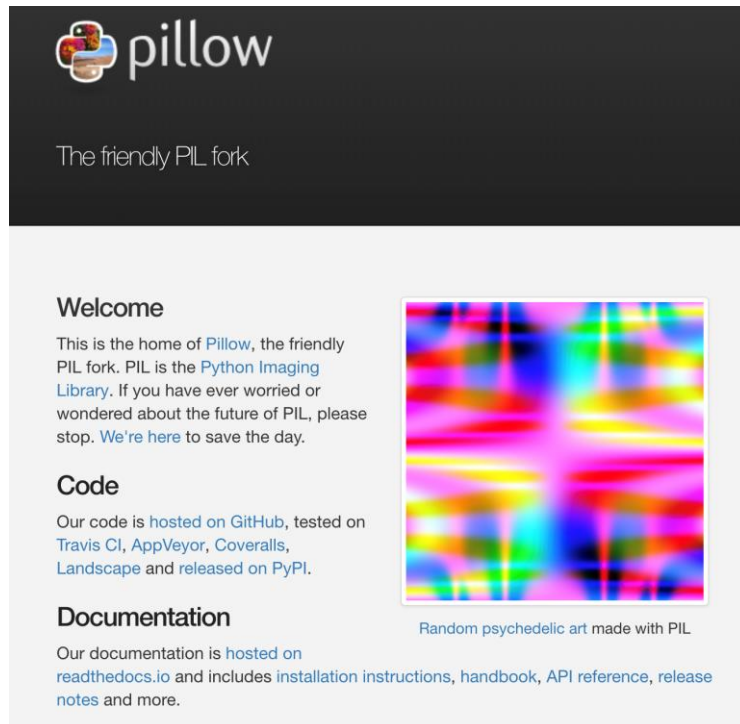
For the next section we are going to use the Python Image Library, or in short Pillow.

Install using the following command:

```
pip install pillow
```

The documentation is at:

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



The screenshot shows the Pillow website homepage. At the top, there's a dark header with the Pillow logo (a snake) and the text "pillow" and "The friendly PIL fork". Below this, the main content area has a light gray background. On the left, there's a "Welcome" section with text about Pillow being the friendly PIL fork. Next to it is a square image of random psychedelic art made with PIL. Below the welcome section are sections for "Code" (mentioning GitHub, Travis CI, AppVeyor, Coveralls, and PyPI) and "Documentation" (mentioning readthedocs.io and including links to installation instructions, handbook, API reference, and release notes).

**Welcome**

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

**Code**

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

**Documentation**

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

Random psychedelic art made with PIL



# Image Processing

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`

```
import os  
from PIL import Image
```

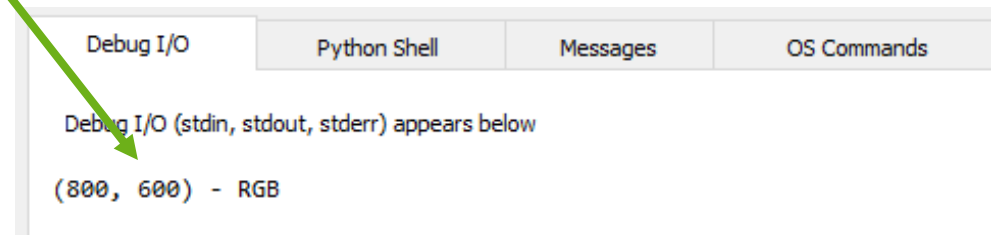
```
filename = "img/clungup.jpg"
```

```
im = Image.open(filename)
```

```
print ("%s - %s" % (im.size, im.mode))
```

```
# show the image  
im.show()
```

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



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
```

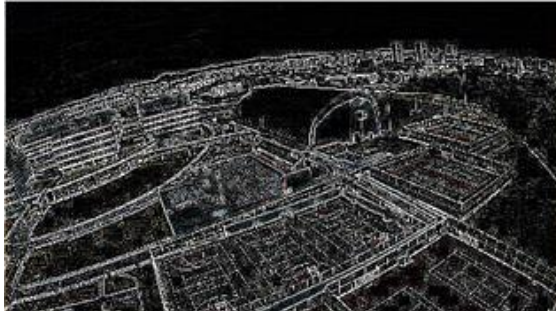
The way you can apply filters is :

