

# Introductory Programming Using Python

---

**Day 2**

By Seow Khee Wei/Tan Kok Cheng

Republic Polytechnic



# Programme Day Two

---

## Morning

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

## Afternoon

- Image processing
- Connecting to the Web
- Sending emails
- Telegram bot
- Generating PDF
- Creating charts



# File Paths

---

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

For example,

/home/example\_user/example\_directory **or**  
C:/system32/cmd.exe

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

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

For example,

example\_directory.

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



# Read files

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

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

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

Call read() to read the contents of a file

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

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

The screenshot shows a Python IDE interface. On the left, there's a toolbar with 'Search' and 'Stack Data' buttons, and search/replace fields for 'Search:' and 'Replace:' with options like 'Case sensitive', 'Whole words', and 'In Selection'. Below that are buttons for 'Prev', 'Next', 'Replace', 'Place', and 'Option'. The main area has tabs for 'Debug I/O' and 'Python Shell'. Under 'Python Shell', it says 'Commands execute without debug. Use arrow keys for history.' and shows the following session:

```
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: 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
```



# 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 se 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*).

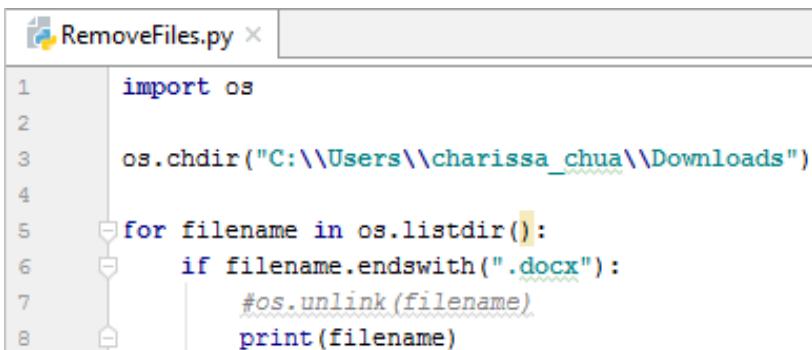
<https://docs.python.org/3/library/shutil.html>



# Deleting files

```
import os  
  
print(os.getcwd())  
  
# delete directory (can only delete empty folder)  
#os.rmdir("folder3")  
  
import shutil  
# delete directory (with content)  
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



```
RemoveFiles.py ×  
1 import os  
2  
3 os.chdir("C:\\\\Users\\\\charissa_chua\\\\Downloads")  
4  
5 for filename in os.listdir():  
6     if filename.endswith(".docx"):  
7         #os.unlink(filename)  
8         print(filename)
```

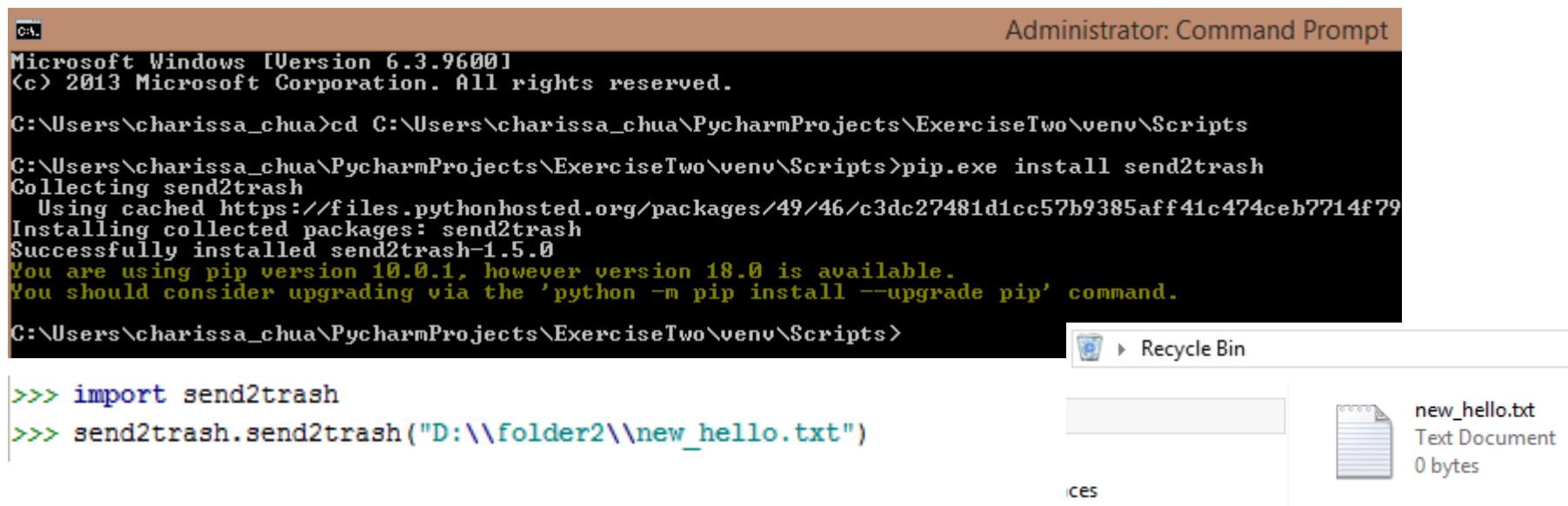


Deleting can be dangerous, so do a dry run first



# send2Trash module

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



