



Introductory Programming using Python

Day 2

Overview: Day Two

Morning	Afternoon
<ul style="list-style-type: none">• String functions• String formatting• Dictionary• Working with Excel	<ul style="list-style-type: none">• File and Folder operations• Connecting to the Web• Demo: Sending Emails (outlook)

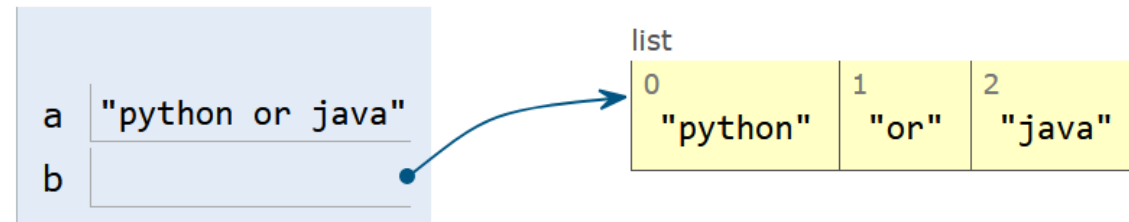
Outline for the day

Time	Agenda
9:00 am - 9:15 am	Welcome and admin matters
9:15 am – 10:30 am	
10:30 am – 10:45 am	Break
10:45 am – 12:30 pm	
12:30 pm – 1:30 pm	Lunch
1:30 pm – 3:15 pm	
3:15 pm – 3:30 pm	Break
3:30 pm – 4:45 pm	
4:45 pm – 5:00 pm	Wrap up, Q&A

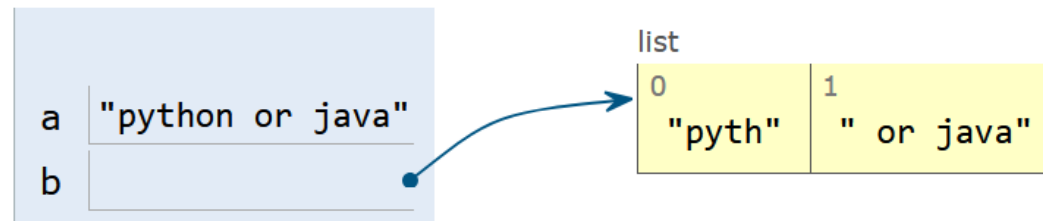
String functions

- Split

```
1 a = "python or java"
2 b = a.split(" ")
3
```

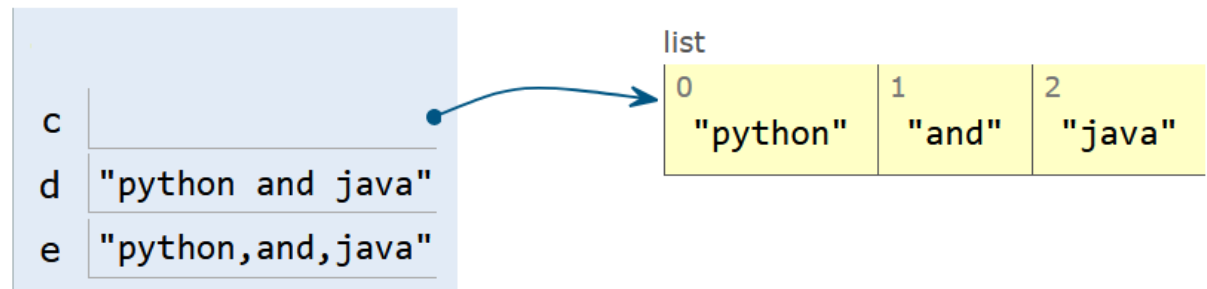


```
1 a = "python or java"
2 b = a.split("on")
3
```



- Join

```
1 c = ["python", "and", "java"]
2
3 d = " ".join(c)
4 e = ",".join(c)
5
```



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

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

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

- String format() method
 - Use {} as a place holder for the text to be printed.
 - Call the string dot format method
 - Pass the desired value into the method

More about string formatting technique can be found here:
<https://docs.python.org/3/library/string.html>

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")
```



#Using String formatting:

```
print("{}'s temperature is {:.2f} degree from 36.9 degree celsius".format(name, diff_from_369))
```

- Formatting types
 - :d Decimal
 - :f Float
 - :+ plus sign if results is positive
 - :< Left aligns the result

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
```

```
b = 5
```

```
c = 'python'
```

```
line = "{} {:.2f} {}".format(c, a, b)  
print(line)
```

```
line = "{:0>5}".format(b)  
print(line)
```



```
python 3.14 5  
00005
```