```
out = im.filter(ImageFilter.BLUR)
```

Try other different filters!



# Image processing - filters



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

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



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



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



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





# Image processing - filters

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

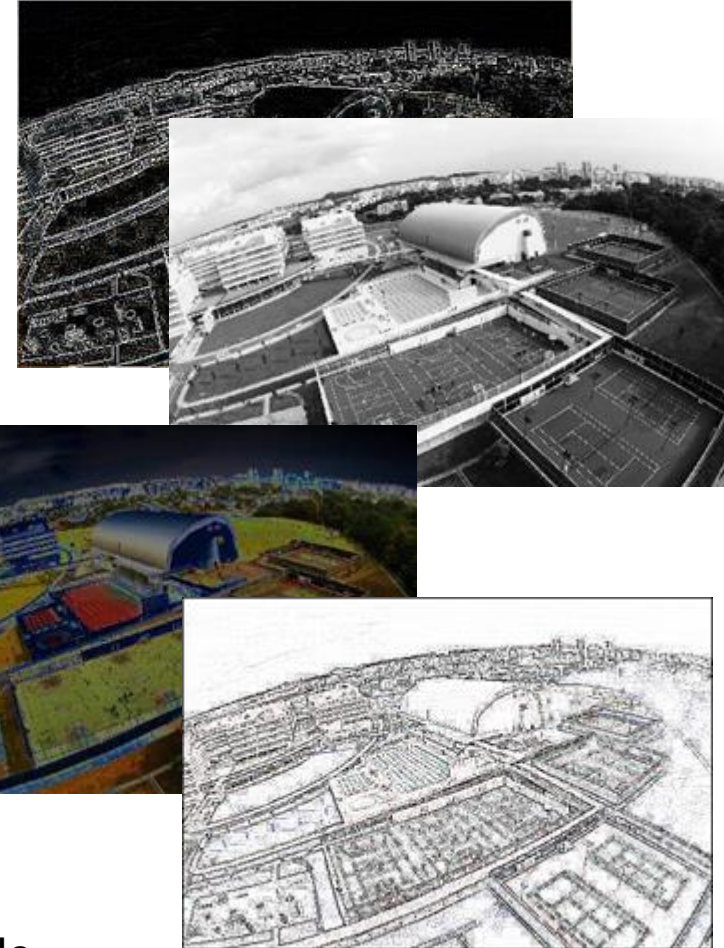
filename = "img/clungup.jpg"

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

---

```
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)
```

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

In this case we can use the \\img\\watermark.png and place it in each image on the bottom right.



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 this:





# 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")
```

LUNCH

# Web Automation with Python



# Connecting to the Web

- requests – download files and web pages from the Web

pip install requests

```
import requests
```

Get the required information from the given URL

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







# Connecting to the Web

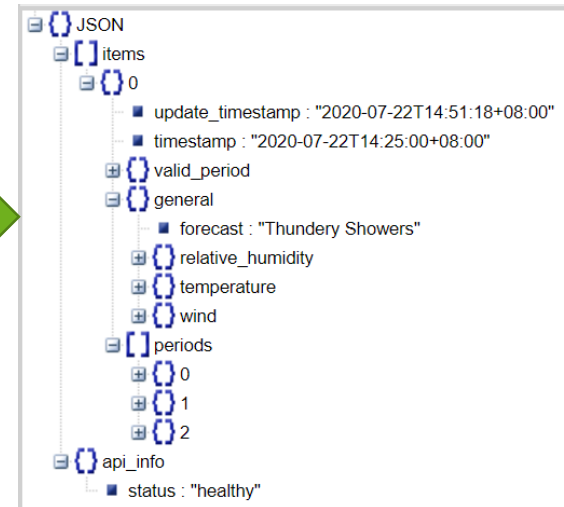
- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

<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": "Thundery 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",
          "forecast": {
            "west": "Moderate Rain",
            "east": "Moderate Rain",
            "central": "Light Rain",
            "north": "Light Rain"
          },
          "time": {
            "start": "2020-07-22T12:00:00+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": "2020-07-23T06:00:00+08:00",
            "end": "2020-07-23T12:00:00+08:00"
          },
          "regions": {
            "west": "Partly Cloudy (Night)",
            "east": "Partly Cloudy (Night)",
            "central": "Partly Cloudy (Night)",
            "north": "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

```
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)

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

```
Update time: 2020-07-22T14:51:18+08:00
Forecast: Thundery Showers
```



# Exercise

- **Car Park Availability Data:**

1. url: <https://api.data.gov.sg/v1/transport/carpark-availability>
2. Write the code to get the timestamp and the Carpark Number for the first set of carpark data.
3. Print out the result as shown.