```
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>
```

Recycle Bin

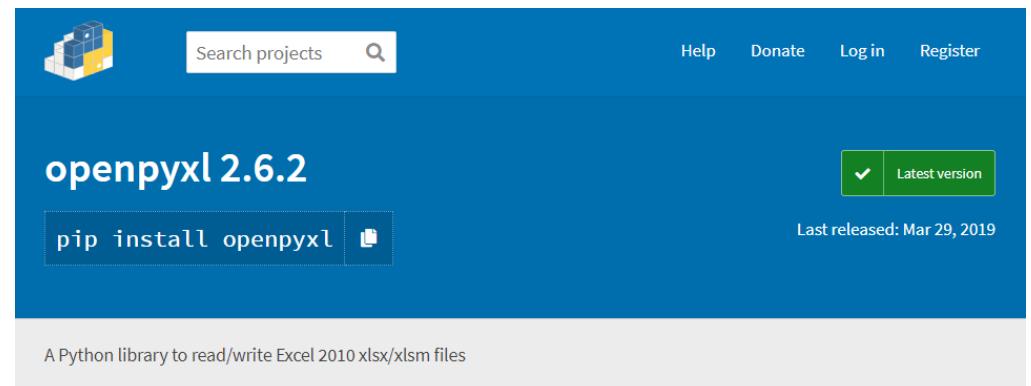
ces

new\_hello.txt  
Text Document  
0 bytes



# Working with Excel

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



A screenshot of the PyPI project page for `openpyxl 2.6.2`. The page features a Python logo icon, a search bar, and navigation links for Help, Donate, Log in, and Register. A prominent green button labeled "Latest version" with a checkmark is visible. Below the header, there's a "pip install openpyxl" button with a download icon. To the right, it says "Last released: Mar 29, 2019". A brief description states: "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/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.



# Reading Excel file

```
1 import openpyxl ← 1) Import openpyxl
2
3 workbook = openpyxl.load_workbook("students_attendance.xlsx")
4 sheet=workbook["Sheet1"]
5
6 max_row = sheet.max_row
7 max_column = sheet.max_column
8
9 #loop through every row
10 for i in range(1,max_row+1):
11
12     #read cell
13     attendance = sheet.cell(row=i, column=3).value ← 2) Load Excel content into
14
15     #check attendance
16     if attendance == "Absent":
17         name = sheet.cell(row=i,column=1).value ← 3) Get the active worksheet
18         email = sheet.cell(row=i,column=2).value ← named "Sheet1"
19         print(name + " is absent") ← 4) Get the number of
                                rows and columns
                                ← 5) Use For loop to go
                                through every row
                                ← 6) Extract the status at
                                Column C to check for
                                attendance
```



# Update Excel file

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



# Create Excel file

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

1) Import openpyxl  
2) Create new workbook  
3) Get default sheet  
4) Create dataset - a list of lists  
5) Insert value into cells  
6) Create a new sheet  
7) Insert value into cells  
8) Save the spreadsheet



# Format Excel

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

Import necessary functions

Setup colors and styles

Loop through cell and set properties



# Working with CSV file

---

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



# What is CSV format

- The same data when viewed with Excel ...

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



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

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

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



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

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



# Reading CSV file

1) Load CSV library

```
1 import csv  
2  
3 readerFileHandle = open("W65Z.csv", "r", newline='')  
4 reader1 = csv.reader(readerFileHandle)  
5 #using for loop to retrieve from the CSV file line by line  
6 for row in reader1:  
7     print(row)  
8  
9 readerFileHandle.close()
```

5) Close the file  
(remove the link)

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

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

4) For-loop retrieve each item in **reader1** (a list) into the loop variable **row** and display it.  
*(Note: Each line in the file becomes a list of values)*

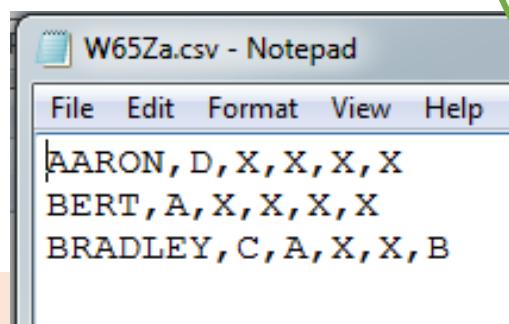


# Writing CSV file

1) Load CSV library

```
1 import csv  
2  
3 writerFileHandle = open("new.csv", "w", newline='')  
4 writer1 = csv.writer(writerFileHandle)  
5 row1 = ["Arron", "D", "X", "X", "X", "X"]  
6 row2 = ["Bert", "A", "X", "C", "B", "X"]  
7 row3 = ["Bradley", "C", "A", "C", "X", "X"]  
8 rowlist = [row1, row2, row3]  
9  
10 for row in rowlist:  
11     writer1.writerow(row)  
12  
13 writerFileHandle.close()
```

6) Close the file  
(Remove the link)



2) Create (if new) & Open the file named "W65z\_new.csv" for writing (indicated by 'w' argument). **writerFileHandle** links the file to the program.

3) "**writer1**" stores content to be written to the file in CSV format

4) **rowlist** stores the content to be written to the CSV file. (**rowlist** is a list containing lists as items)

5) For-loop retrieves each item from **rowlist** into loop variable **row** → **row** (a list) is written as 1 csv formatted line into the file.

csv\_write.py



# Exercise

---

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



# Image Processing

The screenshot shows the official website for Pillow, a Python Imaging Library fork. The header features the Pillow logo (a stylized 'P' with a colorful cross) 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 history, mentioning the Python Imaging Library and its future. The "Code" section links to GitHub and Travis CI. The "Documentation" section links to readthedocs.io. A central image is a vibrant, abstract pattern of overlapping colored rectangles, labeled "Random psychedelic art made with PIL".

**Welcome**

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

**Code**

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

**Documentation**

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

Random psychedelic art made with PIL

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

Install using the following command:  
**pip install Pillow**

The documentation is at:

<http://pillow.readthedocs.io/en/5.1.x/handbook/index.html>



# Image Processing

---

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

```
import os
from PIL import Image

filename = "img/clungup.jpg"

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

im.show()
```



# Image Processing

---

```
import os  
from PIL import Image  
  
filename = "img/clungup.jpg"  
  
im = Image.open(filename)  
  
im.show()
```

Let's explore what Pillow can do.

As a start we need to import it:

**import Image**

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

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



# 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()

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

Pillow has many conversion and filters, we will use some of them. But if you need more, go ahead :

<http://pillow.readthedocs.io/en/5.1.x/handbook/index.html>

To use filters we need to extend our import:

```
from PIL import Image,
ImageFilter
```

The way you can apply filters is :  
`out = im.filter(ImageFilter.BLUR)`  
Try some different filters!



# Image processing - filters

---



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