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

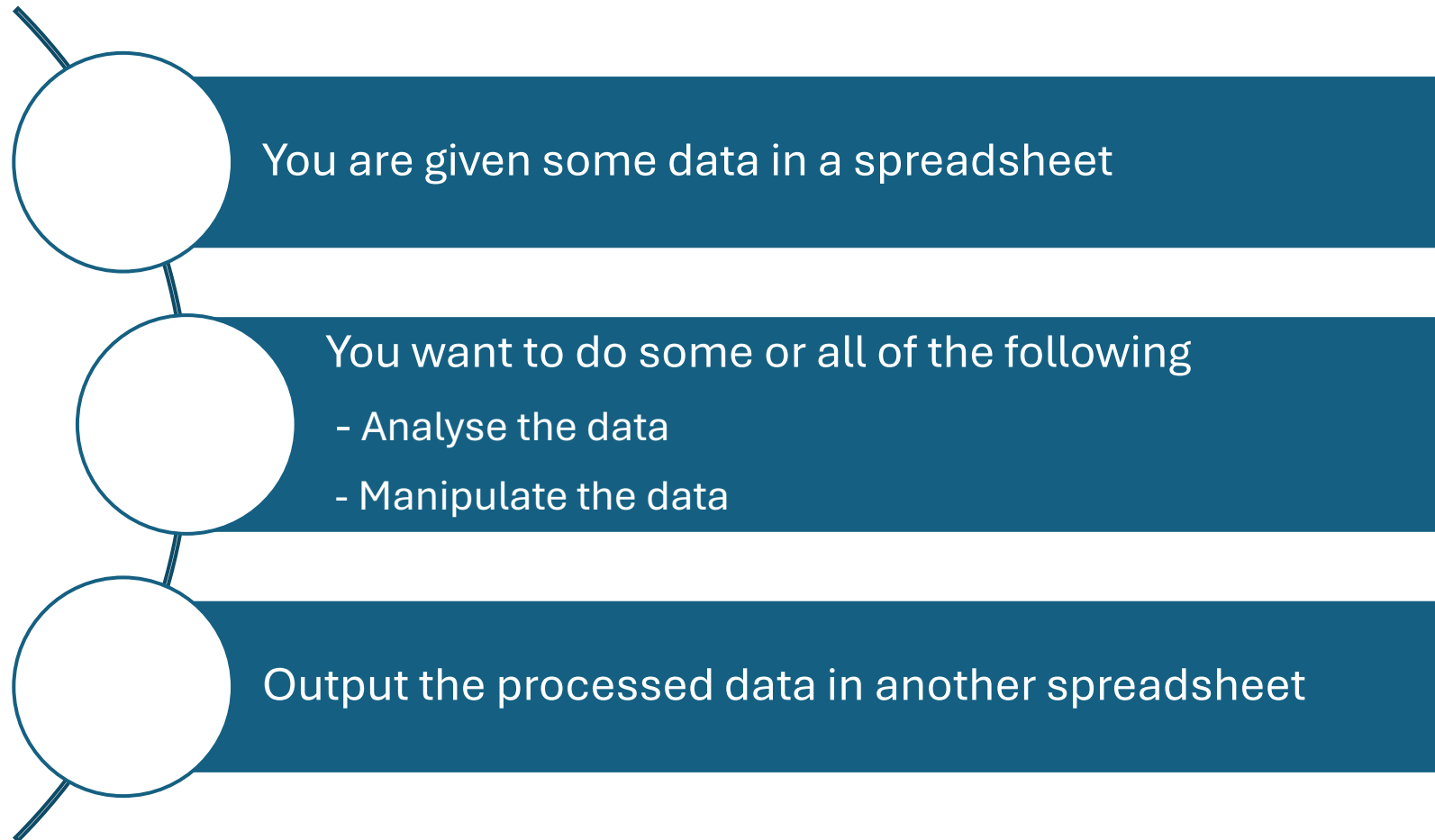


5 mins



Excel Automation using Python

Typical Workflow for Excel Automation



Excel Use Case (Walkthrough)

- Finance department receives monthly sales data from the various product lines.
- Finance department would like to:
 - **Task 1:** Merge the product specific monthly sales files into quarterly files to observe the trends



Excel Use Case (Walkthrough)

- Finance department would like to:
 - **Task 2:** Simple data cleansing to remove duplicates or empty lines