```
{
  - items: [
    - {
      timestamp: "2021-08-19T13:23:28+08:00",
      - carpark_data: [
        - {
          - carpark_info: [
            - {
              total_lots: "105",
              lot_type: "C",
              lots_available: "0"
            }
          ],
          carpark_number: "HE12",
          update_datetime: "2021-08-19T13:10:03"
        },
        - {
          - carpark_info: [
            - {
```

Update time: 2021-08-19T11:49:27+08:00  
Carpark number: HE12

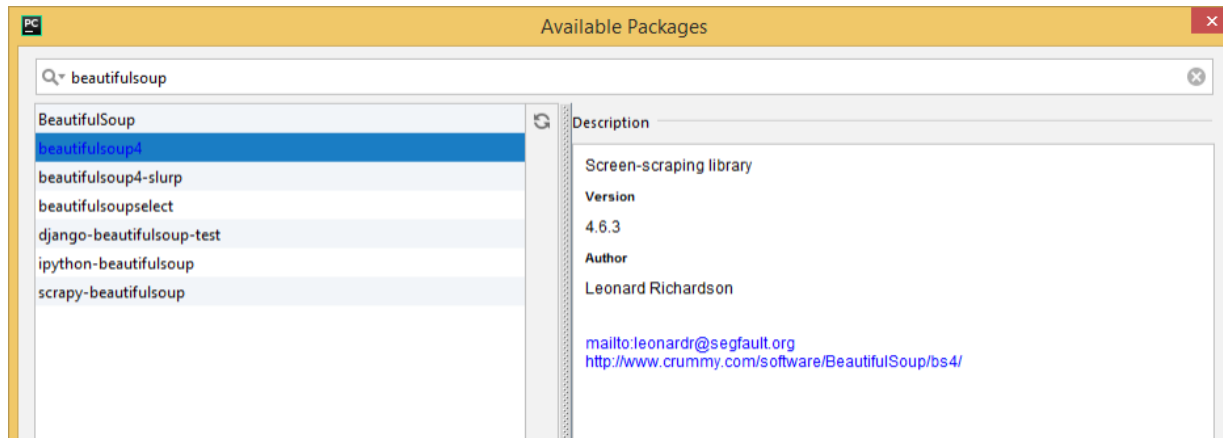


# Connecting to the Web

- 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 | Mon - Fri (10am - 6pm)

FORTYTWO

Search furniture, mattress, home & decor...



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

Qty: 1

Add to Cart

♥ Add to Wishlist

✉ Email to a Friend

100 Day Free  
Returns

Standard  
Delivery

42EXPR



Free Assembly  
Included



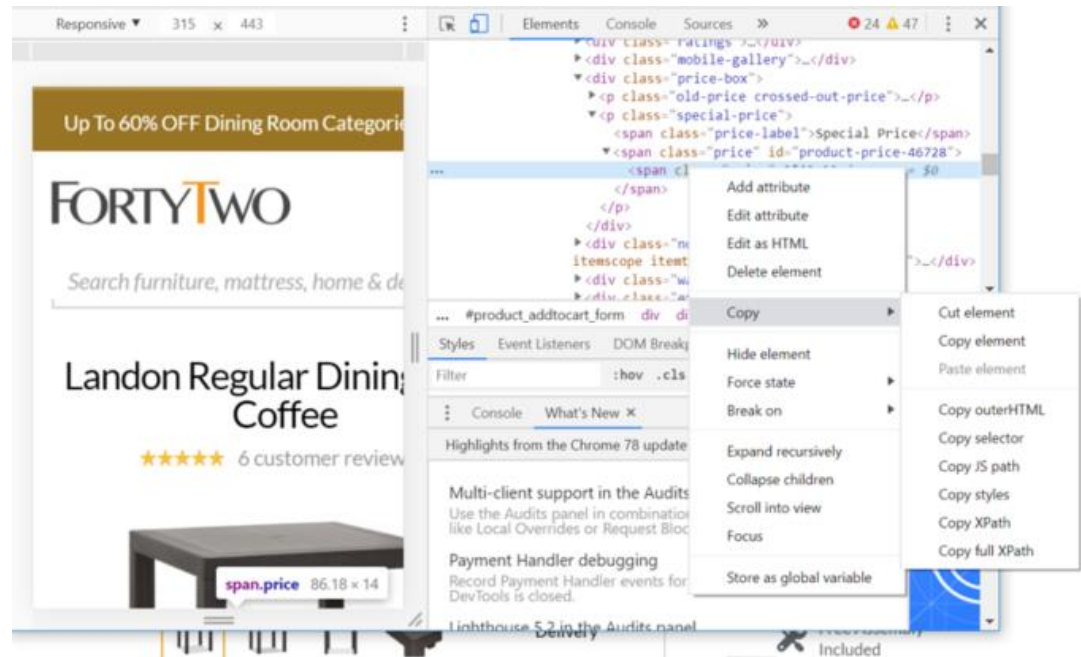


# Connecting to the Web

- Get the url

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

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





# Connecting to the Web

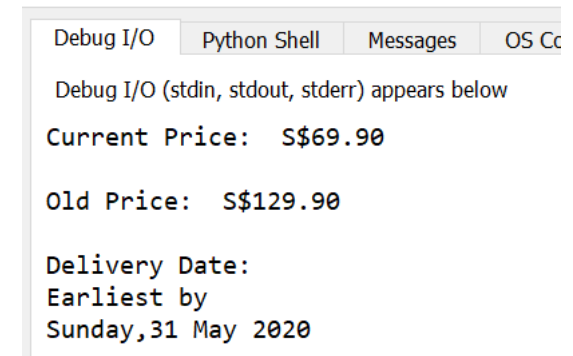
```
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup
```