```
image = ImageOps.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

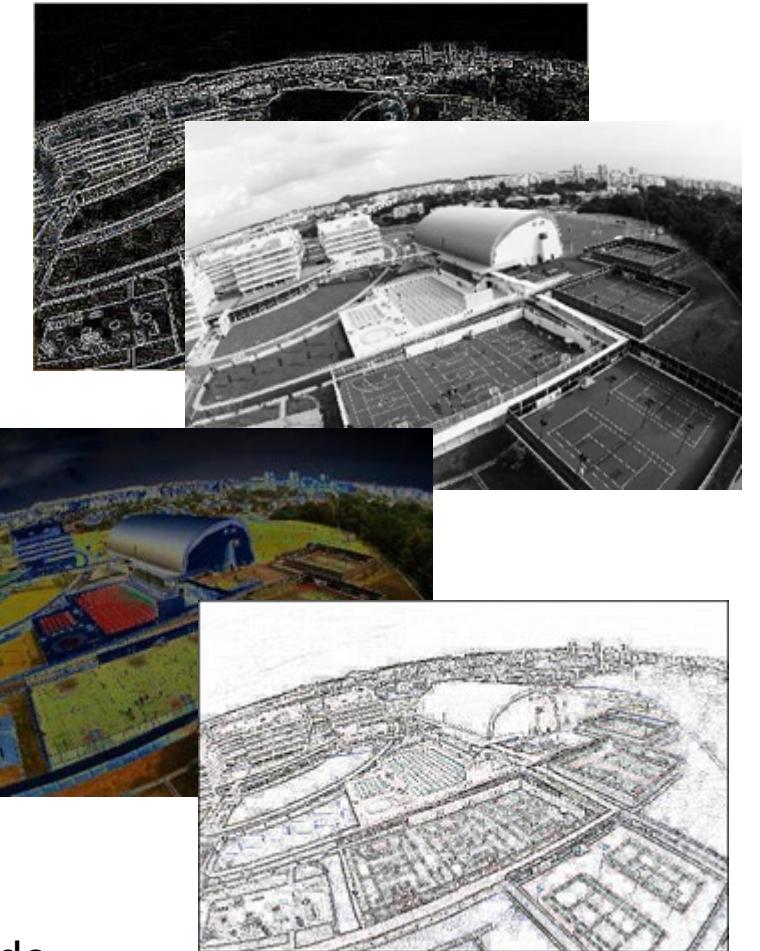
filename = "img/clungup.jpg"

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)
```

We can do a lot with

images.

Let's look at rotation and  
flipping

Rotating the image

```
out = im.transpose(Image.ROTATE_90)  
out = im.transpose(Image.ROTATE_180)  
out = im.transpose(Image.ROTATE_270)
```

Try to rotate and flip your  
images.

Contrast

First add ImageEnhance to our imports:

```
from PIL import Image, ImageFilter, ImageEnhance
```

Then:

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

Another cool effect is to  
make it 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.





# Image processing – Watermark

```
1 from PIL import Image  
2  
3 def watermark(im, mark, position):  
4     layer = Image.new("RGBA", im.size, (0,0,0,0))  
5     layer.paste(mark, position)  
6     return Image.composite(layer, im, layer)  
7  
8 im = Image.open("img\\clungup.jpg")  
9 mark = Image.open("img\\watermark.png")  
10 mark = mark.resize((100,100))  
11  
12 out = watermark(im, mark, (0,50))  
13 out.show()  
|
```

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:



# 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")
```



# Batch Rename

---

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

```
import os
import shutil

files = os.listdir('img')

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



# Connecting to the Web

- requests – download files and web pages from the Web

## Install requests module

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

Get the required information from the given URL

The screenshot shows the homepage of Data.gov.sg. The background features a night-time photograph of a city skyline with illuminated skyscrapers and a highway in the foreground. The main heading 'Data.gov.sg' is at the top center. Below it is a search bar with the placeholder 'Search Singapore's Public Data'. To the right of the search bar is a magnifying glass icon. A prominent orange button labeled 'View All Datasets' with a camera icon is positioned below the search bar. At the bottom of the page, there is a navigation bar with eight categories: Economy (bar chart icon), Education (graduation cap icon), Environment (tree icon), Finance (coins icon), Health (heart icon), Infrastructure (building icon), Society (two people icon), Technology (laptop icon), and Transport (bus icon). Each category has a small description text below its respective icon.



# Connecting to the Web

---

```
1 import requests  
2  
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"  
4 req = requests.get(url)  
5  
6 try:  
7     req.raise_for_status()  
8  
9     playFile = open("downloadedFile.txt", 'wb')  
10    for chunk in req.iter_content(100000):  
11        print(chunk)  
12        playFile.write(chunk)  
13    playFile.close()  
14  
15 except Exception as e:  
16     print("There was a problem: %s" % (e))  
17
```

- Use `requests.get()` to get web content from specified URL
- Use `raise_for_status()` to ensure that download is successful before we continue
- Call `open()` with "wb" to create a new file in write binary mode
- Loop over the Response object using `iter_content()`
- Call `write()` on each iteration to write the content to the file
- Remember to close the file



# Connecting to the Web

- File will be saved in "downloadedFile.txt" (in the same folder as your program)

```
1 import requests  
2  
3 url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"  
4 req = requests.get(url)  
5  
6 try:  
7     req.raise_for_status()  
8  
9     playFile = open("downloadedFile.txt", 'wb')  
10    for chunk in req.iter_content(100000):  
11        print(chunk)  
12        playFile.write(chunk)  
13    playFile.close()  
14  
15 except Exception as e:  
16     print("There was a problem: %s" % (e))  
17
```



downloadedFile.txt

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



# Connecting to the Web

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

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

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

The screenshot shows a JSON viewer interface with the following structure:

- JSON
  - area\_metadata
    - name: Ang Mo Kio
    - label\_location
      - latitude: 1.375
      - longitude: 103.839
  - items
    - valid\_period
      - start: 2019-03-08T18:30:00+08:00
      - end: 2019-03-08T20:30:00+08:00
    - forecasts
      - area: Ang Mo Kio
      - forecast: Partly Cloudy (Night)



# Connecting to the Web

- To work with JSON data, import json first
- Use json.loads() to load the data in JSON format
- Extract and retrieve the required data