Date	Customer	Order #	Revenue	Sales Rep Name	Commission Rate
1/1/2019	Alpha Soft	i-100001	\$202.33	Jimena Carrillo	10%
2/1/2019	Soul Solution	i-100002	\$592.22	Emma Ford	10%
3/1/2019	Ironavigator	i-100003	\$59.44	Lily Grant	4%
4/1/2019	Ironavigator	i-100004	\$283.33	Jamie Webb	10%
5/1/2019	Soul Solution	i-100005	\$19.00	Jamie Webb	0%
6/1/2019	Alpire	i-100006	\$43.03	Emma Ford	0%
7/1/2019	Oystertain	i-100007	\$23.34	Jimena Carrillo	0%
8/1/2019	Titanium Solutions	i-100008	\$230.30	Lily Grant	10%
9/1/2019	Surprise Me	i-100009	\$430.33	Lily Grant	10%
10/1/2019	Apexshow	i-100010	\$492.32	Emma Ford	10%
10/1/2019	Apexshow	i-100010	\$492.32	Emma Ford	10%
10/1/2019	Ironavigator	i-100011	\$23.34	Jimena Carrillo	0%

Excel Use Case (Walkthrough)

- Finance department would like to:
 - **Task 3:** Perform mathematical calculations and create new columns to store the calculated information.

Date	Customer Name	Order #	Revenue	Sales Rep Name	Commission Rate	Total Commission
1/1/2019	Alpha Software	i-100001	\$202.33	Jimena Carrillo	10%	\$20.23
2/1/2019	Soul Solutions	i-100002	\$592.22	Emma Ford	10%	\$59.22
3/1/2019	Ironavigation	i-100003	\$59.44	Lily Grant	4%	\$2.38
4/1/2019	Ironavigation	i-100004	\$283.33	Jamie Webb	10%	\$28.33
5/1/2019	Soul Solutions	i-100005	\$19.00	Jamie Webb	0%	\$0.00
6/1/2019	Alpire	i-100006	\$43.03	Emma Ford	0%	\$0.00
7/1/2019	Oystertainment	i-100007	\$23.34	Jimena Carrillo	0%	\$0.00

Excel Use Case (Walkthrough)

- Finance department would like to:
 - **Task 4:** Glean some simple insights.
 - Total sales of product
 - Highest performing Sales rep
 - Customers with highest purchase value


Excel: Task 1

- Merge the monthly files into quarterly files to observe trends
- New concepts:
 - Libraries:
 - glob and pandas (<https://pypi.org/project/pandas/>)
 - Specifying the location of the data files:

Python Package Index

- Many of the tasks to be performed using a programming language are non unique.
- To make programming more efficient, there exists many pre-written, reusable chunks of codes that allow a programmer to quickly get its code up and working.
- These pre-written, reusable chunks of codes are known as modules (e.g. single functionality) or library (e.g., collection of functionalities)
- <https://pypi.org/> : is a repository of such codes for the Python Programming Language
E.g. send2trash, pillow, pandas, numpy etc
- Installation is easy done with the following command
pip install <software_package>
- Installed packages can be found at:
C:\python313\Lib\site-packages

Install Python Packages

- 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

Install Python Packages - Alternative

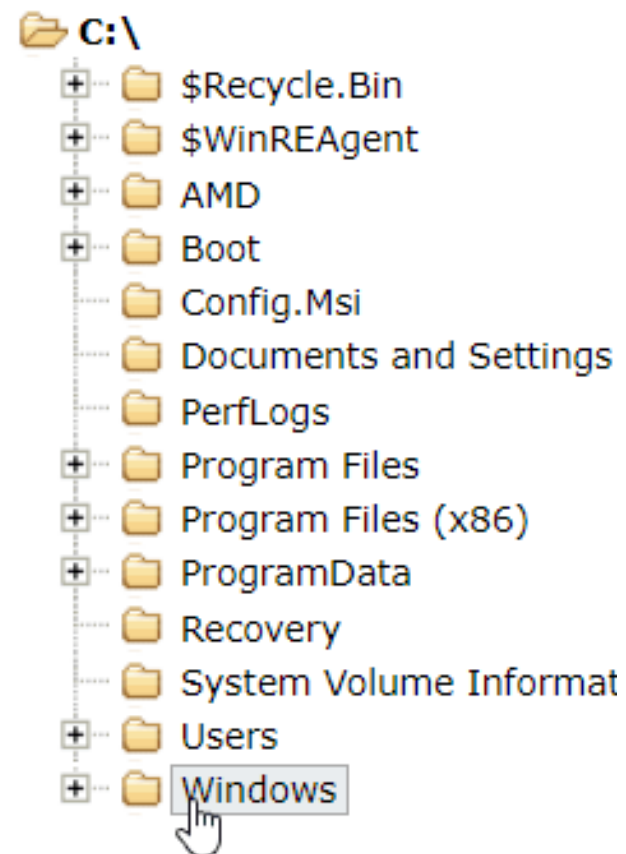
- If command prompt is inaccessible, we can try to install the packages programmatically
- Example below shows package installation via Python code

```
import subprocess
import sys

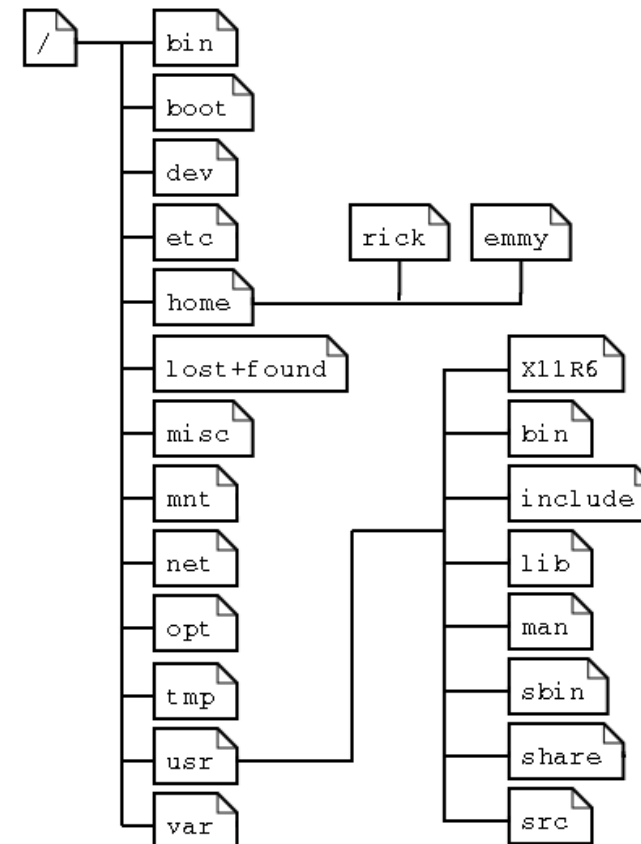
subprocess.check_call([sys.executable, "-m", "pip", "install", "openpyxl"])
subprocess.check_call([sys.executable, "-m", "pip", "install", "pillow"])
```

