

# Introductory Programming Using Python

---

**Day 2**

By Seow Khee Wei / Vincent Ng

Republic Polytechnic

• REC

Please change your log in screen name to resemble the registered name for the course



The instructor will take  
attendance

Course Material Link: <https://bit.ly/py-oct20> (Same link as yesterday)



# Trainers

---

**Seow Khee Wei**

**seow\_khee\_wei@rp.edu.sg**



**Vincent Ng**

**vincent\_ng@rp.edu.sg**





# Programme Day Two

---

## Morning

- Read and writing files
- Copying, moving and deleting files and folders
- Working with Excel
- Image Processing

## Afternoon

- Connecting to the Web
- Sending emails
- Creating charts
- Generating PDF



# Schedule

Time	Activity
9.00am – 10.20am	File I/O + File/Folder operations
10.20am – 10.40am	Break
10.40am – 12.00pm	Excel + Images
12.00pm – 1.30pm	Lunch
1.30pm – 2.50pm	Web + email
2.50pm – 3.10pm	Break
3.10pm – 4.30pm	PDF + charts
4.30pm – 5.00pm	Survey / 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

The screenshot shows the Python IDLE interface. At the top, there's a toolbar with 'Search' and 'Stack Data' buttons. Below that is a search bar labeled 'Search:' and a replace bar labeled 'Replace:'. There are checkboxes for 'Case sensitive', 'Whole words', and 'In Selection'. At the bottom of this panel are buttons for 'Prev', 'Next', 'Replace', 'place', and 'option:'. The main window has tabs for 'Debug I/O' and 'Python Shell'. Under 'Python Shell', it says 'Commands execute without debug. Use arrow keys for history.' Below this, there's a command prompt '(>>>)' followed by several lines of code and output:

```

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

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

```



# Write files

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

Pass 'w' to open() to open the file in write mode or 'a' for append mode.



Opening a non-existent file in write or append mode will create that file

Call write() to write a string to a file.

Search Stack Data

Search:

Replace:

Case sensitive  Whole words  In Selection

Prev Ne: replac 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]  
TThis is also another line  
  
TThis is also another line  
Hello world again
```



# 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.

Replace: Options

Case sensitive Whole In Selection

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

3.5.6 |Anaconda, Inc.| (default, Aug 26 2018, 16:30:03)  
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE\_401/final)]  
Python Type "help", "copyright", "credits" or "license" for more information

- `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")
```

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

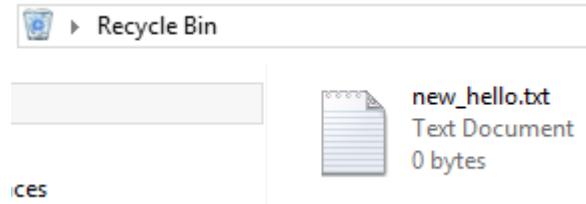
- Install send2trash module using pip.exe  
`pip install send2trash`
- `send2trash.send2trash()` will send a file or folder to the recycling bin

```
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>
```

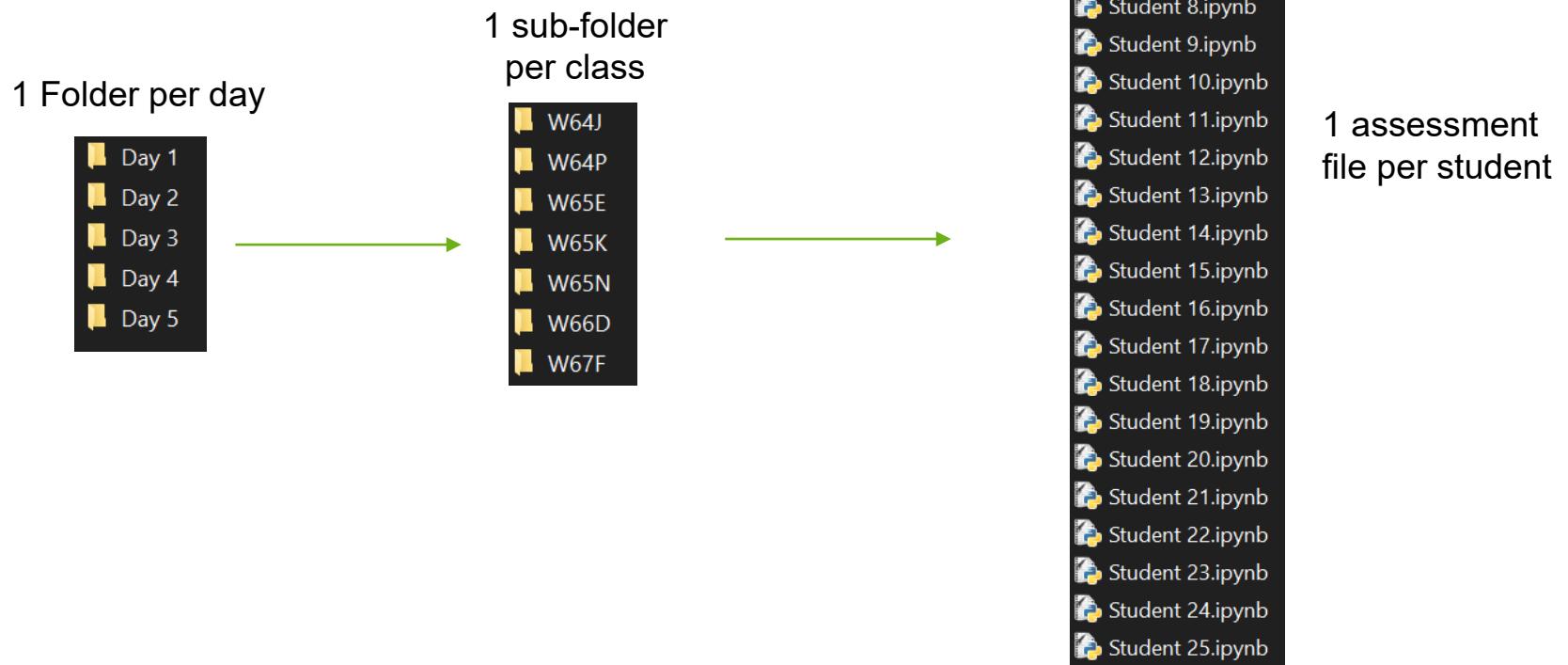
```
>>> import send2trash
>>> send2trash.send2trash("D:\\\\folder2\\\\new_hello.txt")
```