```
1  import json
2  import requests
3
4  url = "https://api.data.gov.sg/v1/environment/2-hour-weather-forecast"
5  req = requests.get(url)
6
7  data = json.loads(req.text)
8
9  forecasts = data["items"][0]["forecasts"]
10
11 for forecast in forecasts:
12     area = forecast["area"]
13     weather = forecast["forecast"]
14     print(area + ": " + weather)
```



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

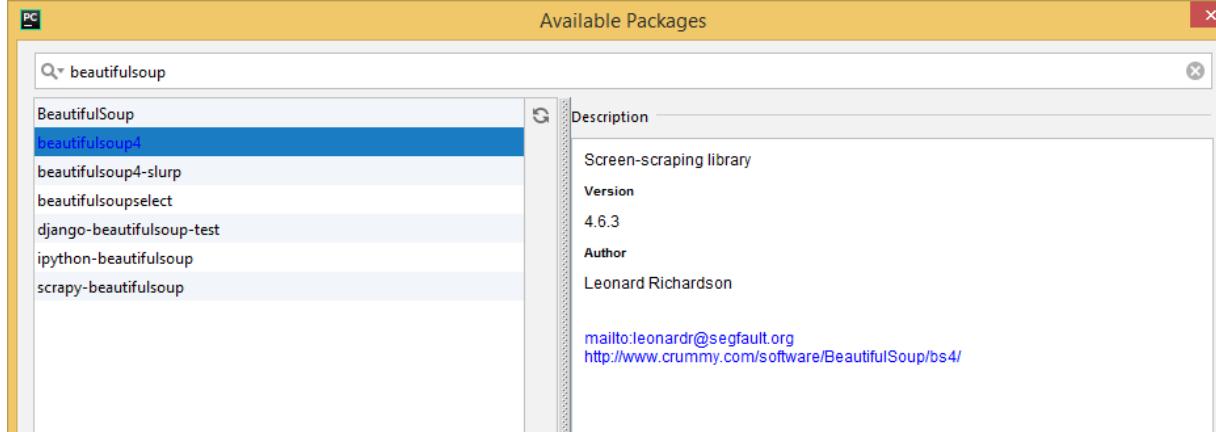


# Connecting to the Web

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

Web Scraping – download and process Web content

- Install Beautiful Soup 4 - **pip install beautifulsoup4**





# Connecting to the Web

- What's the URL?

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

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

The screenshot shows a product page for a 'Landon Regular Dining Table Coffee'. The page includes a search bar, user profile, and shopping cart icon. A navigation menu at the top lists categories like Furniture, Bedding & Mattresses, Décor | Essentials, Kitchen | Dining, Lightings | Fans, and Sale. The product details show a dark wood dining table with a rectangular top and four legs. The price is listed as S\$69.90. Other features include a 100 Day Free Returns policy and Free Assembly Included. The page also shows a 'Warranty: 1 Year' and delivery options like Standard Delivery.

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

Qty: 1

Add to Cart

Add to Wishlist

Email to a Friend

Warranty: 1 Year

Standard Delivery

42EXPR

Free Assembly Included

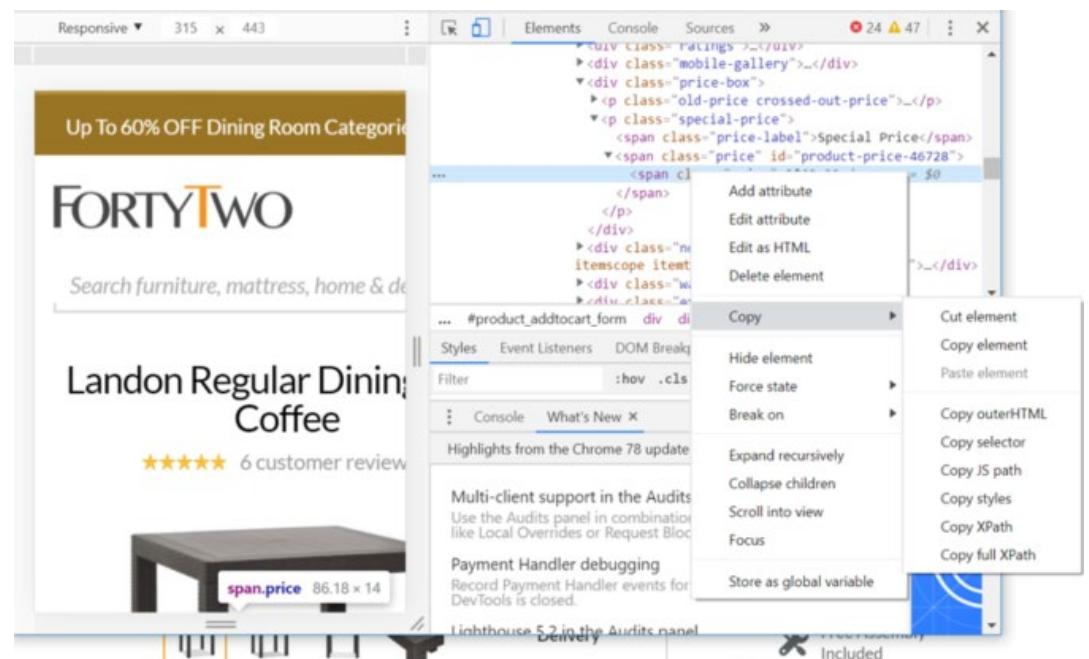


# Connecting to the Web

- 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

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

The screenshot shows the Chrome DevTools Elements panel. A specific element in the DOM tree has been selected, and a context menu is open from the right-clicked item. The 'Copy' submenu is expanded, and 'Copy selector' is the option being highlighted.

