

Introductory Programming using Python

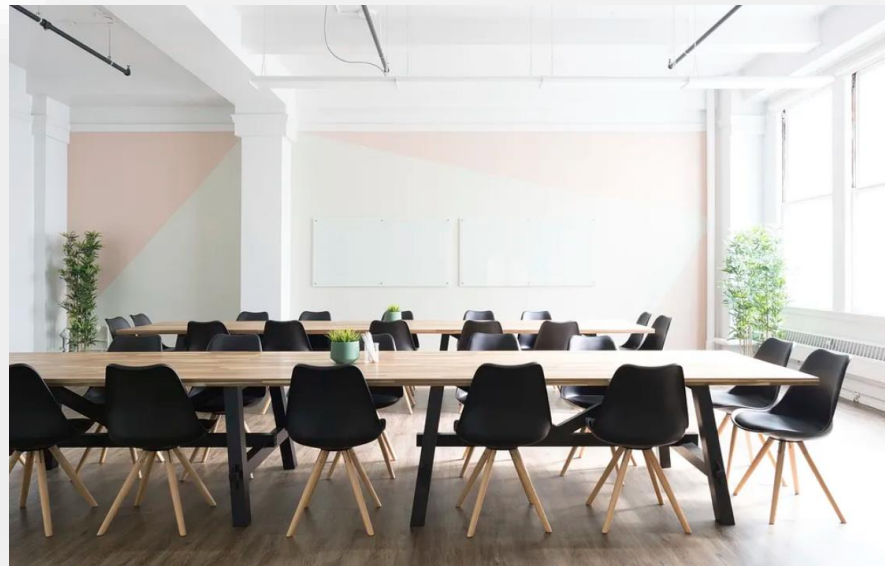
Day 2

Republic Polytechnic



Welcome and admin matters

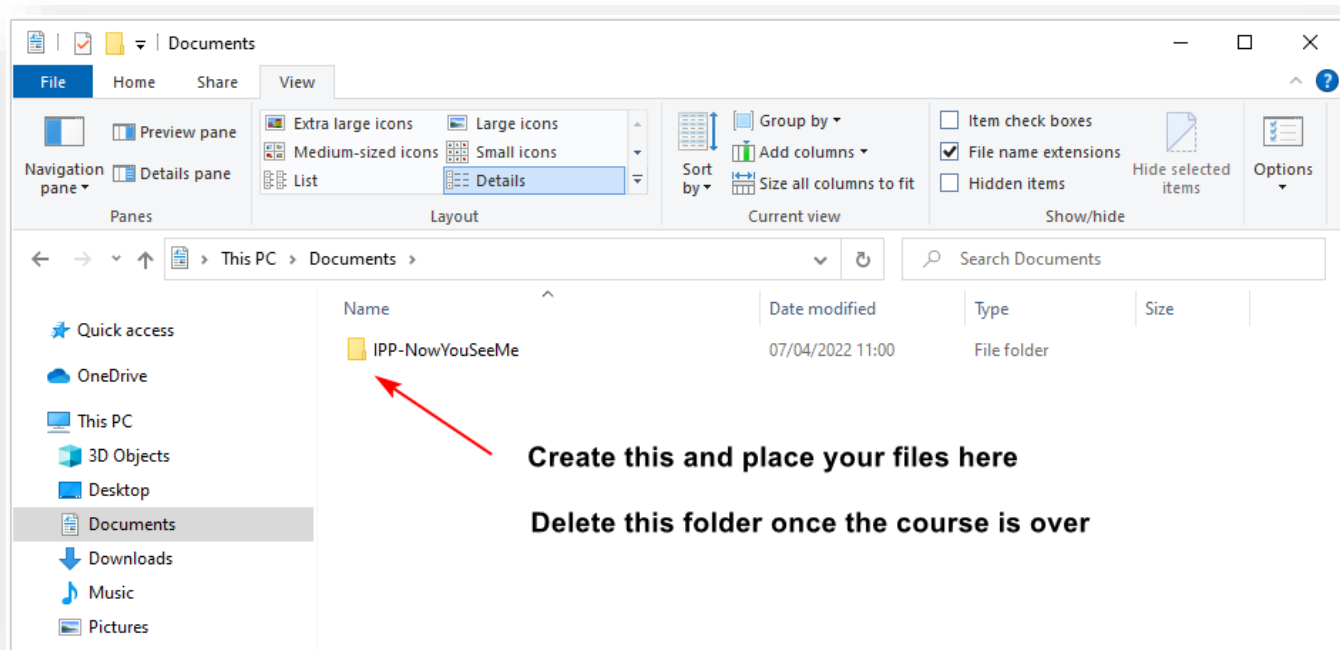
- Please ensure that:
 - your attendance has been captured
 - you have a learning laptop with you
 - you have a good view on the display





Welcome and admin matters

- A tidy/clean laptop is a good laptop for learning
- Created files on the laptop will linger on. To prevent this:
 - Create a folder in the "Documents" location for this workshop
 - E.g. "IPP-Alan" or "IPP-PeterPan"
 - All user created files to be placed in this folder
 - Delete that folder after the course





Overview: Day Two

Morning	Afternoon
<ul style="list-style-type: none">• String functions• String formatting• Dictionary• Read and writing files• Copying, moving and deleting files and folders	<ul style="list-style-type: none">• Working with Excel• Image Processing• Connecting to the Web• Demo: Sending Emails (outlook)



String functions

- Split

```
>>> a='python or java'
>>> b=a.split(' ')
>>> type(b)
<type 'list'>
>>> b
['python', 'or', 'java']
>>>
```

```
>>> a='python or java'
>>> b=a.split('on')
>>> b
['pyth', ' or java']
>>>
```

- Join

```
>>> a=['python','and','java']
>>> b=' '.join(a)
>>> b
'python and java'
>>> c=', '.join(a)
>>> c
'python,and,java'
>>>
```



String Slicing

```
>>> s = "freedom"  
>>> print(s[:4])  
free  
>>> print(s[-3:])  
dom  
>>> |
```

- Slicing works for any sequence (e.g. list), so it works for strings too.