# Use Case Sharing

- Generating 600+ students assessment files, organized by day of week and classes folder
  - 5 Days of week
  - ~7 Classes a day
  - ~22+ students per class





# 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

---

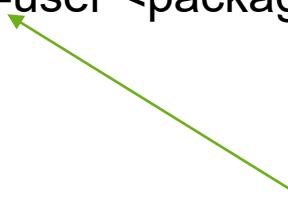
- <https://pypi.org/>
- A repository of software for the Python Programming Language
- Python Installation provides the core libraries needed for the common tasks
  - Additional packages can be found at the website and installed as extension
    - E.g. send2trash, openpyxl, pillow etc
- Installation 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>



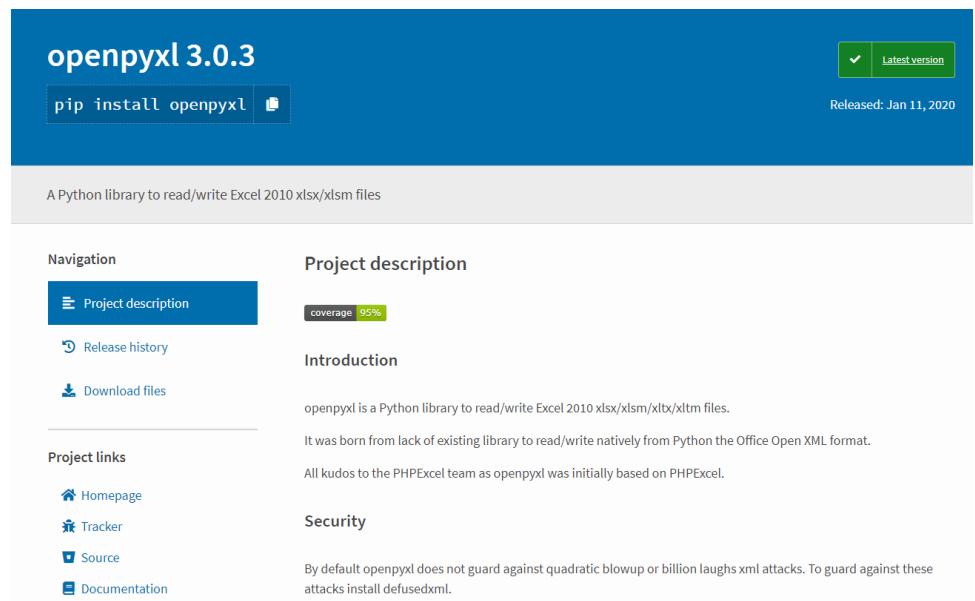
Double-Dash

# 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 green button labeled "Latest version". To its right, it says "Released: Jan 11, 2020". Below the button, there's a "pip install openpyxl" link and a download icon. The main content area has a blue header with the text "openpyxl 3.0.3". Below the header, a sub-header reads "A Python library to read/write Excel 2010 xlsx/xlsm files". The page is divided into two main sections: "Navigation" on the left and "Project description" on the right.

**Navigation**

- Project description
- Release history
- Download files

**Project links**

- Homepage
- Tracker
- Source
- Documentation

**Project description**

coverage 95%

**Introduction**

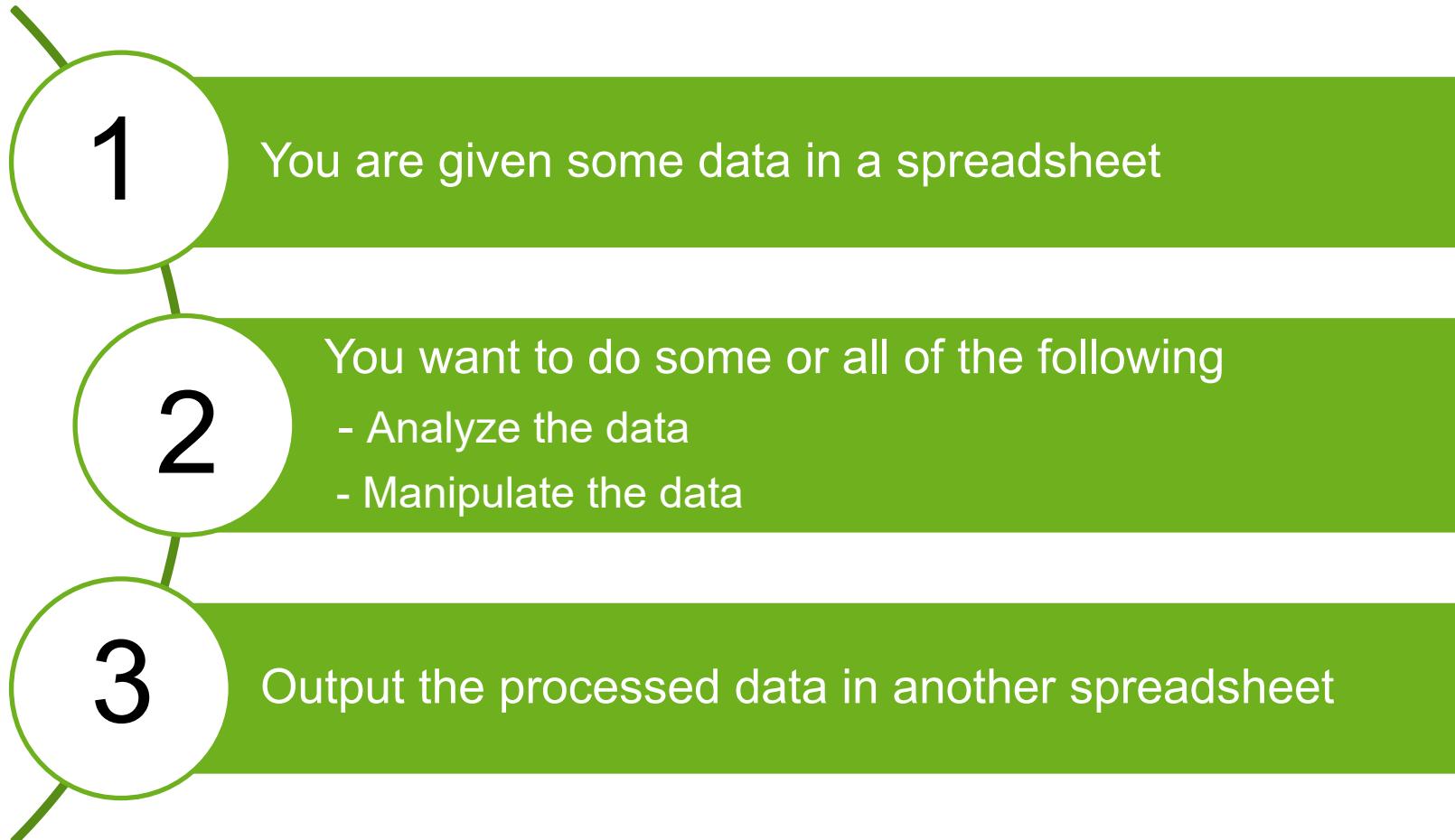
openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xltb/xltm files. It was born from lack of existing library to read/write natively from Python the Office Open XML format. All kudos to the PHPExcel team as openpyxl was initially based on PHPExcel.