```
import bs4
import requests

requestObj = requests.get("https://www.fortytwo.sg/dining/dining-tables/landon-regular-dining-table-coffee.html")
requestObj.raise_for_status()
soup = bs4.BeautifulSoup(requestObj.text, 'html.parser')
elements = soup.select("#product-price-46728")
print(elements[0].text)
```



Debug I/O    Python Shell    Messages    OS Commands

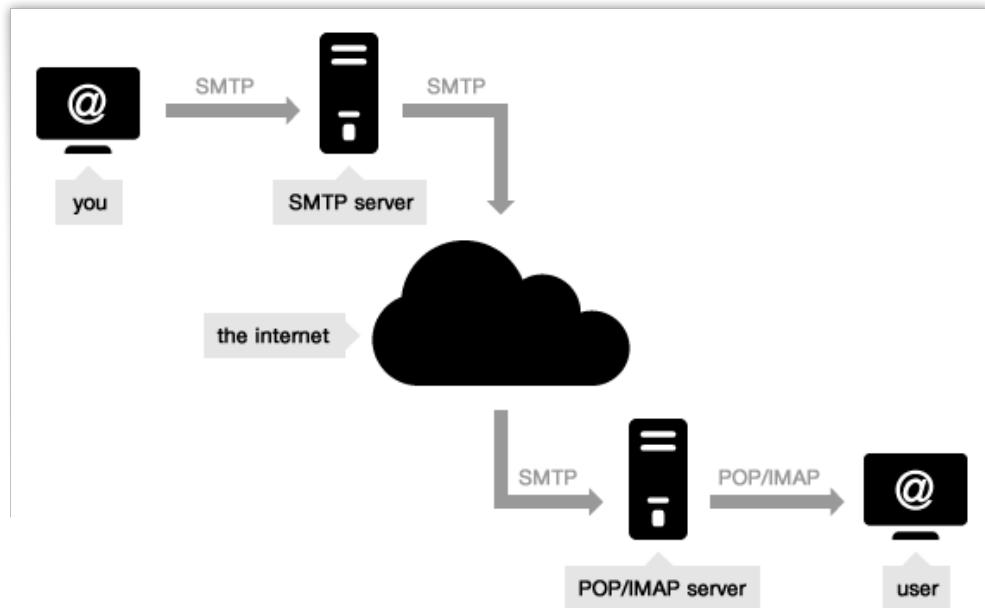
Debug I/O (stdin, stdout, stderr) appears below

S\$69.90

web\_scrap.py



# Send Email

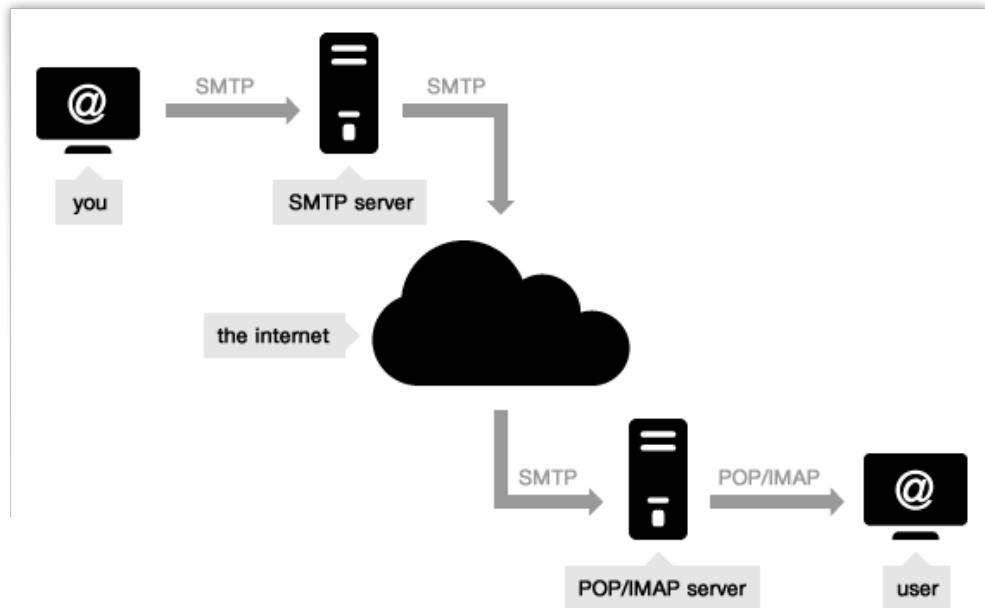


- SMTP (Simple Mail Transfer Protocol) is used for sending and delivering from a client to a server via port 25: it's the **outgoing server**.
- IMAP and POP are two methods to access email. IMAP is the recommended method when you need to check your emails from several different devices, such as a phone,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

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



# Send Email



- **Note:** The SMTP servers used when you send your emails- Hotmail, Gmail , Yahoo Mail – are **shared among users**
- Common providers establish some **strict limits** on the number of emails you can send (e.g. Yahoo's restriction is 100 emails per hour).
- If you plan to send a bulk email or set up an email campaign you should opt for a professional outgoing email server like turboSMTP,
- which guarantees a controlled IP and ensure that all your messages reach their destination.



# Send Email using Gmail

---

Incoming Mail (IMAP) Server	imap.gmail.com Requires SSL: Yes Port: 993
Outgoing Mail (SMTP) Server	smtp.gmail.com Requires SSL: Yes Requires TLS: Yes (if available) Requires Authentication: Yes Port for SSL: 465 Port for TLS/STARTTLS: 587
Full Name or Display Name	Your name
Account Name, User name, or Email address	Your full email address
Password	Your Gmail password



# Send Email using Gmail

---

- Import `smtplib` module
- Specify Gmail email & password, receiver's email address, email title & content
- Connect to SMTP server using Port 587
- Call `starttls()` to enable encryption for your connection
- Login using email and password
- Call `sendmail()`
- Call `quit()` to disconnect from the SMTP server