`[:4]` gets from the start till the fourth character

`[-3:]` gets the last third till the last character



FIX_ME exercise

split or slice?

- Walkthrough on the following use case:
 - Extract the digit part of a Singapore NRIC number
 - Extract the user id from an email address
- Let's solve it together



Exercise – Find Longest Word

- Create the function **findLongestWord** that takes in a sentence and returns the longest word.

Hint: Use `split()`, repetitions



15 mins

findLongestWord.py



Strings – Useful functions

- Useful functions for validations
 - `String.isupper()` – Returns True if all characters in *String* are in uppercase.
 - `String.islower()` – Returns True if all characters in *String* are in lowercase.
 - `String.isalpha()` – Returns True if *String* contains alphabets only.
 - `String.isdigit()` – Returns True if all characters in *String* are numbers only.
 - `len(String)` – Returns the total number of characters (including spaces) in *String*
 - `String.count(text)` – Returns the number of times *text* appears in *String*.
 - `String.startswith(text)` – Returns True if *String* starts with *text*.
 - `String.endswith(text)` – Returns True if *String* ends with *text*.
- Useful functions for manipulations
 - `String.upper()` – Returns a String with all characters in the original String converted to uppercase
 - `String.lower()` – Returns a String with all characters in the original String converted to lowercase
 - `String.replace(x, y)` – Returns a String with all occurrences of *x* in the original String replaced with *y*



String formatting

• Formatting numbers

- %d int
- %f float

More about string formatting technique can be found here:

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

• Special formatting

- %.2f float two decimal places
- %+d sign printing (+)
- %+f e.g. +5.6

```
>>> import math
>>> print("Pi is " + str(math.pi))
Pi is 3.141592653589793
>>> print("Pi is approx %.2f"%(math.pi))
Pi is approx 3.14
>>> print("Pos or Neg: %+d %+d"%(-5,3))
Pos or Neg: -5 +3
>>> |
```

• Formatting Strings

- %15s reserve 15 characters for string, right-justified
- %-15s reserve 15 characters for string, left-justified

In the earlier exercise on temperature_calculator.py, the output can be rewritten as:

```
print(name + "'s temperature is " + str(diff_from_369) + " degree from 36.9 degree celsius")
```



```
print("%s's temperature is %.2f degree from 36.9 degree celsius"%(name,diff_from_369))
```



Exercise – String formatting (1)

- Try this out yourself!
 - Open *string-format1.py*
 - Change the values for the value line
 - Execute and observe the effect

```
>>> import math
>>> a = math.pi
>>> a
3.141592653589793
>>> b=5
>>> c="python"
>>> line="%s %f %d"%(c,a,b)
>>> line
'python 3.141593 5'
```

```
>>> line="%03d"%(b)
>>> line
'005'
```



5 mins





Exercise – String formatting (2)

- Given the variable

```
x = "admin:$E*G$@R:/users/root:"
```

Write a program to display the following output:

User	: admin
Password	: \$E*G\$@R
Homedir	: /users/root



10 mins



Python Dictionary

```
dictionary = {'a':1, 'p':1, 'r':2, 't':1, 'o':1}
```

- A dictionary stores multiple **key-value** pairs
- Each key-value pair are separated by a colon (:)
- Every key is unique; no duplicate key within a dictionary
- A dictionary uses a set of curly brackets { } to store its key-value pairs
 - Contrast with a Python list that uses square brackets []
- To access a value in the dictionary, we use the key as an index
 - e.g. `print(dictionary["r"]` displays value of 2



Python Dictionary

- You can *add*, *edit* and *delete* elements from a dictionary

```
# create dictionary with some elements
members = {'mary': 18, 'alan': 20, 'peter': 21}

# add element
members['john'] = 20

# edit element
members['mary'] = 25

# delete element
del members['alan']

# get number of element
print(len(members))

# display all the elements
print(members)
```

Key "john" **NOT IN** the *members* dict. This **ADDS** the key "john" and value 20 pair

Key "mary" **IS IN** the *members* dict. This **EDITS** the value of key "mary"



Python Dictionary

- We can traverse and iterate over a dictionary using *for* loop
 - e.g. assume members contain name (key) and age (value)
To calculate the average age:

```
1 total = 0
2 members = { "mary":18, "alan":20, "peter":21 }
3
4 for key in members: # Iterate the key:value pairs using a for loop
5     age = members[key] # Assign the value for each key in the variable age
6     total = total + age
7
8 average = total / len(members)
9 average = round(average, 2)
10 print("The average age is " + str(average) )
```



Exercise – Dictionary Operations

- Create a dictionary with the following key:value pairs
 - "alan" : 80001234
 - "mary": 90004567
 - "peter": 61234567
- Update the value for "mary" to 91110000
- Remove the element with "peter" as the key