File Tree

- **Windows**

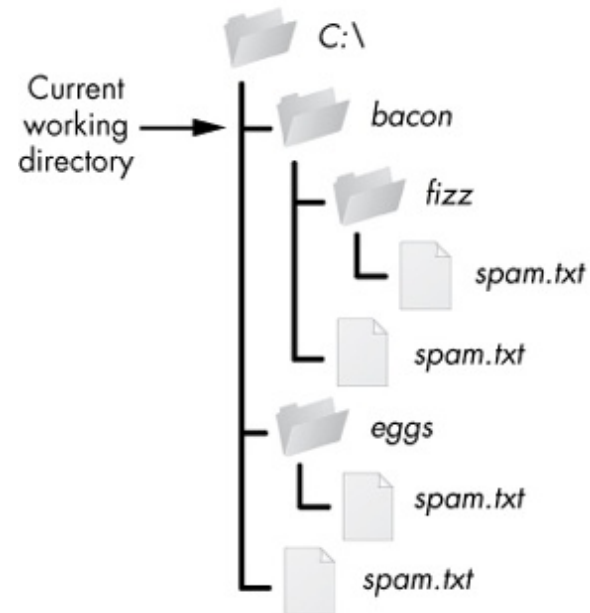


- **Linux**



File Paths

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



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

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

Relative Paths	Absolute Paths
<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>

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

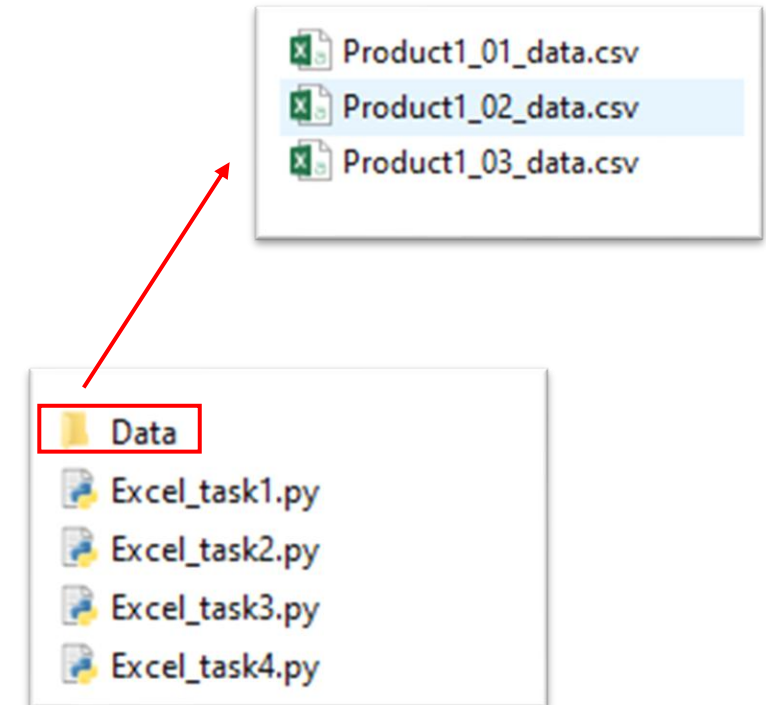
Excel: Task 1 (Code)

```
#1. Import the necessary libraries
import glob #search files that matches required patterns
import pandas as pd #analyze data

# 2. Define path to the csv data files
path = './Data/'

# 3. Create a list using glob to store file names starting with
#     Product1 and ending with .csv
fileList = glob.glob(path + "Product1*.csv")
print (fileList)

# 4. Create empty list to store the csv file as pandas dataframes
dfList = []
```