```
import smtplib

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

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

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

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

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

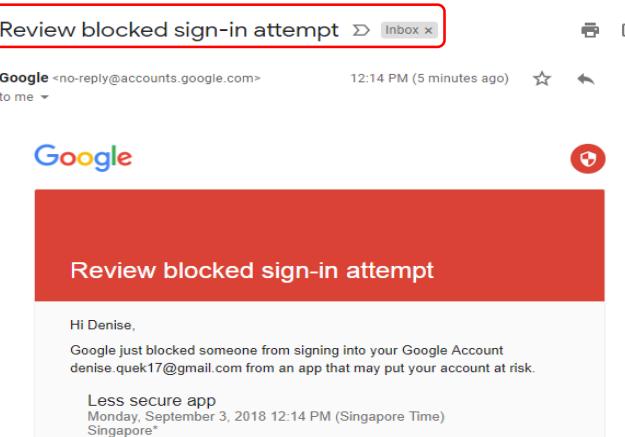
smtpObj.sendmail(sender_email_address, receiver_email_address, email_title_content)
print("Email sent successfully...")

smtpObj.quit()
```



# Send Email using Gmail

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



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

Process finished with exit code 1



# 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

[securesally:Windows Computer](#)

**How to use it**

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

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

DONE



# Send Email using Gmail

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 includes the sub-section "Settings and recommendations to help you keep your account secure".

The first section, "Security issues found", features an illustration of a smartphone, laptop, and padlock with a yellow exclamation mark, and the text "Protect your account now by resolving these issues".

The second section, "Signing in to Google", features an illustration of a smartphone, laptop, and key with a yellow exclamation mark, and the text "Last changed 10 Jan 2018". It includes three items:

- 2-Step Verification** (checkbox checked, labeled "On") - highlighted with a red box.
- 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)

At the bottom of the page, there is a taskbar with the entry "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 = "xxxxxxxxxxxxxx"
receiver_email_address = "another_email_address@gmail.com"
email_title_content = "Subject: Sending Email Using Python\nThis is a test email."
```

- Run your email program

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

Process finished with exit code 0
```



# Send Email using Gmail

---

- Send email to students who were absent

A	B	C	
1	Student	Email	Status
2	Alicia	code.musically@gmail.com	Present
3	Bryan	code.musically@gmail.com	Present
4	Carol	code.musically@gmail.com	Absent
5	David	code.mu	1 <i>#!/ python3</i>
6	Evelyn	code.mu	2
7			3 <i>import openpyxl, smtplib</i>
			4
			5 <i>def sendEmail(name, emailTo):</i>
			6 <i>    email_body = "Subject: Your attendance. \nDear %s, \nYou were absent for class.\n" %(name)</i>
			7
			8 <i>    smtpObj = smtplib.SMTP("smtp.gmail.com", 587)</i>
			9 <i>    smtpObj.starttls()</i>
			10 <i>    smtpObj.login("code.musically@gmail.com", "xxxxxxxxxxxx")</i>
			11 <i>    smtpObj.sendmail('code.musically@gmail.com', emailTo, email_body)</i>
			12
			13 <i>    smtpObj.quit()</i>



# Send Email using Gmail

---

- Send email to students who were absent

```
16     workbook = openpyxl.load_workbook("D:\\CET_Python\\students_attendance.xlsx")
17     sheet = workbook["Sheet1"]
18
19     max_row = sheet.max_row
20     max_column = sheet.max_column
21
22     for i in range(1, max_row+1):
23
24         attendance = sheet.cell(row=i, column=3).value
25
26         if attendance == "Absent":
27             name = sheet.cell(row=i, column=1).value
28             email = sheet.cell(row=i, column=2).value
29
30             print(name + " is absent.")
31             sendEmail(name, email)
32             print("Email sent to " + email)
33             print()
34
```



# Telegram Bot

The screenshot shows a conversation with the BotFather bot in the Telegram interface. The user has run the command /start at 11:35 AM. The bot has responded with its introductory message about creating and managing bots. The user then runs the command /newbot at 11:36 AM. The bot asks for a name for the new bot at 11:36 AM. The user replies with 'ezlink\_skw\_bot' at 11:37 AM. The bot confirms the creation of the bot and provides its username at 11:37 AM. It also gives the user a token for the HTTP API at 11:37 AM. The user is advised to keep the token secure and store it safely.