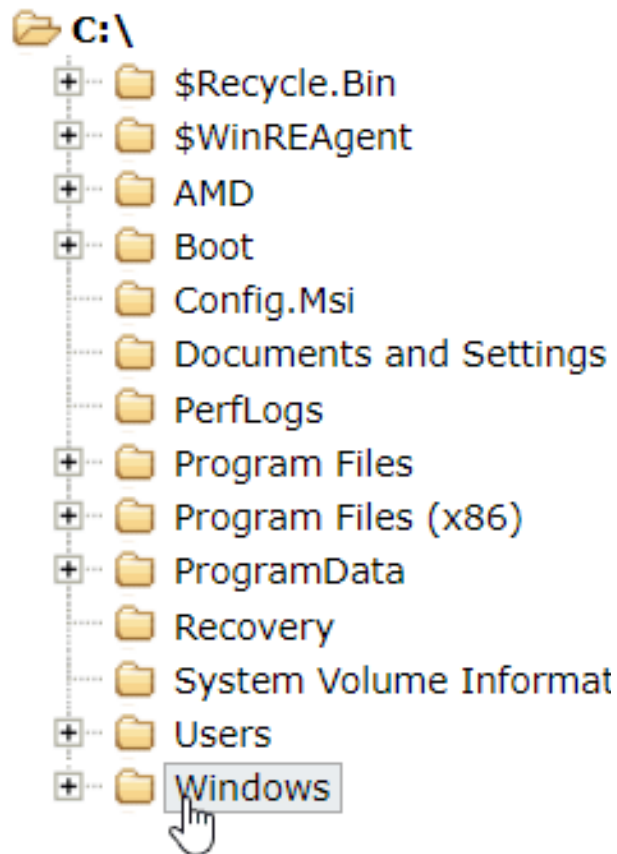


5 mins

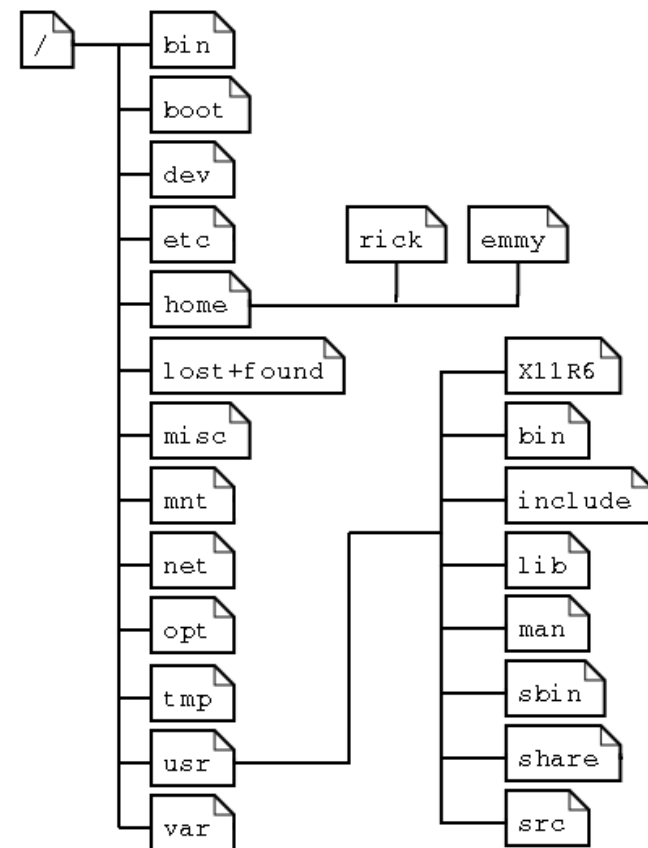


File Tree

- **Windows**



- **Linux**

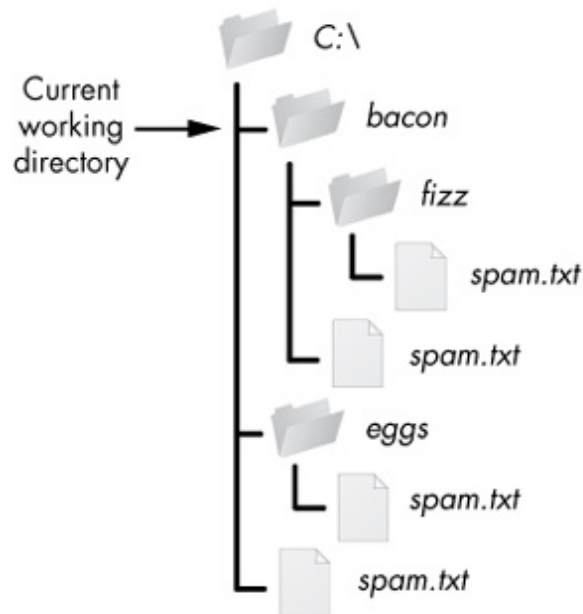




File Paths

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

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



Relative Paths	Absolute Paths
<code>..\</code>	<code>C:\</code>
<code>.\</code>	<code>C:\bacon</code>
<code>.\fizz</code>	<code>C:\bacon\fizz</code>
<code>.\fizz\spam.txt</code>	<code>C:\bacon\fizz\spam.txt</code>
<code>.\spam.txt</code>	<code>C:\bacon\spam.txt</code>
<code>..\eggs</code>	<code>C:\eggs</code>
<code>..\eggs\spam.txt</code>	<code>C:\eggs\spam.txt</code>
<code>..\spam.txt</code>	<code>C:\spam.txt</code>

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

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



Read files

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

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

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

Call read() to read the contents of a file

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

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

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. Options

```
>>> 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019)
Python Type "help", "copyright", "credits() or license()
[evaluate file_read_01.py]
THis is also another line
Hello world again

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

file_read.py



Write files

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

Search Stack Data

Search:

Replace:

☐ Case sensitive ☐ Whole words ☐ In Selection

Previ Ne: eplac place Option:

Debug I/O Python Shell

Commands execute without debug. Use arrow keys for history.