```
site= "https://www.fortytwo.sg/dining/dining-tables/landon-regular-dining-table-coffee.html"
hdr = {'User-Agent': 'Mozilla/5.0'}
req = Request(site,headers=hdr)
page = urlopen(req)
soup = BeautifulSoup(page, 'html.parser')
```

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

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

```
elements = soup.select('div[class="delivery est-date"]') # Earliest by Sunday, 31 May 2020
print(elements[0].text)
```





# Exercise

- **Table Tennis Bat Price:**

1. url: [https://www.tabletennis11.com/other\\_eng/butterfly-viscaria](https://www.tabletennis11.com/other_eng/butterfly-viscaria)
2. Write the code to get the price of the table tennis bat
3. Print out the result as shown.



Current Price:  
€133.25

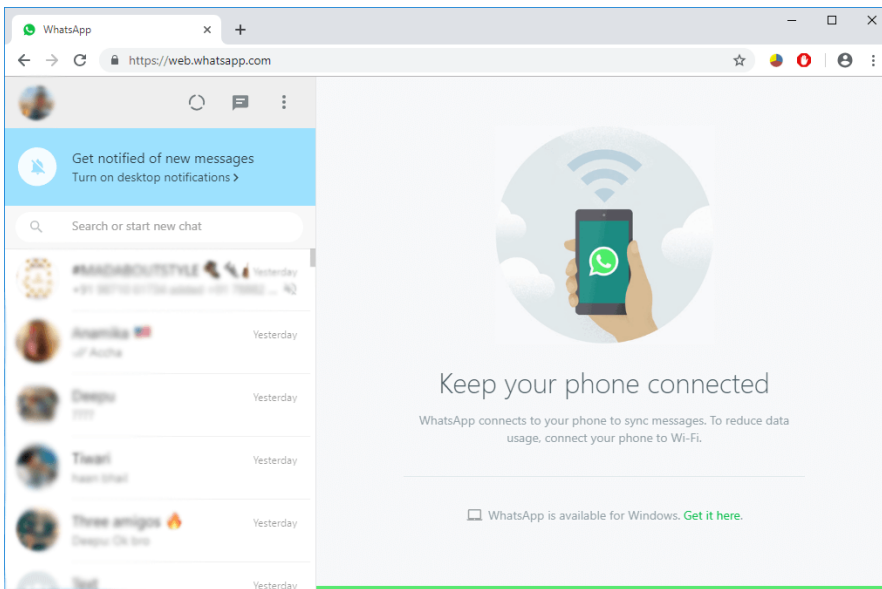
web\_scrap\_tabletennis.py





# Sharing other Use Cases

- Using another library: selenium
  - Filling up google form
  - Sending WhatsApp message



selenium python automation

Name

Your answer

Gender

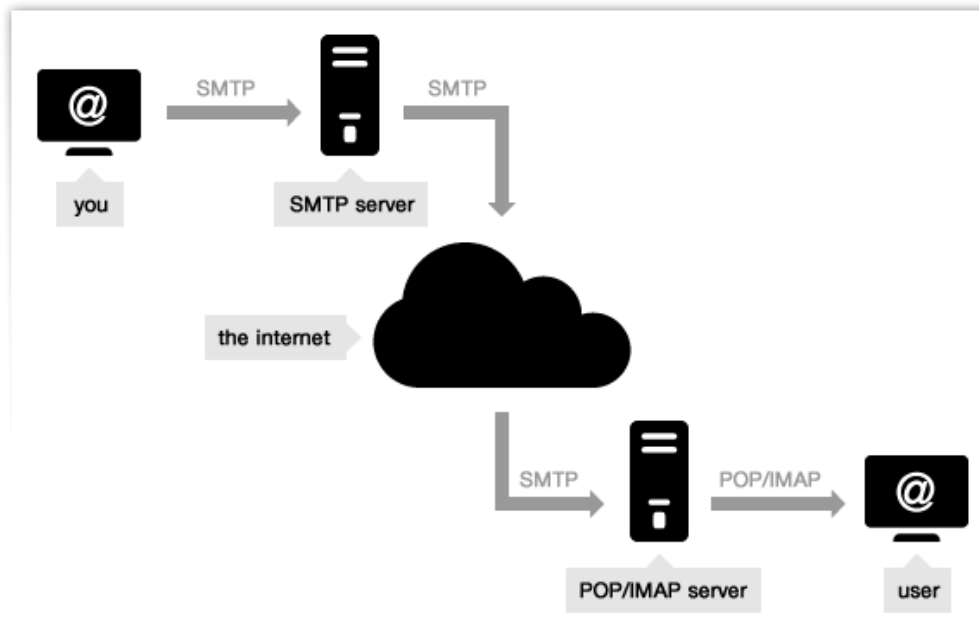
☐ Male

☐ Female

# Email Automation with Python



# Send Email



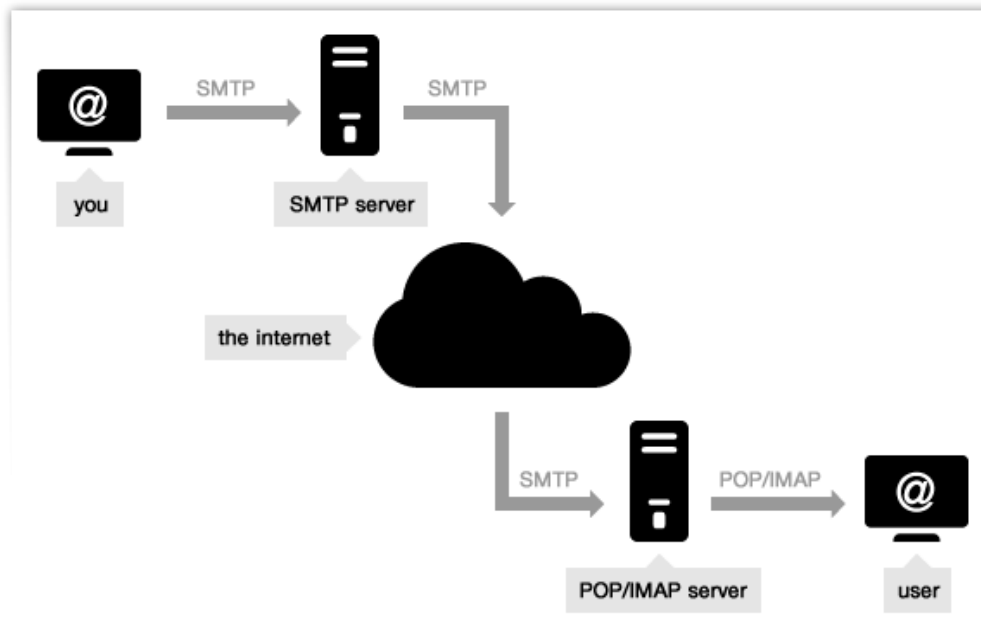
- SMTP (Simple Mail Transfer Protocol) is used for sending and delivering from a client to a server via port 25, 465 or 587: it's the **outgoing server**.
- IMAP and POP are two methods to access email. IMAP is the recommended method when you need to check your emails from several different devices, 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 = "xxxxxxxxxxxxxxxx"  
receiver_email_address = "another_email_address@gmail.com"  
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."  
  
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."
```