**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





# Reading Excel file

1) Import openpyxl

```
import openpyxl
```

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

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

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"]  
  
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) 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']
]

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

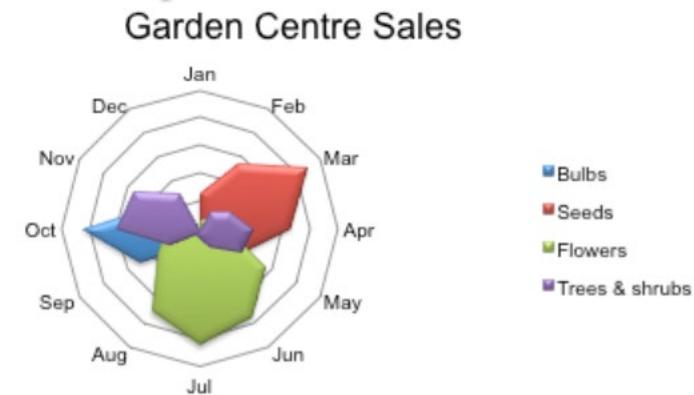
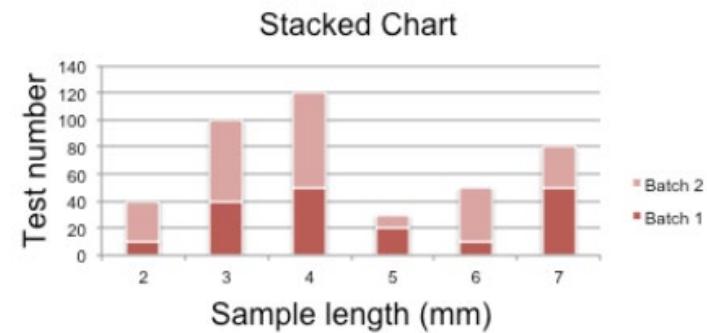
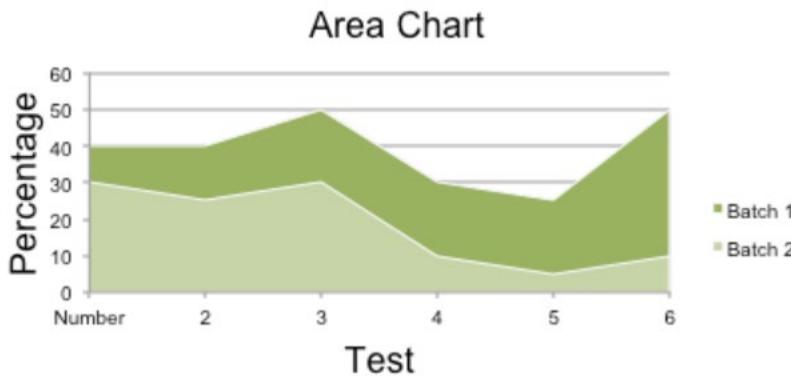
2) Setup colors and styles

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) ← 4) Specify values for title x-axis
and y-axis for the chart
chart.title = "Bar 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

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

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

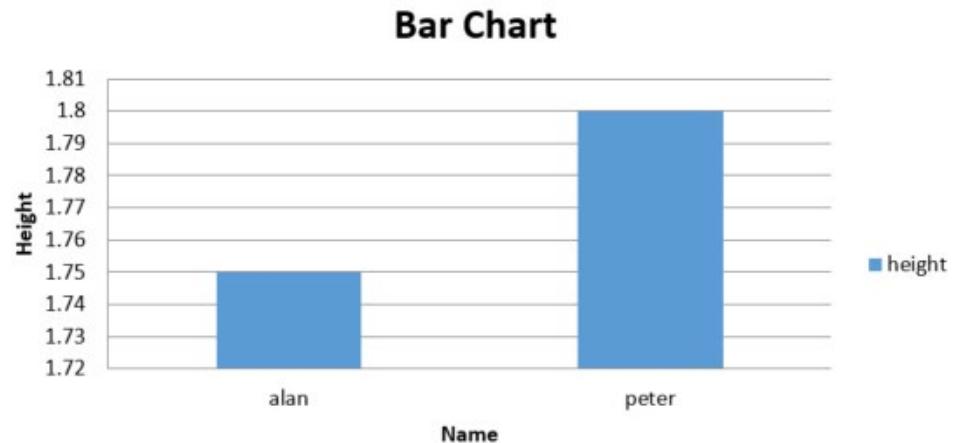
	A	B	C
1	name	weight	height
2	alan	65	1.75
3	peter	70	1.8