```
>>> 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC
Python Type "help", "copyright", "credits" or "license" for
[evaluate file_write.py]
This is also another line

This is also another line
Hello world again
```

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



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

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



Copying and moving files

```
import shutil

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

# copy file, destination must exist, and into a new name
shutil.copy("hello.txt", "folder1/newhello.txt")

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

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

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

`shutil.copy(src, dst)` – Copy the file *src* to the file or directory *dst*

`shutil.copytree(src, dst)` - Recursively copy an entire directory tree rooted at *src*.

`shutil.move(src, dst)` - Recursively move a file or directory (*src*) to another location (*dst*).

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



Deleting files

```
import os

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

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

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

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

- `os.unlink()` will delete a file
- `os.rmdir()` will delete a folder (but folder must be empty)
- `shutil.rmtree()` will delete a folder and all its contents

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

```
import os

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



Deleting can be dangerous, so do a dry run first



Use Case Sharing

- Organizing students' submissions into separate folder
 - Class of 25 students into 5 teams

25 folders,
one for
each
student

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



Team 1
Team 2
Team 3
Team 4
Team 5

Student
submission
sorted by teams



Other Use Cases

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



Exercise – File operations

- Write a program 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



10 mins

Summary

Lunch Break



Python Package Index

<https://pypi.org/>

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



Using pip install

- For all windows users by default

- Open command prompt
pip install <package_name>

- For Mac User

- Open terminal
pip3 install <package_name>

- For staff using company issued laptop with no Admin rights

- Open command prompt
pip install --user <package_name>

↑
Double-dash



Alternative

- If command prompt is inaccessible, we can try to install the packages programmatically
- Example below shows package installation via Python code

```
import subprocess
import sys

subprocess.check_call([sys.executable, "-m", "pip", "install", "openpyxl"])
subprocess.check_call([sys.executable, "-m", "pip", "install", "pillow"])
```

Excel Manipulation using Python



Working with Excel

- Install **openpyxl** module using “**pip install openpyxl**”

The screenshot shows the PyPI page for openpyxl 3.0.3. The header is blue with the text 'openpyxl 3.0.3' and a green 'Latest version' badge. Below the header, there's a grey bar with the text 'A Python library to read/write Excel 2010 xlsx/xlsm files'. The main content area is white and divided into two columns. The left column contains a 'Navigation' section with links for 'Project description', 'Release history', and 'Download files', and a 'Project links' section with links for 'Homepage', 'Tracker', 'Source', and 'Documentation'. The right column contains a 'Project description' section with a 'coverage: 95%' badge, an 'Introduction' section with text about the library's purpose and origin, and a 'Security' section with text about default security settings.

openpyxl 3.0.3

pip install openpyxl

Released: Jan 11, 2020

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

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Tracker
- Source
- Documentation

Project description

coverage: 95%

Introduction

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

- Full openpyxl documentation:
<https://openpyxl.readthedocs.io/en/stable/index.html>



Typical Workflow for Excel Automation

1

You are given some data in a spreadsheet

2

You want to do some or all of the following

- Analyse the data
- Manipulate the data

3

Output the processed data in another spreadsheet



Reading Excel file

1) Import openpyxl

```
import openpyxl
```

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

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

3) Get the worksheet named
"Sheet1"

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

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

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

5) Display the retrieved
values, only for a row



Reading Excel file

- Reading excel file typically uses a *for* loop to go through each row to read the data

```
import openpyxl
```

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

1) Get the number of rows
and columns

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

```
#loop through every row
```

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

2) Use For loop to go through
every row

```
    #read cell
```

```
    name = sheet.cell(row=i, column=1).value
```

```
    weight = sheet.cell(row=i, column=2).value
```

```
    height = sheet.cell(row=i, column=3).value
```

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

3) Extract the status at
Column C for height



Update Excel file

```
import openpyxl
```

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

1) Load file into memory & get the sheet

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

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

2) Adds a header at the 4th column

```
#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
```

3) Perform calculation with values taken from the excel files

```
    bmi = weight / (height * height)
```