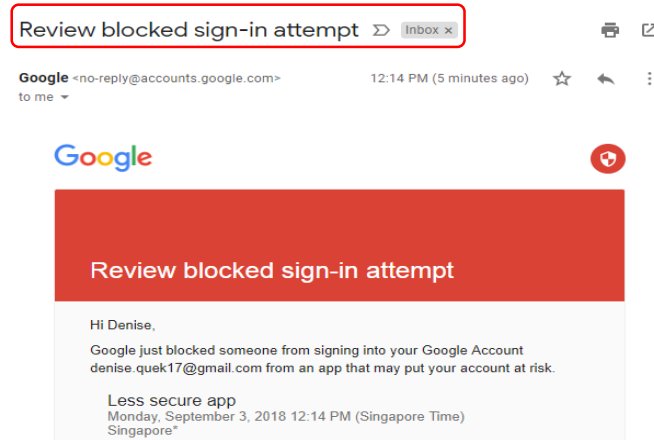
➤ The start of the email body must begin with "Subject: " for the subject line. The "\n" newline character separates the subject line from the main body content.

```
print("Trying to connect to Gmail SMTP server")  
smtpObj = smtplib.SMTP("smtp.gmail.com", 587)  
smtpObj.starttls()  
  
print("Connected. Logging in...")  
smtpObj.login(sender_email_address, sender_email_password)  
  
smtpObj.sendmail(sender_email_address, receiver_email_address, email_title_content)  
print("Email sent successfully...")  
  
smtpObj.quit()
```