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



min\_col = 3 ← height  
Min\_col = 2 ← weight



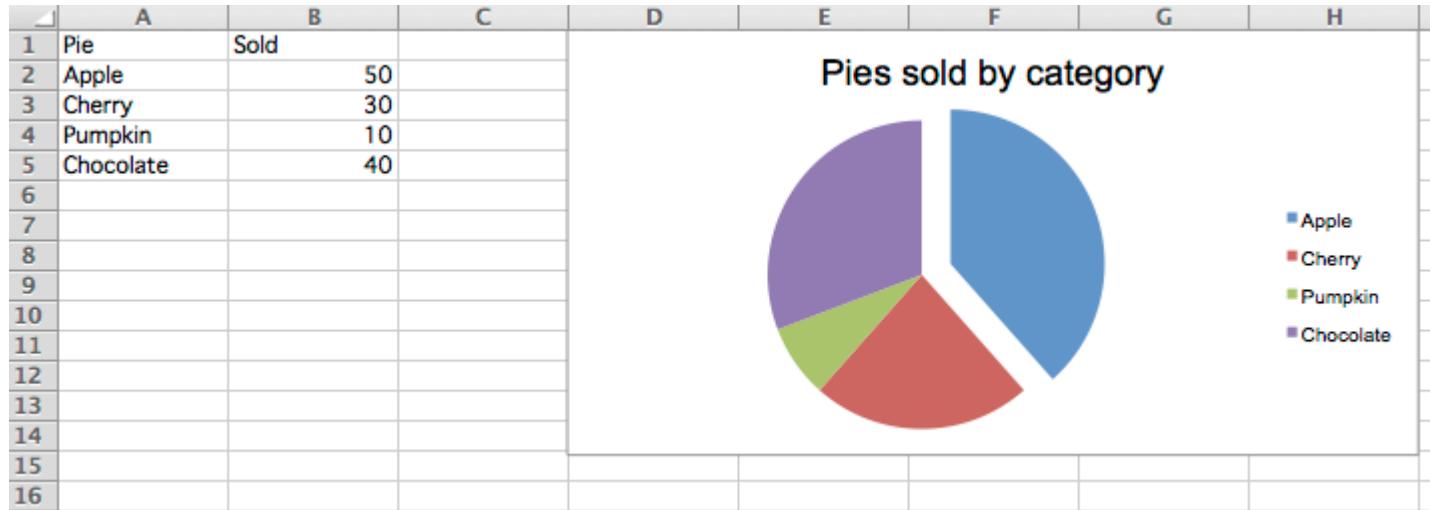


# Exercise

---

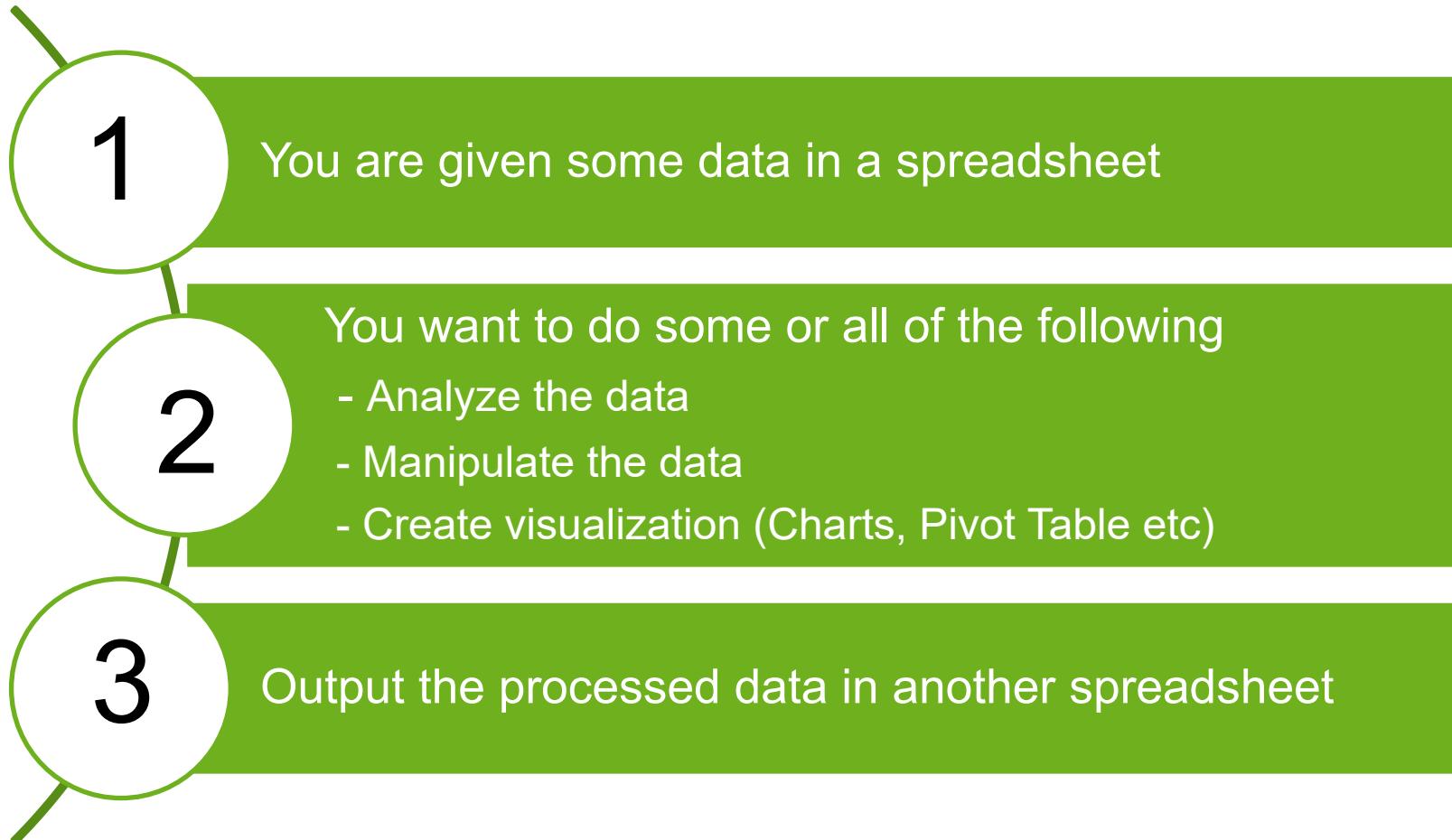
- **Write code to achieve the following:**

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.