Excel: Task 1 (Code)

```
# 5. Cycle through the files
for file in fileList:

    #read the file and convert to pandas dataframe
    data = pd.read_csv(file, encoding='utf-8')

    #append to the list of dataframes
    dfList.append(data)

# merge the data in the csvList into a single pandas dataframe
csvMerged = pd.concat(dfList)

# 6. Output the pandas dataframes as csv
csvMerged.to_csv(path+ './Output/Product1_Q1_data.csv', index=False)
```

Excel: Task 2

- Clean data to remove duplicates or empty lines
- New concepts:
 - Reading the documentations to identify useful functions
- Useful Pandas functions:
 - **dropna()** to remove rows that are empty
 - <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html#pandas.DataFrame.dropna>
 - **drop_duplicates()** to remove rows that are exactly the same
 - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html

Excel: Task 2 (Code)

```
# 5. Cycle through the files
for file in fileList:

    #read the file and convert to pandas dataframe
    data = pd.read_csv(file, encoding='utf-8')

    #Task 2: remove empty lines
    data.dropna(axis = 0, how='all', inplace=True)

    #Task 2: remove duplicates entries based on names
    data.drop_duplicates(inplace=True)
    |
    #append to the list of dataframes
    dfList.append(data)

# merge the data in the csvList into a single pandas dataframe
csvMerged = pd.concat(dfList)
```

Excel: Task 3

- Perform mathematical calculations and create new columns to store the calculated information.
- New concepts:
 - Specifying the required columns
 - Stripping the data of special characters
 - Converting the data type of the columns
 - Performing simple mathematical functions on column values
 - Creating a new column to store the values of the calculations

Excel: Task 3 (Code)

```
#Task 3: Calculate Total Commission
data['Revenue'] = data['Revenue'].str.replace('$', '', regex=True)
data['Revenue'] = data['Revenue'].astype(float)

data['Commission Rate'] = data['Commission Rate'].str.replace('%', '', regex=True)
data['Commission Rate'] = data['Commission Rate'].astype(float)
|
#Create a new column and perform the mathematical calculations
data['Total Commission'] = data['Revenue']*(data['Commission Rate']/100)
```

Excel: Task 4

- **Task 4: Glean some simple insights.**
 - Total sales of a product
 - Single largest customer order
 - Highest performing Sales rep of a product
- New concepts:
 - Extract values based on conditions
 - Group data by conditions

Excel: Task 4 (Code)

```
#-----  
#Task 4: Generating insights  
#-----  
  
# Total sales of a product in the quarter  
totalSales=sum(csvMerged['Revenue'])  
print ("Total sales of product 1 in this quarter is ${:.2f} \n".format(totalSales))  
  
# What is the single largest customer order?  
largestOrder = csvMerged['Revenue'].max()  
customer = csvMerged[csvMerged['Revenue']==largestOrder]['Customer Name'].to_string(index=False)  
salesRep = csvMerged[csvMerged['Revenue']==largestOrder]['Sales Rep Name'].to_string(index=False)  
print("Customer with singles largest order is {} by {} served by {}\n".format(largestOrder, customer, salesRep))  
  
#Which sales rep brought in the highest revenue  
salesByRep = csvMerged.groupby('Sales Rep Name')['Revenue'].sum()  
salesByRepByProportion = salesByRep/totalSales  
print(salesByRepByProportion)
```



Summary



Lunch Break



Files and folders

File Reading

```
1  # make sure you have a hello.txt in your current working directory
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 content2 = helloFile.readlines()
12 print(content2)
13
14 #prints the first line
15 print("first line:", content2[0])
16
17 helloFile.close()
```

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

File Writing

```
1  #this creates a file named hello2.txt
2  helloFile = open("hello2.txt", "w")
3  helloFile.write("I am teaching in RP today\nthis is 2nd line\n")
4  helloFile.write("how are you\n")
5  helloFile.close()
6
7
8  #change line 1 to the following and observe the outcome
9  #helloFile = open("hello2.txt", "a")
10 |
```