# Send Email using Gmail

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



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

Process finished with exit code 1
```



# Send Email using Gmail

## Steps To Create Google App Password

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

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

Step 3: Create an App password

← 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

Add your Google account

Enter the information below to connect to your Google account.

Email address  
seuresally@gmail.com

Password  
\*\*\*\*\*

☐ Include your Google contacts and calendars

Your app password for Windows Computer

16-digit app password

### 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.

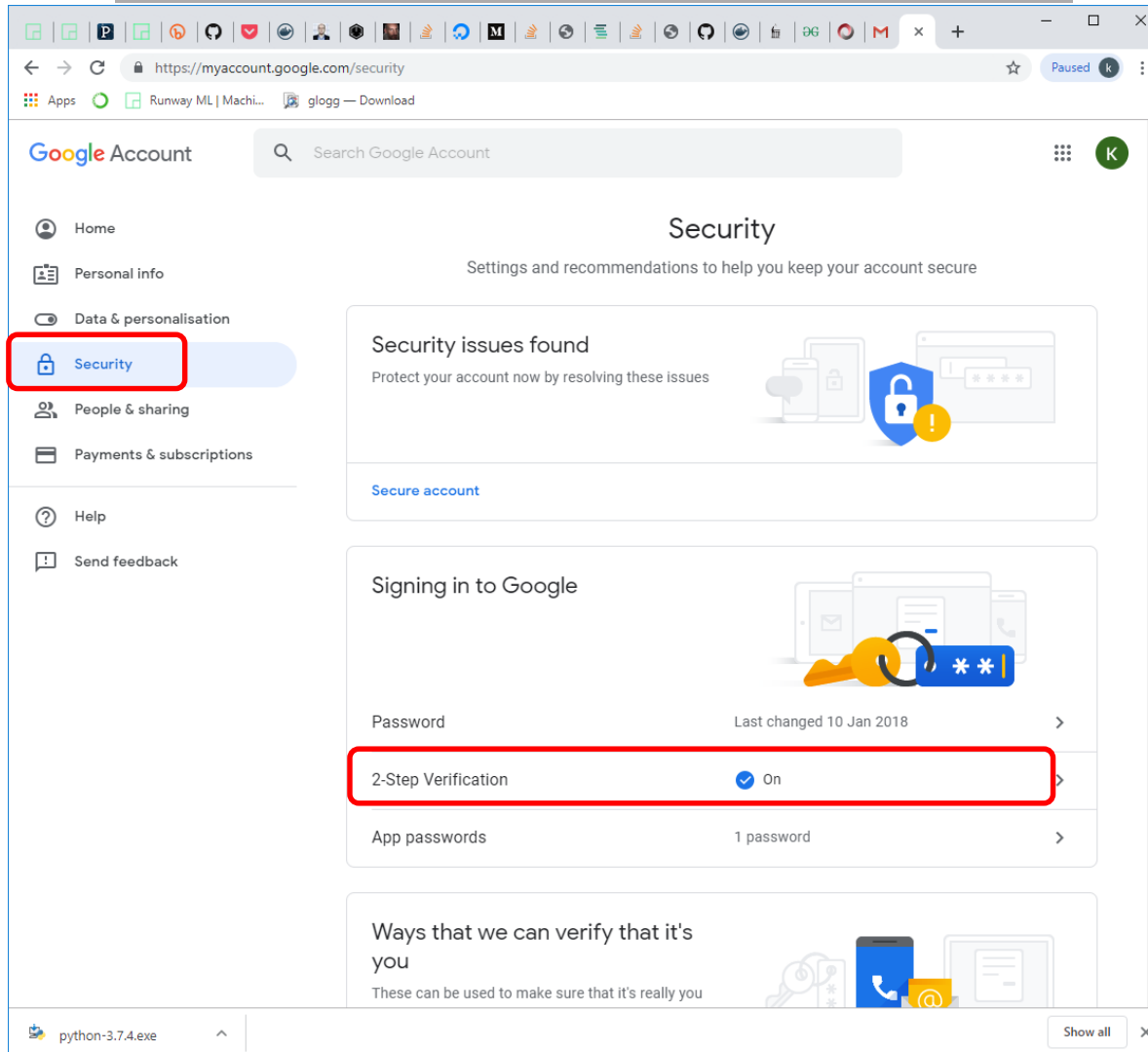
[Learn more](#)

DONE





# Send Email using Gmail



The screenshot shows the Google Account Security page in a web browser. The browser's address bar displays the URL `https://myaccount.google.com/security`. The page features a left-hand navigation menu with the following items: Home, Personal info, Data & personalisation, Security (highlighted with a red rectangle), People & sharing, Payments & subscriptions, Help, and Send feedback. The main content area is titled "Security" and includes the subtitle "Settings and recommendations to help you keep your account secure". It contains three primary sections: "Security issues found" with a "Secure account" link; "Signing in to Google" which lists "Password" (last changed 10 Jan 2018), "2-Step Verification" (marked as "On" with a blue checkmark and highlighted by a red rectangle), and "App passwords" (1 password); and "Ways that we can verify that it's you". The browser's taskbar at the bottom shows an open instance of "python-3.7.4.exe".



# 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 = "xxxxxxxxxxxxxxxxxx"
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
```



# Sharing other Use Cases

---

- Sending Emails using Outlook
- Create Appointment using Outlook

# Generate PDF Report with Python



# PDF

PyFPDF

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

*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).