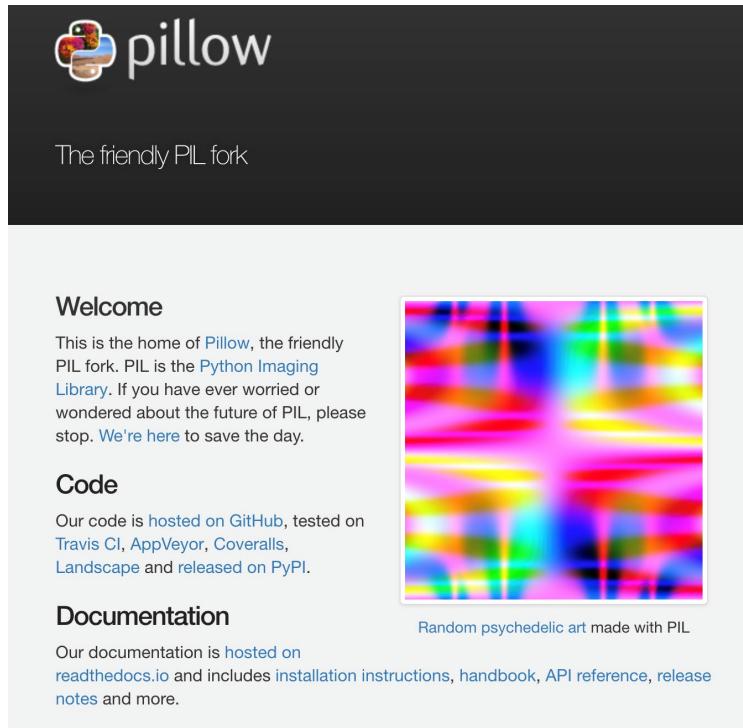
# Typical Workflow for Excel Automation



# 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' icon) and the word "pillow". Below the logo is the tagline "The friendly PIL fork". The main content area includes sections for "Welcome", "Code", and "Documentation". The "Welcome" section contains a paragraph about the project's purpose and a link to a "We're here" page. The "Code" section links to GitHub and other CI services. The "Documentation" section links to the ReadTheDocs site. A central image is a colorful, abstract pattern generated by 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()
```

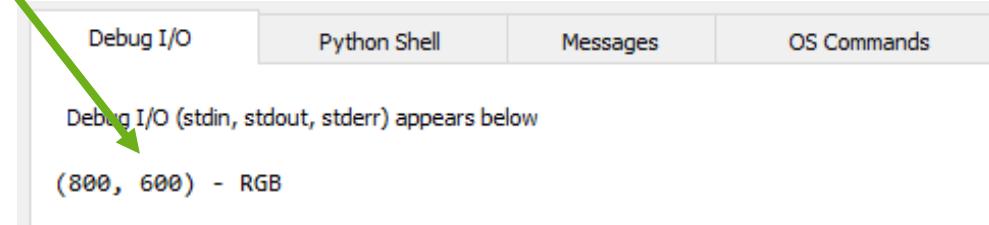


As a start we need to import it:  
`import Image`

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`



```
Debug I/O
Python Shell
Messages
OS Commands

Debug I/O (stdin, stdout, stderr) appears below
(800, 600) - RGB
```

Size: 800 x 600, Mode: RGB

image\_show\_info.py



# 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

---

```
>>> fname1 = "holiday.gif"
>>> fname2 = fname1.split(".")[0] + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

```
>>> fname1 = "holiday.gif"
>>> f, e = os.path.splitext(fname1)
>>> fname2 = f + ".jpg"
>>> print(fname2)
holiday.jpg
>>>
```

Maybe you want to keep all your photos in the same format.

We have some gif files and maybe you would have bmp or png images.