```
BotFather
bot
Today
/start 11:35 AM ✓
I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.
You can control me by sending these commands:
/newbot - create a new bot
/mybots - edit your bots [beta]
Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot
Bot Settings
/token - generate authorization token
/revoke - revoke bot access token
/setinline - toggle inline mode
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups
Games
/mygames - edit your games! [beta]
/newgame - create a new game
/listgames - get a list of your games
/editgame - edit a game
/deletegame - delete an existing game
11:35 AM
11:36 AM ✓
/ezlink_skw_bot 11:37 AM ✓
Good. Now let's choose a username for your bot. It must end in 'bot'. Like this, for example: TetrisBot or tetris_bot.
ezlink_skw_bot 11:37 AM
Done! Congratulations on your new bot. You will find it at t.me/ezlink\_skw\_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
Use this token to access the HTTP API:
726962401:AAHoIAxriiIzyEpBds9cJW3eJHochb0lo2i
Keep your token secure and store it safely, it can be used by anyone to control your bot.
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
11:37 AM
Write a message...
```

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



# Telegram Bot

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

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

response

intents



# Telegram Bot

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

Process the message



# Telegram Bot

The screenshot shows a Telegram chat window with a background image of a daisy flower. The bot, named "Wookiee", has sent several messages:

- A "Help" message at 10:15 AM: "Use one of the following commands:  
help: To show this help  
offers: To see this week offers  
events: To see this week events" (timestamped 10:15 AM)
- An "Offers" message at 10:15 AM: "Offers 10:15 AM ✓" (timestamped 10:15 AM)
- An "Offers" message at 10:15 AM: "Offers 10:15 AM" (timestamped 10:15 AM)
- An "Events" message at 10:16 AM: "Events 10:16 AM ✓" (timestamped 10:16 AM)
- An "Events" message at 10:16 AM: "Events 10:16 AM" (timestamped 10:16 AM)
- A message at 10:17 AM: "I don't understand you. Please type 'help' for options" (timestamped 10:17 AM)
- A "Help" message at 10:17 AM: "Help 10:17 AM ✓" (timestamped 10:17 AM)
- A message at 10:17 AM: "I don't understand you.  
Please type 'help' for options" (timestamped 10:17 AM)
- A "Start" message at 10:17 AM: "Start 10:17 AM ✓" (timestamped 10:17 AM)

At the bottom right of the screen, there is a yellow hand icon with a checkmark inside it, followed by the timestamp "10:17 AM" and a checkmark.



# Telegram – Custom Keyboard

Wookiee  
bot

Events 10:16 AM

Join us for an incredible party the Thursday in our Marina Bay Sands shop! 10:16 AM

10:17 AM ✓

I don't understand you. Please type 'help' for options 10:17 AM

Help 10:17 AM ✓

Start 10:17 AM ✓

I don't understand you.  
Please type 'help' for options 10:17 AM

Help 10:36 AM ✓

Use one of the following commands:  
help: To show this help  
offers: To see this week offers  
events: To see this week events 10:17 AM

Use one of the following commands:  
help: To show this help  
offers: To see this week offers  
events: To see this week events 10:36 AM

Write a message...

help

offers

events



# Telegram – Custom Keyboard

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



# Telegram – Custom Keyboard

---

- Set the reply\_markup parameter to the custom keyboard in your call to sendMessage()

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



# Charting



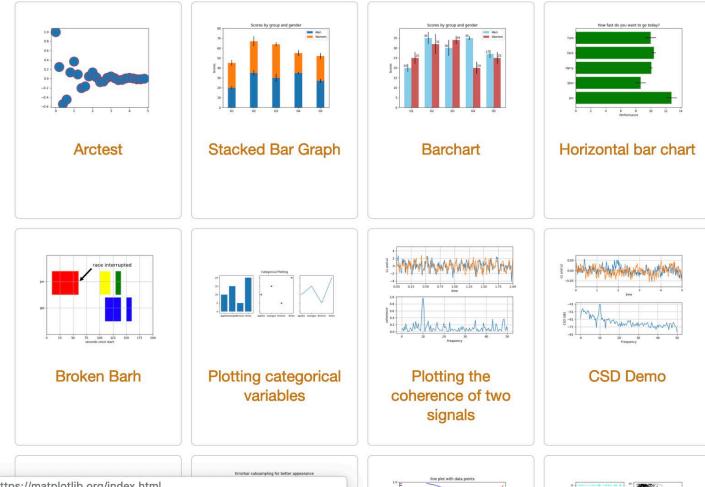
[home](#) | [examples](#) | [tutorials](#) | [API](#) | [docs](#) »

## 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`

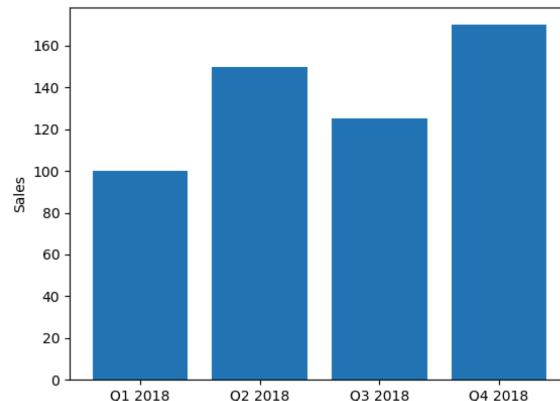
Full documentation:  
<https://matplotlib.org/>



# 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

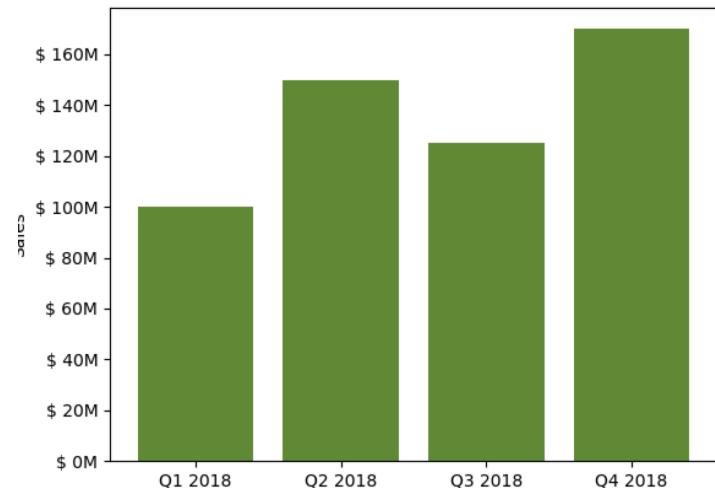
[https://matplotlib.org/api/\\_as\\_gen/matplotlib.pyplot.bar.html](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html)



# Charting

```
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



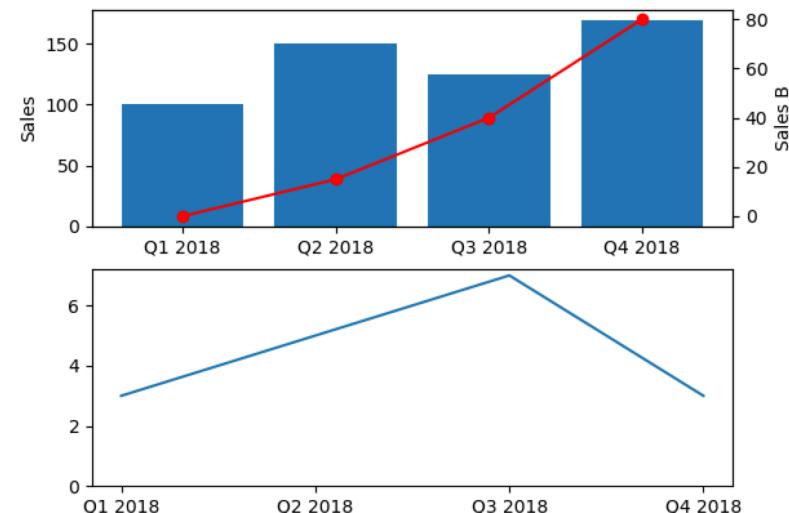


# Charting

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

- Multiple charts

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)