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

4) Add calculated value to cell

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

```
#save the file
```

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

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 lists as data source
```

```
data = [  
    [225.7, 'Gone with the Wind', 'Victor Fleming'],  
    [194.4, 'Star Wars', 'George Lucas'],  
    [161.0, 'ET: The Extraterrestrial', 'Steven Spielberg']  
]
```

1) Some data in nested list

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

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

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

3) Save the spreadsheet



Use Case Sharing

- Write script
 - to split contact list for students based on intake year
 - to add country code: 65 to all the numbers
- The Excel file can be the input file, to be used to send WhatsApp messages to students individually (can use RPA too!)

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

2 students
in 2017
intake

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

Create worksheets for
students of different intake



FIX_ME exercise

Reinforcement learning

- Read Excel file, produce the following output:

First Name: James	Last Name: Cameron
First Name: Steven	Last Name: Spielberg
First Name: Michael	Last Name: Bay
First Name: Chloe	Last Name: Zhao

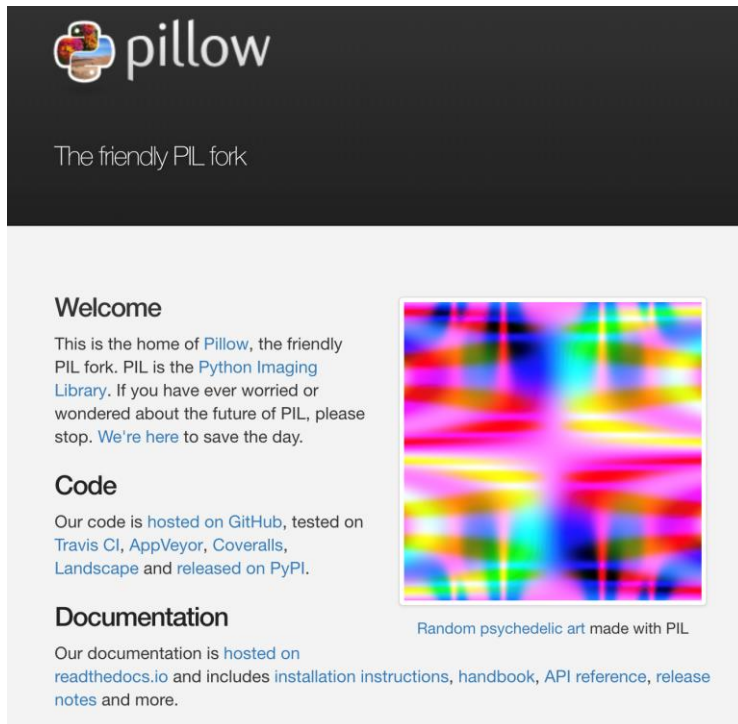
- Try to fix it, could you?

Image Processing with Python



Image Processing

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



The screenshot shows the Pillow website homepage. At the top, there is a dark header with the Pillow logo (a colorful snake head) and the text "pillow" in white. Below the header, it says "The friendly PIL fork". The main content area has a light gray background. On the left, there is a "Welcome" section with text explaining that Pillow is the friendly PIL fork and that PIL is the Python Imaging Library. It also mentions that the code is hosted on GitHub, tested on Travis CI, AppVeyor, Coveralls, and released on PyPI. Below this is a "Code" section with a link to the GitHub repository. Further down is a "Documentation" section with a link to the documentation on readthedocs.io. On the right side of the main content area, there is a square image of random psychedelic art made with PIL, with a caption below it that reads "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

- Install using the following command:
`pip install pillow`
- Documentation:
<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`

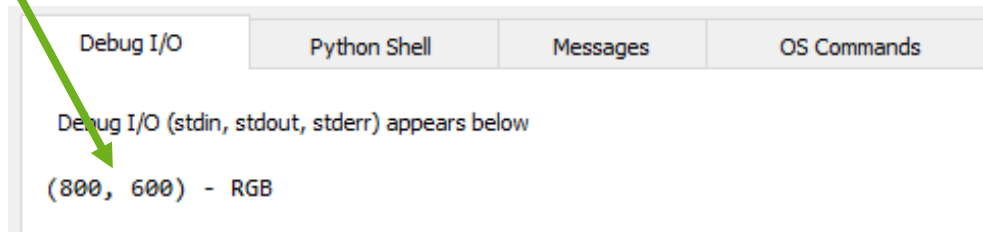




Image Processing

```
import os
from PIL import Image, ImageFilter

filename = "img/clungup.jpg"

im = Image.open(filename)

out = im.filter(ImageFilter.BLUR)

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



Pillow has many conversion and filters, to use filters we need to extend our import:

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

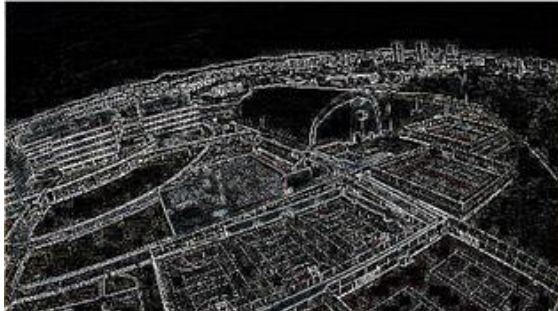
The way you can apply filters is :

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

Try other different filters!



Image Processing - filters



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

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



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



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



* Remember to include
ImageOps in your import statement



Image Processing - filters

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

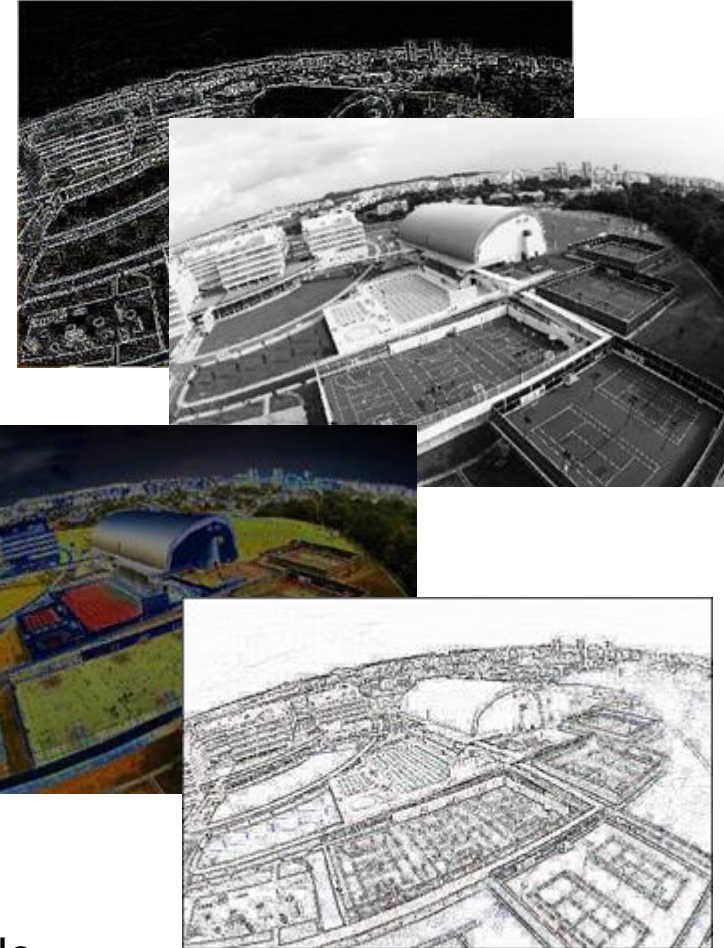
filename = "img/clungup.jpg"

im = Image.open(filename)