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

fname1		fname2
holiday.gif	->	holiday.jpg

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



# 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
1. Resize all the file to 60 x 90.
1. 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
1. Copy all the files to the renamed folder
1. 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
```

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

Get the required information from the given URL



Economy



Education



Environment



Finance



Health



Infrastructure



Society



Technology



Transport

web\_request.py



# Connecting to the Web

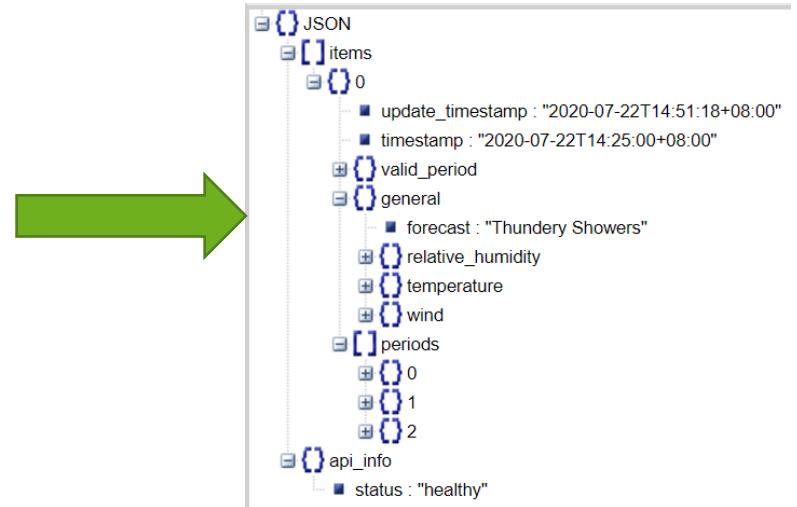
- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

<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": "2020-07-22T12:00:00+08:00", "end": "2020-07-23T12:00:00+08:00"}, "general": {"forecast": "Thunderous Showers", "relative_humidity": {"low": 70, "high": 95}, "temperature": {"low": 22, "high": 30}, "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": "2020-07-22T12:00:00+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"}}, {"start": "2020-07-23T12:00:00+08:00", "end": "2020-07-23T18:00:00+08:00", "regions": [{"west": "Partly Cloudy (Night)", "east": "Partly Cloudy (Night)", "central": "Partly Cloudy (Night)", "north": "Partly Cloudy (Night)"}], "time": {"start": "2020-07-23T12:00:00+08:00", "end": "2020-07-23T18:00:00+08:00"}]}}, {"start": "2020-07-23T18:00:00+08:00", "end": "2020-07-24T06:00:00+08:00", "regions": [{"west": "Partly Cloudy (Night)", "east": "Partly Cloudy (Night)", "central": "Partly Cloudy (Night)", "north": "Partly Cloudy (Night)"}], "time": {"start": "2020-07-23T18:00:00+08:00", "end": "2020-07-24T06:00:00+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)

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

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

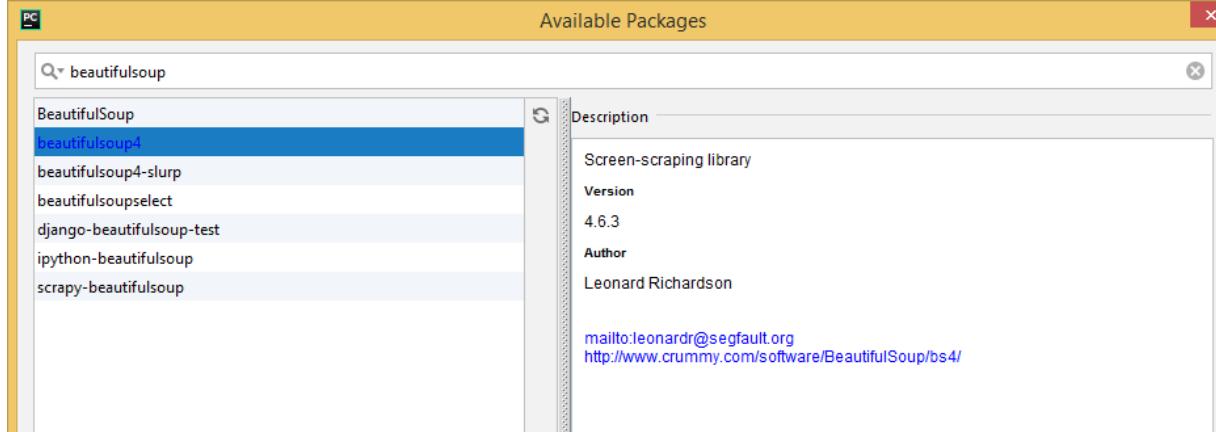


# Connecting to the Web

- 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/living-room/coffee-table/nina-coffee-table-brown.html>

✉ Enquiries: [cs@fortytwo.sg](mailto:cs@fortytwo.sg) Contact Us About Us

**fortytwo**

Search furniture, mattress, home & decor...



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

Home > Living Room Furniture > Coffee Tables > Nina Coffee Table (Brown)



## Nina Coffee Table (Brown)

★★★★★ 190 customer reviews

S\$79.00

**S\$49.90**

Warranty: 1-Year

Standard Delivery



Earliest by  
Saturday, 10  
October 2020

Choose your preferred  
delivery date at  
checkout

42EXPRESS



Today  
(Charges Apply)

Get this product  
delivered by 6PM, 08  
October 2020

Qty: 1

Add to Cart

Heart Add to Wishlist  
Email to a Friend

100 Day Free Returns

Cutter Free Assembly Included

Thumbs Up 300k+ Satisfied Customers

List 0% Instalment Plans Available



# Connecting to the Web

- Get the url

<https://www.fortytwo.sg/living-room/coffee-table/nina-coffee-table-brown.html>

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

The screenshot shows a product page for the "Nina Coffee Table (Brown)" on the fortytwo website. The page features a large image of the dark brown wicker-style coffee table. To the right of the image, the product name "Nina Coffee Table (Brown)" is displayed in bold. Below the name is a price tag showing "\$49.90". Further down, there's a section for "190 customer reviews" with a 5-star rating icon. On the left side, there are smaller images of the table from different angles. On the right side, there are delivery options: "Standard Delivery" (with a box icon) and "42EXPRESS" (with a truck icon). A "Warranty: 1-Year" section is also present.

The screenshot shows the browser's developer tools with the element inspector open. The element being inspected is a price box containing "\$49.90". A context menu is open over this element, with the "Copy selector" option highlighted by a red arrow. The background shows the HTML code for the page, which includes various CSS classes and IDs for styling the product listing.



# Connecting to the Web

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

Nina Coffee Table (Brown)

5 ★★★★ 190 customer reviews

**\$49.90**

Warranty: 1-Year

Standard Delivery OR 42EXPRESS

Earliest by Saturday, 10 October 2020

Today (Charges Apply)

Get this product delivered by 6PM, 09 October 2020

Choose your preferred delivery date at checkout

Add to Wishlist Email to a Friend

100 Day Free Returns

Free Assembly Included

300+ Satisfied Customers

0% Instalment Plans Available

Copy selector

```

1 #https://www.fortytwo.sg/living-room/coffee-table/nina-coffee-table-brown.html
2
3 import bs4
4 import requests
5
6 requestObj = requests.get("https://www.fortytwo.sg/living-room/coffee-table/nina-coffee-table-brown.html")
7 requestObj.raise_for_status()
8 soup = bs4.BeautifulSoup(requestObj.text, 'html.parser')
9
10 elements = soup.select("#product-price-41537 > span") # $49.90
11 print("Current Price: " + elements[0].text)
12
13 elements = soup.select("#old-price-41537") # $79.00
14 print("\nOld Price: " + elements[0].text)