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 [sFPDF](#) LGPL3 PHP version from [Ian Back](#)
- Improved installers ([setup.py](#), [py2exe](#), [PyPI](#)) support
- Barcode 12of5 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.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.



# 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")
```

```
fpdf.cell(w, h = 0, txt = '', border = 0, ln = 0,
         align = '', fill = False, link = '')
```

- 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.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.

This is an example of a long paragraph.



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

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





# Use Cases

---

- Automation:
  - Generation of reports with data from spreadsheet or database
  - Generation of Course Certificates in PDF format

# Charting/Visualisation with Python



# Charting

**matplotlib** Version 3.0.2

home | examples | tutorials | API | docs » modules | index

## 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

- Arc test
- Stacked Bar Graph
- Bar chart
- Horizontal bar chart
- Broken Barh
- Plotting categorical variables
- Plotting the coherence of two signals
- CSD Demo

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

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

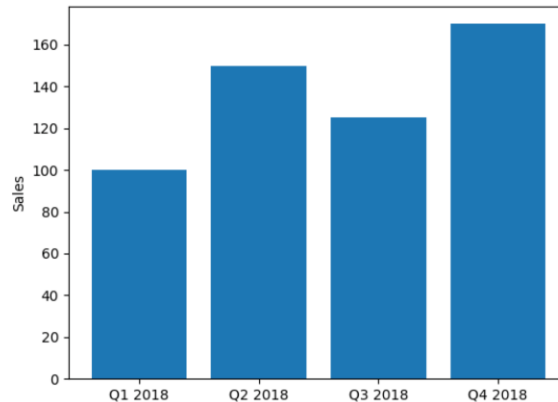
pip install matplotlib

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)



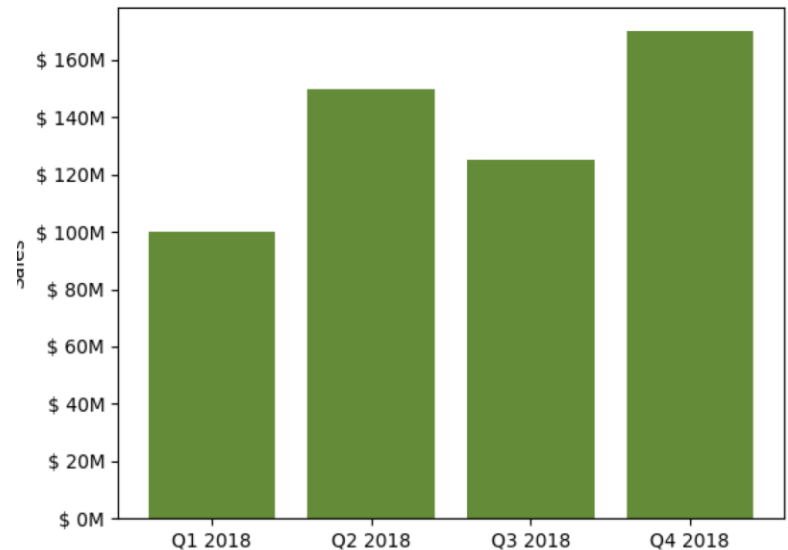
# Charting - Formatting

```

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 # 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()

```

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





# Charting - Subplots

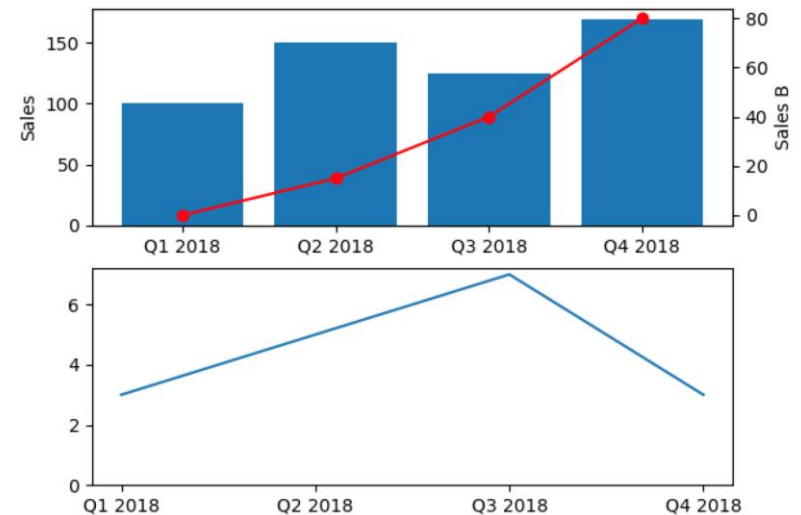
- Multiple charts

2 rows, 1 column, index starting from 1)