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



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

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

Copy and moving files

```
1  import shutil
2
3  # copy file, must create folder2 first
4  shutil.copy("folder1/hello.txt", "folder2")
5
6  # recursively copy an entire directory
7  # error if the destination folder already exist
8  |shutil.copytree("folder2", "folder2_backup")
9
10 # move file, destination folder must be an existing folder
11 shutil.move("hello2.txt", "folder3")
12
13 # move and/or rename file
14 shutil.move("folder2/hello.txt", "folder2/newhello.txt")
15
```

- `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

```
1 import os
2
3 # error if file do not exist
4 - if os.path.exists("hello2.txt"):
5     os.unlink("hello2.txt")
6
7 # get current working directory
8 print(os.getcwd())
9
10 # delete directory (can only delete empty folder)
11 os.rmdir("folder2_backup")
12
13 import shutil
14 # delete directory (with content)
15 # error if folder is not found
16 shutil.rmtree("folder2_backup")
17
```

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

```
import os

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

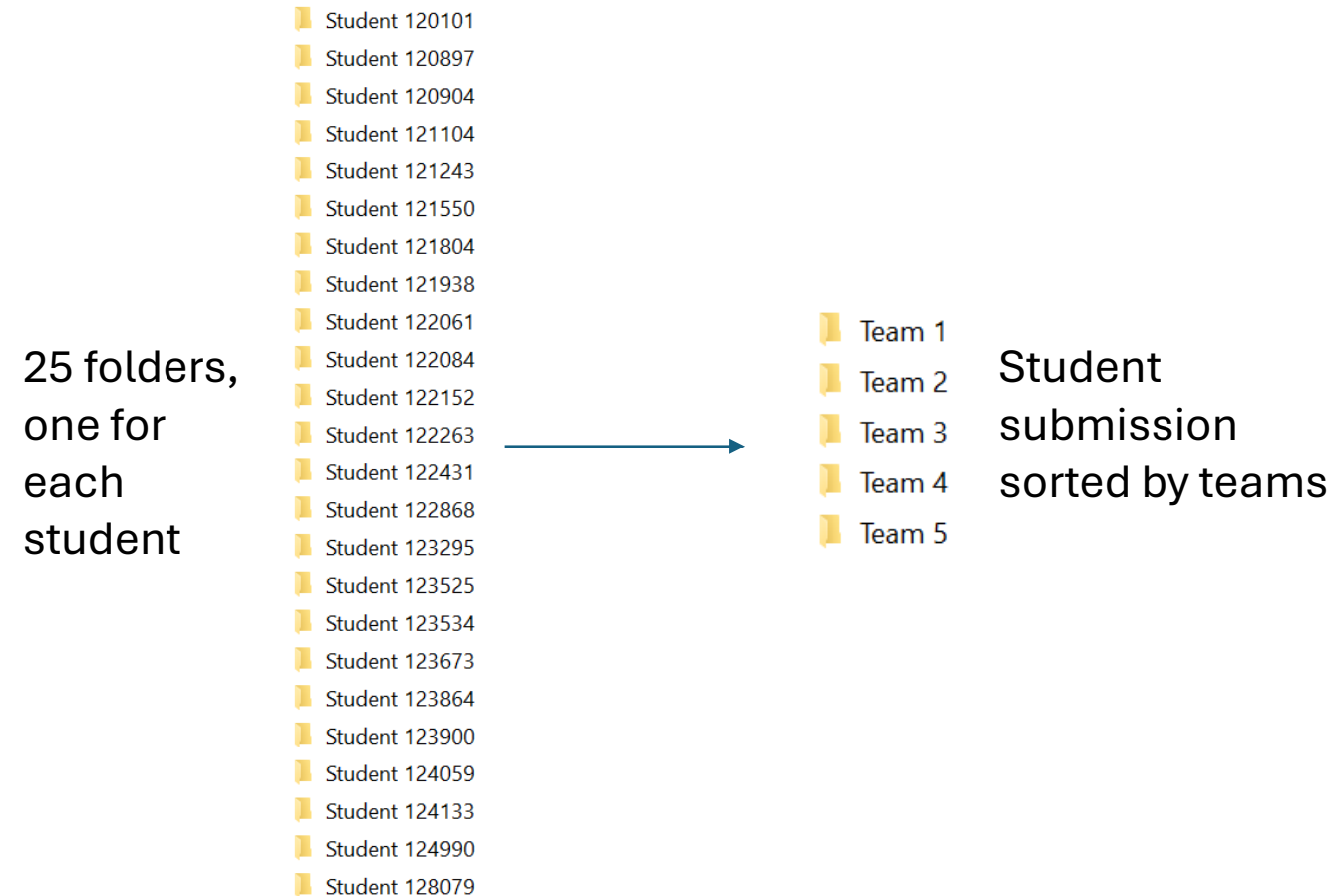
- `os.unlink()` will delete a file
- `os.rmdir()` will delete a folder (but folder must be empty)
- `shutil.rmtree()` will delete a folder and all its contents



Deleting can be dangerous, so do a dry run first

Use case sharing

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



Other use case

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

Demo: Coding with ChatGPT

Prompt:

You are tasked with creating a backup system for a directory.
Your program should do the following:

1. Create a backup directory if it doesn't already exist.
2. Copy all files from a source directory to the backup directory.
3. If any subdirectories exist in the source directory, copy them recursively to the backup directory.
4. After the backup is complete, move the source directory to a different location (e.g., an archive directory)

Demo: Coding with ChatGPT