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

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

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



* Remember to include
ImageOps in your import statement



Image Processing – Rotating

Flipping the image horizontally or vertically

```
out = im.transpose(Image.FLIP_LEFT_RIGHT)  
out = im.transpose(Image.FLIP_TOP_BOTTOM)
```

} Flip images

Rotating the image

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

} Rotate images

Contrast

Note: Need to add ImageEnhance to our imports:

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

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

← make image brighter by
changing the contrast



Image Processing - Writing

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

filename = "clungup.jpg"

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

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

outFilename = out_folder + filename # out/clungup.jpg

out.save(outFilename)
```

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

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



Image Processing - Converting

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

filename = "clungup.jpg"

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

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

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

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

outFilename = out_folder + fname2 # out/clungup.gif

out.save(outFilename)
```

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



Image Processing – Watermark

Create the mark image
You can reduce the size to 100,100

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

Create a new function called

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

It takes the original image, the watermark image and the desired position that we want the watermark to appear. The function will return the result.

We can use this function like:

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

or

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

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

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





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:





FIX_ME exercise

Flip image

- Identify the “odd” image in the img
- Flip it to make it upright
- Try to fix it, could you?



FIX_ME exercise

Grayscale image

- Pick an image in the “img” folder
- Turn it into grayscale
- Try to fix it, could you?



Use Case I: Batch Resize

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

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

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

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



Use Case II: Batch Rename

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

```
import os
import shutil

files = os.listdir('img')

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



Exercise on Pillow

- **Write Python code to:**
 - Resize all jpg images in “img” folder to half its original size
 - Place a watermark at the lower right of the image
 - Save/store the modified images into “resized” folder



Web Automation with Python



Connecting to the Web

- requests – download files and web pages from the Web

pip install requests

```
import requests
```

Get the required information from the given URL

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





Connecting to the Web

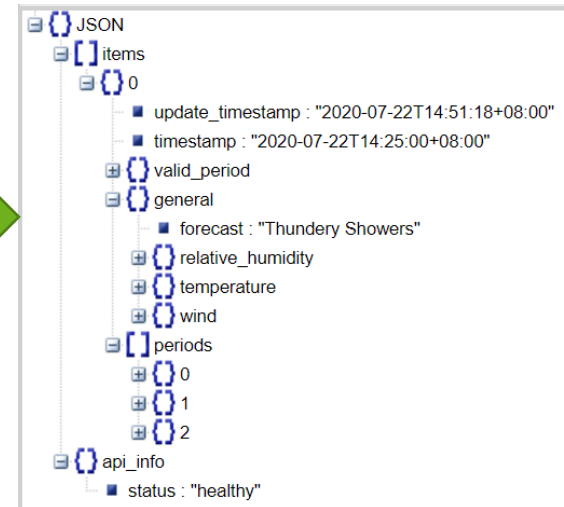
- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

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

```
import requests
```

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

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





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
```



FIX_ME exercise

Get weather info

- Get the weather information for “Ang Mo Kio”
- Produce the output as below

```
name : Ang Mo Kio, forecast : Cloudy
```

- Try to fix it, could you?



Exercise

- **Car Park Availability Data:**

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

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

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

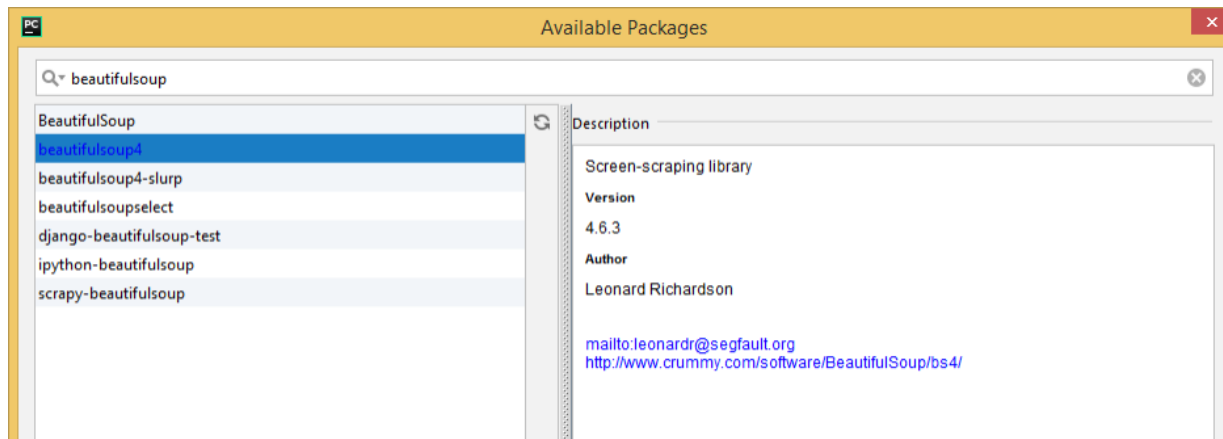


Connecting to the Web

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

Web Scraping – download and process Web content

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





Connecting to the Web

- What's the URL?

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

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

FORTYTWO

Search furniture, mattress, home & decor...