```

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 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()

```



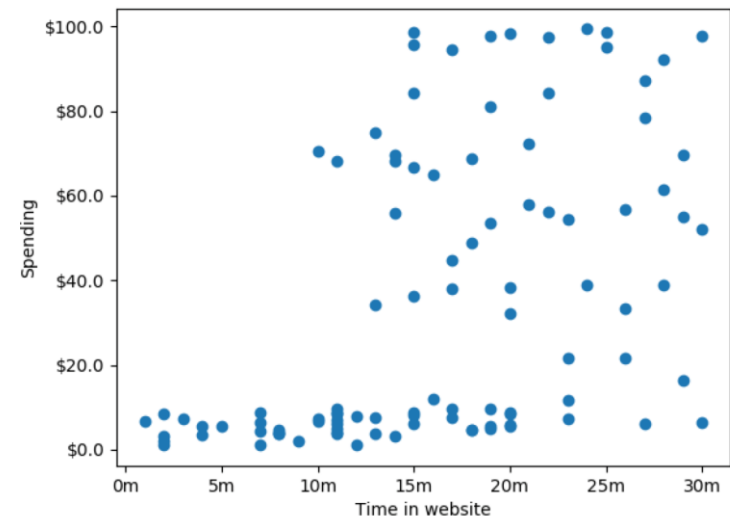
[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.subplot.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplot.html)



# 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

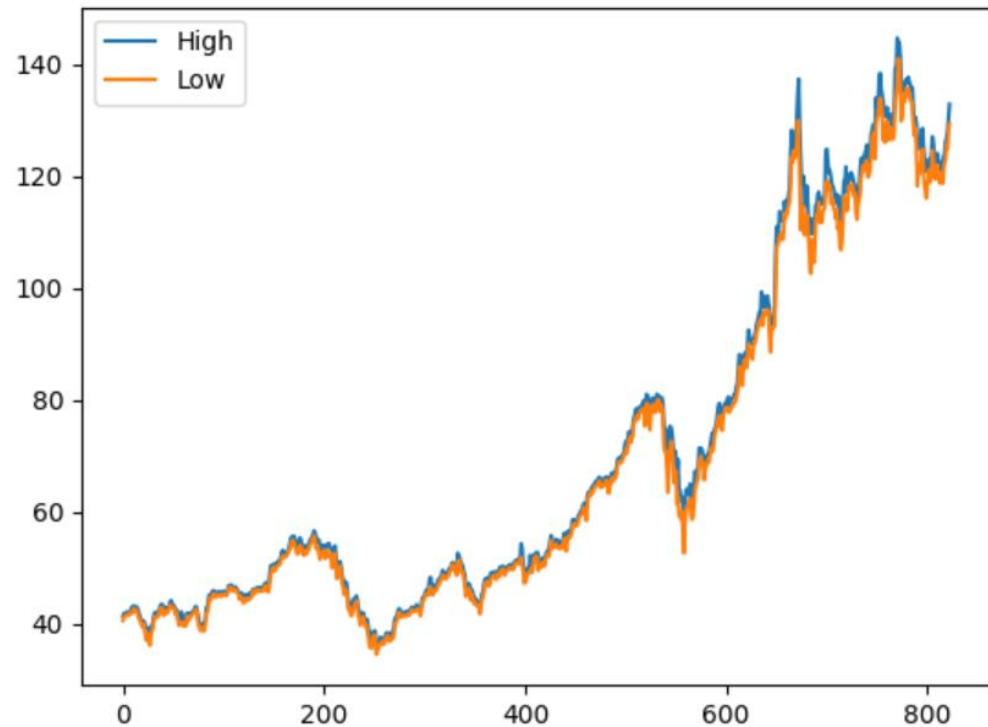




# Use Case: Stock Chart

- Use for algorithmic Trading
  - Plotting of Stock Chart

	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	1/2/2018	40.80647	41.31966	40.59063	41.31007	1.02E+08
3	1/3/2018	41.37482	41.85925	41.23813	41.30288	1.18E+08
4	1/4/2018	41.37723	41.60025	41.26691	41.49474	89738400
5	1/5/2018	41.59306	42.05589	41.49953	41.96716	94640000
6	1/8/2018	41.81128	42.11345	41.71056	41.81128	82271200
7	1/9/2018	41.85925	41.98156	41.58587	41.8065	86336000
8	#####	41.52591	41.79929	41.48754	41.79689	95839600
9	#####	41.86884	42.08467	41.84486	42.03431	74670800
10	#####	42.25013	42.53312	42.12303	42.46836	1.02E+08







# Other Python Libraries

- Play music using winsound
- Generate QR code using qrcode
- Face detection using opencv





# End of Day 2

---

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

Thank you !

QUESTIONS ?