```
import os
import shutil

usage new *
def backup_system(source_dir, backup_dir, archive_dir):
    # Create backup directory if it doesn't exist
    if not os.path.exists(backup_dir):
        os.mkdir(backup_dir)
        print(f"Backup directory '{backup_dir}' created.")

    # Copy files and directories from source to backup
    for item in os.listdir(source_dir):
        source_item = os.path.join(source_dir, item)
        backup_item = os.path.join(backup_dir, item)
        if os.path.isfile(source_item):
            shutil.copy(source_item, backup_item)
            print(f"File '{item}' copied to backup.")
        elif os.path.isdir(source_item):
            shutil.copytree(source_item, backup_item)
            print(f"Directory '{item}' copied to backup.")

    # Move source directory to archive directory
    shutil.move(source_dir, archive_dir)
    print(f"Source directory '{source_dir}' moved to archive.")
```

File/Folder related command

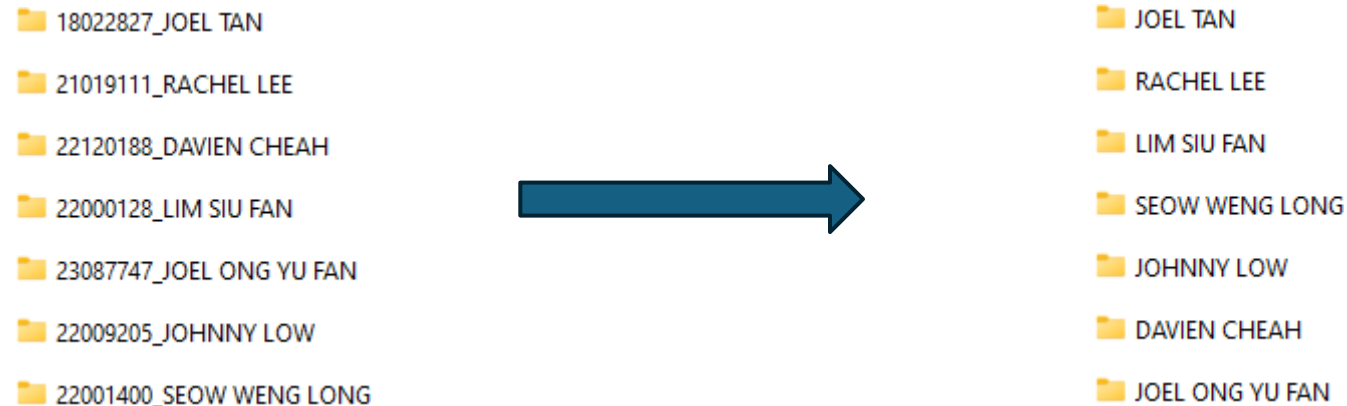
1. os.path.exists
2. os.mkdir
3. os.listdir
4. os.path.isfile
5. os.path.isdir
6. shutil.copy
7. shutil.copytree
8. shutil.move

```
# Prompt user for directory paths
source_dir = "src"
backup_dir = "dest"
archive_dir = "arch"

# Backup the system
try:
    backup_system(source_dir, backup_dir, archive_dir)
    print("Backup completed successfully!")
except Exception as e:
    print("An error occurred:", e)
```

Exercise: Coding with ChatGPT

- Use ChatGPT to generate the code to rename all the folders based on the requirement shown below



Exercise: Coding with ChatGPT

```
import os

def rename_folder(path):
    # List all items (files and folders) in the directory
    items = os.listdir(path)

    for item in items:
        # Construct full path of the item
        item_path = os.path.join(path, item)

        # Check if it's a directory
        if os.path.isdir(item_path):
            # Extract the new folder name by removing 9 characters
            new_name = item[9:]