web_scrap.py
  
```

Debug I/O    Exceptions    Python Shell    M

Debug I/O (stdin, stdout, stderr) appears below

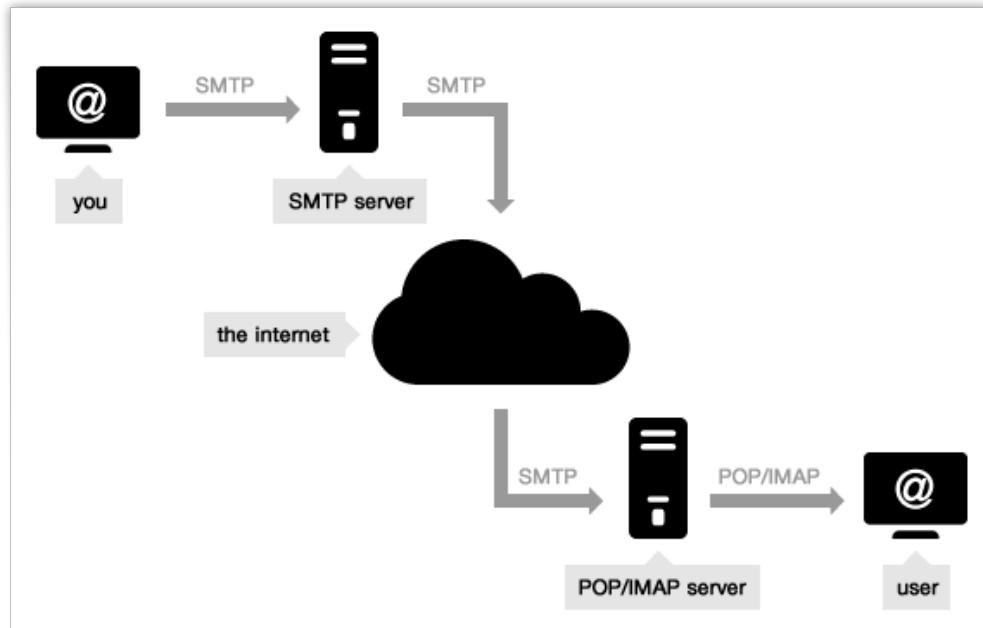
Current Price: S\$49.90

Old Price: S\$79.00

# Email Automation with Python



# Send Email



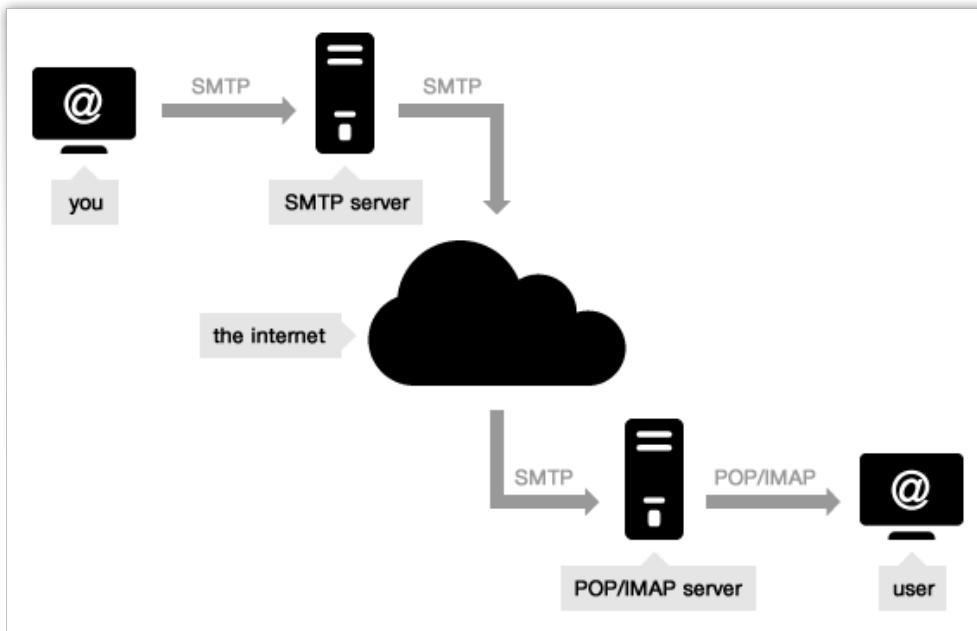
- SMTP (Simple Mail Transfer Protocol) is used for sending and delivering from a client to a server via port 25, 465 or 587: it's the **outgoing server**.
- IMAP and POP are two methods to access email. IMAP is the recommended method when you need to check your emails from several different devices,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

➤ 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)

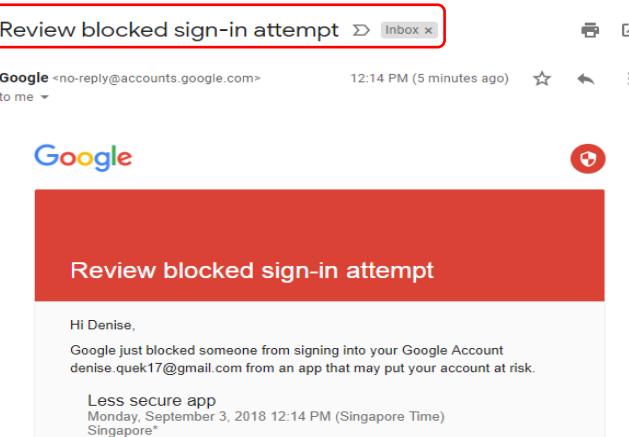
smtpObj.sendmail(sender_email_address, receiver_email_address, email_title_content)
print("Email sent successfully...")

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**

Your app password for Windows Computer

[View 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

Screenshot of the Google Account Security settings page.

The left sidebar shows navigation links: Home, Personal info, Data & personalisation, **Security** (highlighted with a red box), People & sharing, Payments & subscriptions, Help, and Send feedback.

The main content area is titled "Security" and describes settings to keep the account secure.

### Security issues found

Protect your account now by resolving these issues



[Secure account](#)

### Signing in to Google



Last changed 10 Jan 2018

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



Show all X



# 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
```

# 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 [sPDF](#) LGPL3 PHP version from [Ian Back](#)
- Improved installers (setup.py, py2exe, PyPI) support
- Barcode 128f5 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



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

## 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>



[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

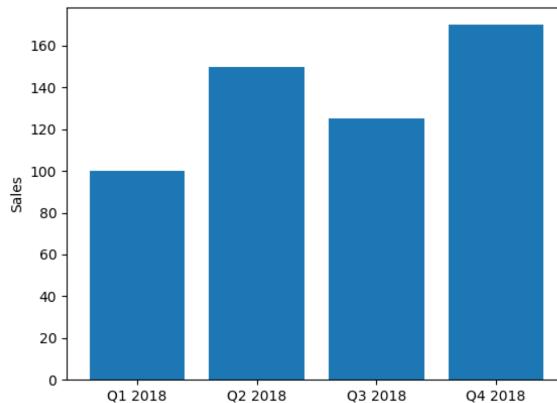
`pip install matplotlib`

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)