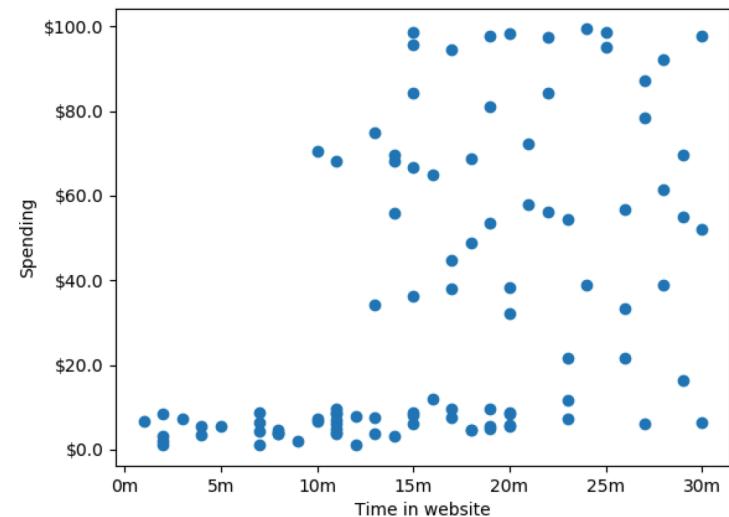


# Charting – Scatter Plot

```
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





# PDF

PyPDF

Search docs

Project Home

Home

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

Docs » Project Home » Home

Edit on GitHub

## FPDF for Python

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

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



### Main features

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

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

- Python 2.5 to 3.4+ support (see [Python3 support](#))
- [Unicode](#) (UTF-8) TrueType font subset embedding (Central European, Cyrillic, Greek, Baltic, Thai, Chinese, Japanese, Korean, Hindi and almost any other language in the world) **New!** based on [sPDF](#) LGPL3 PHP version from [Ian Back](#)
- Improved installers (setup.py, py2exe, PyPI) support
- Barcode 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



# PDF – Basic document

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

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

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

pdf\_01.py



# PDF – adding images

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

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



PDF Test Document





# PDF – Adding password

- pip install PyPDF2

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

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



# End of Day 2

---

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

Thank you !

**QUESTIONS ?**





# Where to go from here ?

---

Getting started step by step

<http://www.python.org/about/gettingstarted/>

Run through the python tutorials:

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

Keep the API doc under your pillow:

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

Advanced examples:

<http://www.diveintopython.org/toc/index.html>



# Where to go from here ?

---

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

Edx  
<https://www.edx.org/>

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

The screenshot shows the DataCamp website with a search result for "python". There are five course cards displayed:

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

Below the DataCamp screenshot is the edX homepage. It features a banner for "Cyber Monday" with the text "THE COUNTDOWN IS ON! Get 15% off your purchase." and a "Start Exploring" button. The page also includes logos for various partner universities: MIT, Harvard, Berkeley, University of Texas at Austin, The Hong Kong Polytechnic University, and The University of British Columbia.



# Where to go from here ?

---



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

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

# Lifelong Learning



Scan me



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

## Short Courses



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

[AI](#) [Data Analytics](#) [IT Security](#) [DevOps](#) [Software Development](#) [New/ Digital Media](#) [UAV](#) [RPA](#)

+ [Artificial Intelligence for Everyone - A Practical Experience \(1 day Beginner\)](#)

+ [Artificial Intelligence for Techies - A Hands-On Approach \(1 day Beginner\)](#)

+ [An Introduction to Code-Free Machine Learning \(1 day Beginner\)](#)

# **(SF) Introductory Programming using Python**

## **(12-13 Dec 2019)**



**<https://tinyurl.com/uwhaeno>**

**Applied Artificial  
Intelligence with  
Python  
(2 days)**



**Intro Code-Free  
Machine Learning  
(1 day)**



**AI for Everyone –  
A Practical  
Experience  
(1 day)**



**Mining Data for  
Insights @ Work  
(2 days)**



**Fundamentals of  
Data Visualisation  
(2 days)**



**Deep Learning  
with Python  
(2 days)**



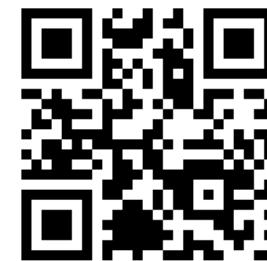
**Developing  
your first  
Chatbot  
(1 day)**



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



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



**AWS Academy  
Cloud  
Foundations  
(3 days)**



**An Introduction to  
DevOps  
(2 days)**



**Computer Vision  
with Deep  
Learning (4 days)**



.fortytwo.sg/dir

sts

```
    requests.get("http://fortytwo.sg/dir")
raise_for_status()
beautifulSoup(red
oup.select("#prod
ts[0].text)
```

Up To 60% OFF Dining Room Categories

# FORTYTWO

Search furniture, mattress, home & decor

## Landon Regular Dining Coffee



6 customer reviews



```
<div>
  <div>
    <div>
      <p>c</p>
      <p>c</p>
      <span></span>
    </div>
  </div>
</div>
```

...

```
</p>
</div>
</div>
  <div>
    <div>
      <div>
        <div>
          <div>
            <div>
              <div>
                <div>
                  <div>
                    <div>
                      <div>
                        <div>
                          <div>
                            <div>
                              <div>
                                <div>
                                  <div>
                                    <div>
                                      <div>
                                        <div>
                                          <div>
                                            <div>
                                              <div>
                                                <div>
                                                  <div>
                                                    <div>
                                                      <div>
                                                        <div>
                                                          <div>
                                                            <div>
                                                              <div>
                                                                <div>
                                                                  <div>
                                                                    <div>
                                                                      <div>
                                                                        <div>
                                                                          <div>
                                                                            <div>
                                                                              <div>
                                                                                <div>
                                                                                  <div>
                                                                                    <div>
                                                                                      <div>
                                                                                        <div>
              itemsco
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

... #product\_addtocart\_form

Styles

Event Listeners

DO

Filter

⋮

Console

What's New

Highlights from the Chrome DevTools

Multi-client support in the DevTools  
Use the Audits panel in combination with Local Overrides or Request Interception

Payment Handler debugging

Stack Data

E

location