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

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



Landon Regular Dining Table
Coffee

★★★★★ 6 customer reviews

~~S\$129.90~~
S\$69.90

Warranty: 1 Year

Qty: 1

Add to Cart



Add to Wishlist



Email to a Friend



100 Day Free
Returns



Standard
Delivery

42EXPR



Free Assembly
Included

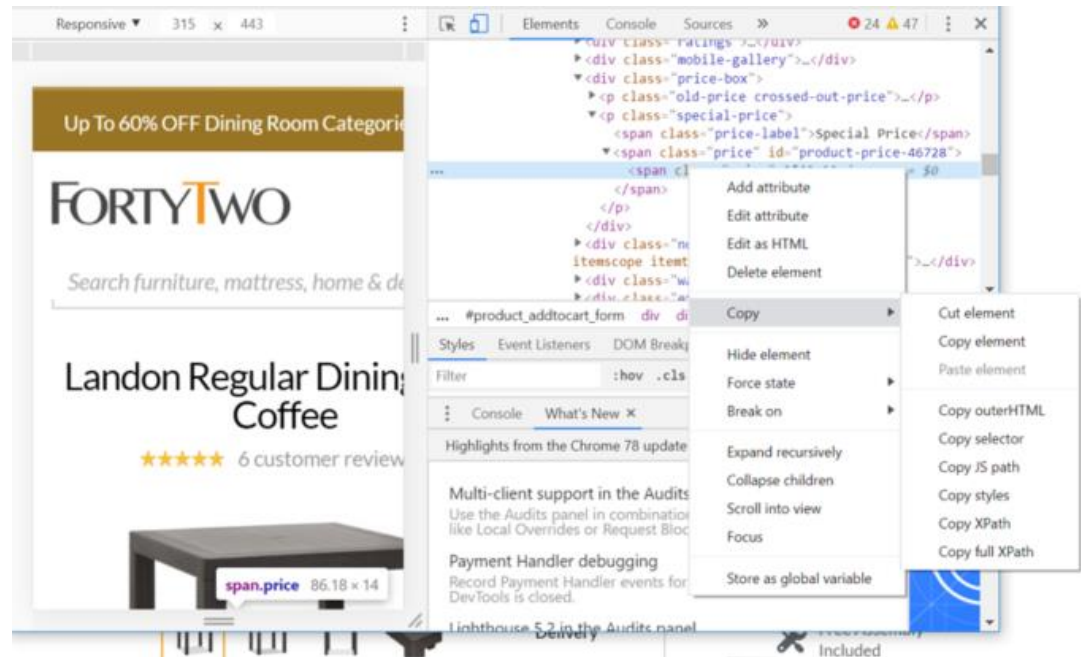


Connecting to the Web

- Get the url

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

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





Connecting to the Web

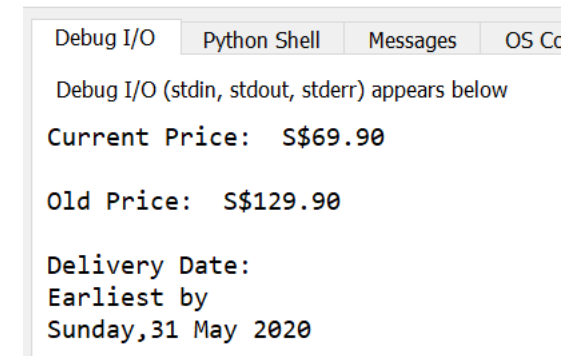
```
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup
```

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

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

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

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

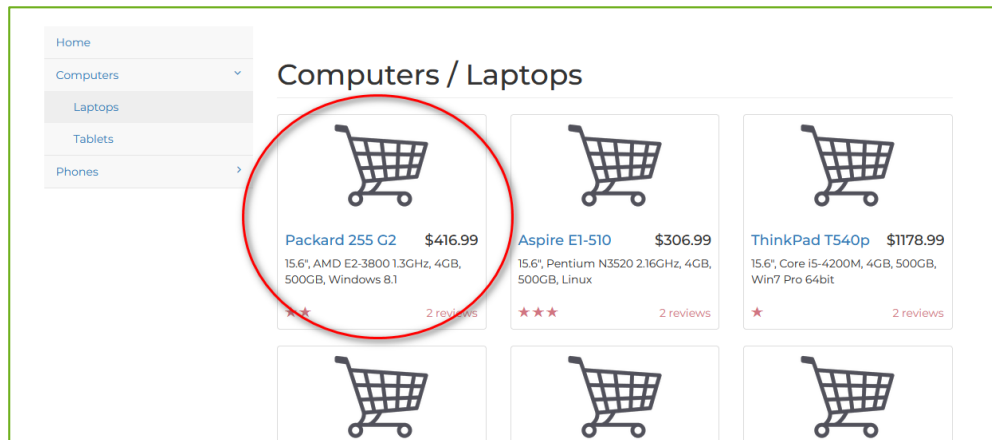




FIX_ME exercise

Getting item information

- Extract the item name and price, and produce the output below:



Item: Packard 255 G2
Price: \$416.99

- Try to fix it, could you?