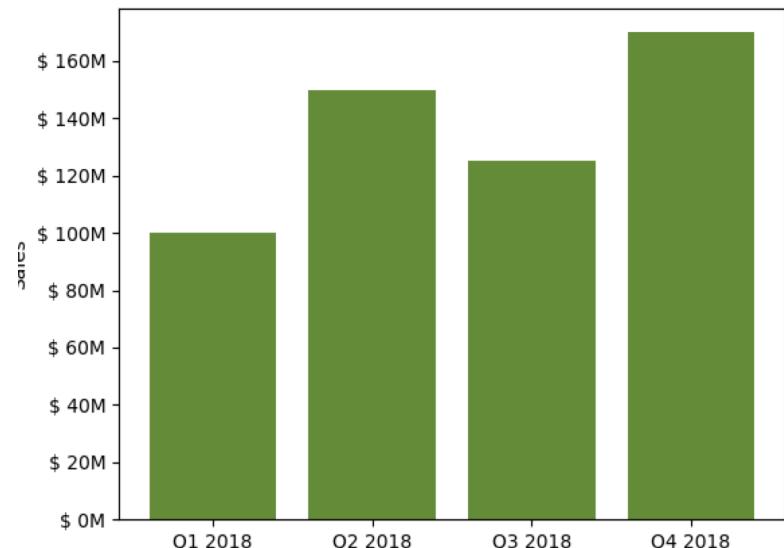
# Charting - Formatting

```

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

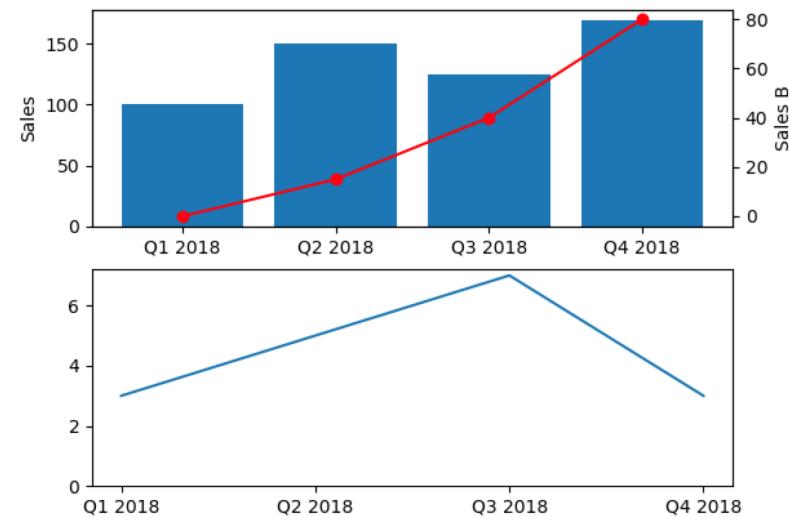
```

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

2 rows, 1 column, index starting from 1)



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

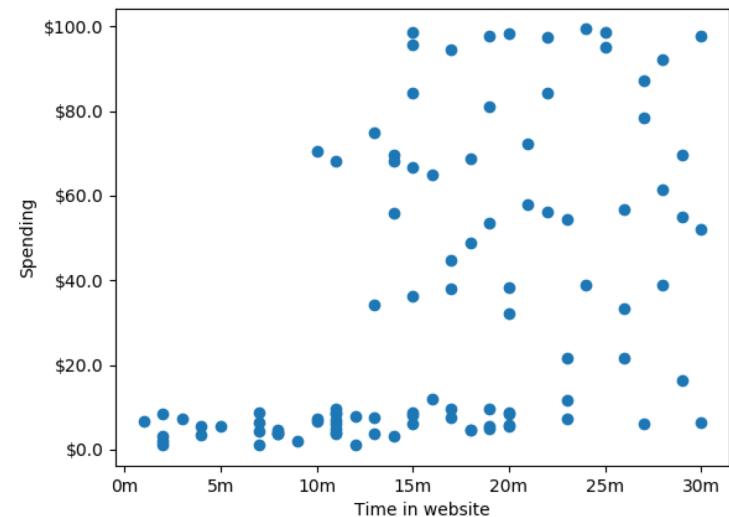


# Charting – Scatter Plot

```
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.

Thank you !

**QUESTIONS ?**





# TRAQOM survey

---

- **Link sent by SSG via Email or SMS**
- **Message looks like:**  
*“You’ve completed a course supported by SSG. PIs complete the TRAQOM survey at this link:  
<https://traqom.azcentric.com/xxxxx>”*
- **Do let us know if you have not received**
- **Please complete the survey found in the link**



# 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:

<https://diveintopython3.problemsolving.io/>



# Where to go from here ?

---

**MOOC:**  
DataCamp  
<https://www.datacamp.com/>

The screenshot shows the DataCamp website with a purple header banner that says "Subscribe now. Save 50% on DataCamp and skill up. Offer ends in 04 days 04 hrs 56 mins 51 secs". Below the banner, the search bar shows "python". The main content area displays "57 results for 'python'" in white text. There are six course cards visible in two rows of three. The courses are: "Intro to Python for Data Science" (by Filip Schouwenaars), "Intermediate Python for Data Science" (by Filip Schouwenaars), "Python Data Science Toolbox (Part 1)" (by Hugo Bowne-Anderson), "Deep Learning in Python" (by Filip Schouwenaars), "Supervised Learning with scikit-learn" (by Filip Schouwenaars), and "pandas Foundations" (by Hugo Bowne-Anderson). Each card includes a thumbnail of the instructor, their name, title, and a brief description.

**Edx**  
<https://www.edx.org/learn/python>

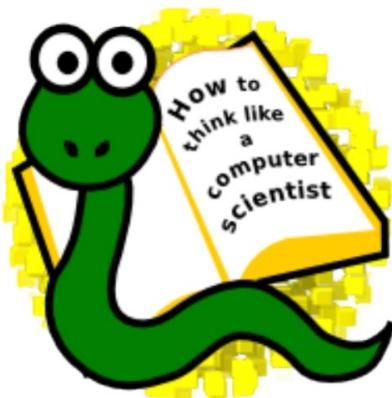
**Udemy**  
<https://www.udemy.com/topic/python/>

The screenshot shows the edX website with a banner at the top that reads "Accelerate your future. Learn anytime, anywhere." Below the banner is a search bar with the placeholder "What do you want to learn?". At the bottom of the page, there is a large promotional banner for Cyber Monday. The banner features the text "THE COUNTDOWN IS ON! Get 15% off your purchase." and a yellow button labeled "Start Exploring". To the right of the banner, the words "CYBER MONDAY" are written vertically in large, bold, yellow letters. Logos for various partner universities are displayed along the bottom edge of the page, including MIT, Harvard, Berkeley, The University of Texas System, The Hong Kong Polytechnic University, and The University of British Columbia.



# Where to go from here ?

---



*How to think like a computer scientist* 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.

*How to think like a computer scientist* is a Free Book. It is available under the [GNU Free Documentation License](#), which means that you are free to copy, distribute, and modify it, as long as derivative works of the document must themselves be free in the same sense.

<https://buildmedia.readthedocs.org/media/pdf/howtothink/latest/howtothink.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.



+ 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)