Exercise

- **Table Tennis Bat Price:**

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

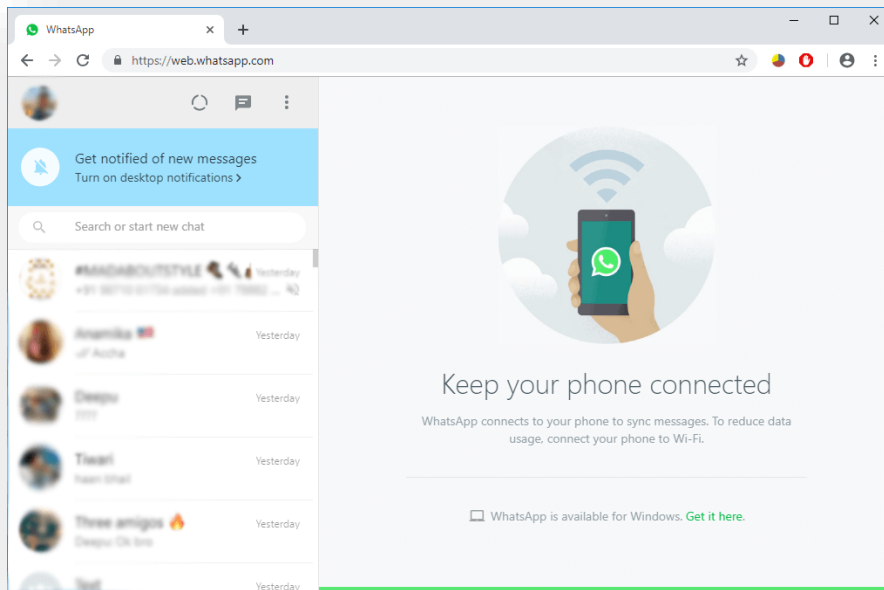


Current Price:
€133.25



Sharing other Use Cases

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



selenium python automation

Name

Your answer

Gender

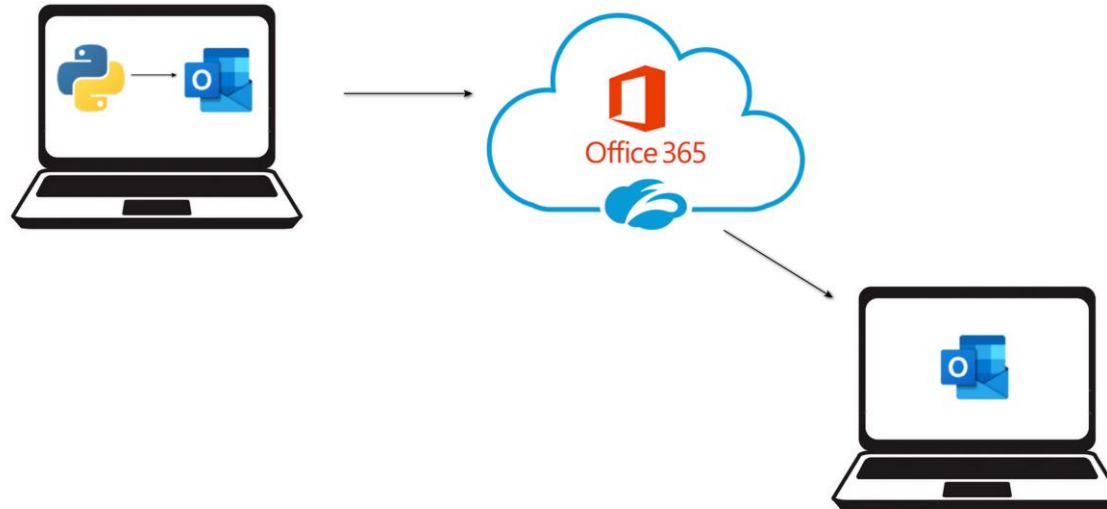
☐ Male

☐ Female

Email Automation with Python



pywin32 package



- This assumes that we are using MS Outlook on the machine the python code is run
- pywin32 package is used to allow Python to access MS Outlook
- Python is not accessing the Outlook server directly, but it is using existing account set up in the MS Outlook



Send Email using Outlook

```
import win32com.client

subject = 'email subject'
body = '<html><body>' + 'This is a test email. Oct Python 2021<br />' + '</body></html>'
recipient = 'jason_lim@rp.edu.sg'

#Create and send email
obj = win32com.client.Dispatch("Outlook.Application")
newMail = obj.CreateItem(0)
newMail.Subject = subject
newMail.HTMLBody = body
newMail.To = recipient

newMail.Send()
```

- The code above sends a simple HTML (meaning we can have tables, text formatting in it)
- No credentials are needed for the Python program as Python uses MS Outlook



Create appt using Outlook

```
import win32com.client

outlook = win32com.client.Dispatch("Outlook.Application")
appt = outlook.CreateItem(1)
appt.Start = "2021-10-08 15:30"
appt.Subject = "Meeting 1"
appt.Duration = 60
appt.Location = "Library"
appt.MeetingStatus = 0
appt.display()
appt.Save()
```

- The code above creates a calendar appointment
- No credentials are needed for the Python program as Python uses MS Outlook



Sharing Use Cases

- Sending Emails using Outlook to students
- Create Appointments using Outlook



Other Python Libraries

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





Where to go from here ?

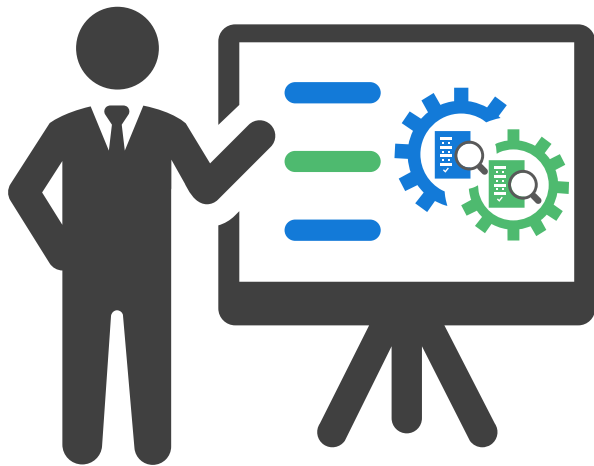


Think Python is an introduction to Python programming for beginners. It starts with basic concepts of programming, and is carefully designed to define all terms when they are first used and to develop each new concept in a logical progression. Larger pieces, like recursion and object-oriented programming 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](http://creativecommons.org/licenses/by-nc/3.0/), 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>



Thank you



Email:

wong_hau_shian@rp.edu.sg
jason_lim@rp.edu.sg