            # Construct full path for the new folder name
            new_path = os.path.join(path, new_name)

            # Rename the folder
            os.rename(item_path, new_path)
            print(f"Renamed '{item}' to '{new_name}'")

# Example usage
folder_path = "."
rename_folder(folder_path)
```

File/Folder related command

1. `os.listdir`
2. `os.path.join`
3. `os.path.isdir`
4. `os.path.join`
5. `os.rename`

Exercise

Write code to achieve the following:

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



Web Automation with Python

Connecting to the Web

- requests – download files and web pages from the Web

```
pip install requests
```

```
import requests
```

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

Get the required information from the given URL



Connecting to the Web

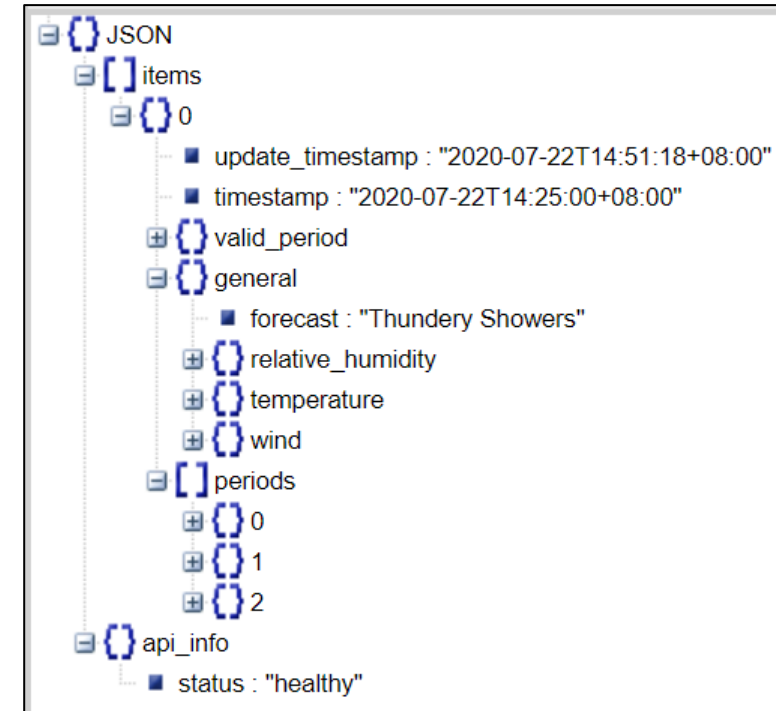
- Data is in JSON format
- Use a JSON formatter tool to present the data in a nicer form

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

```
import requests
```

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

```
{
  "items": [
    {
      "update_timestamp": "2020-07-22T14:51:18+08:00",
      "timestamp": "2020-07-22T14:25:00+08:00",
      "valid_period": {
        "start": "2020-07-22T12:00:00+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
```

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: [
            - {
```

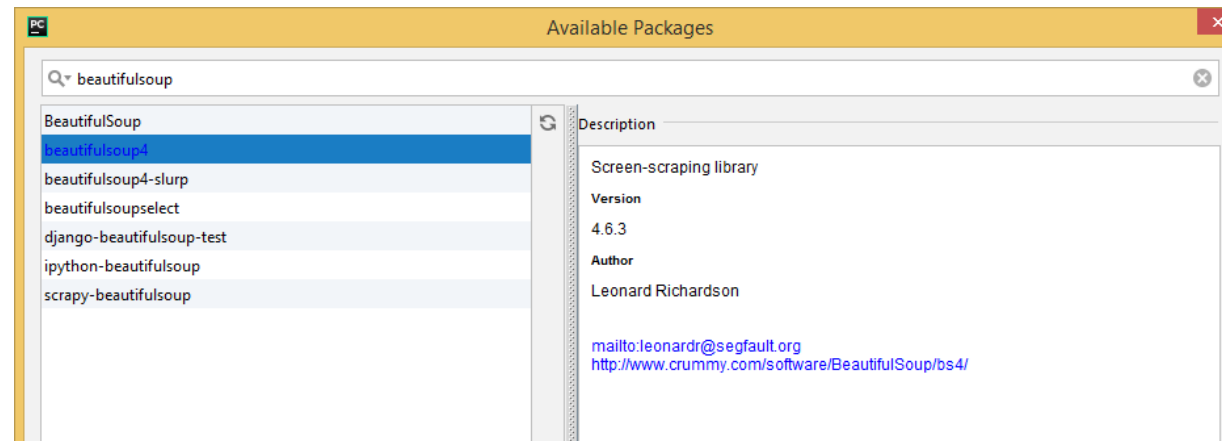
Update time: 2021-08-19T11:49:27+08:00
Carpark number: HE12

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

- Target -> 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

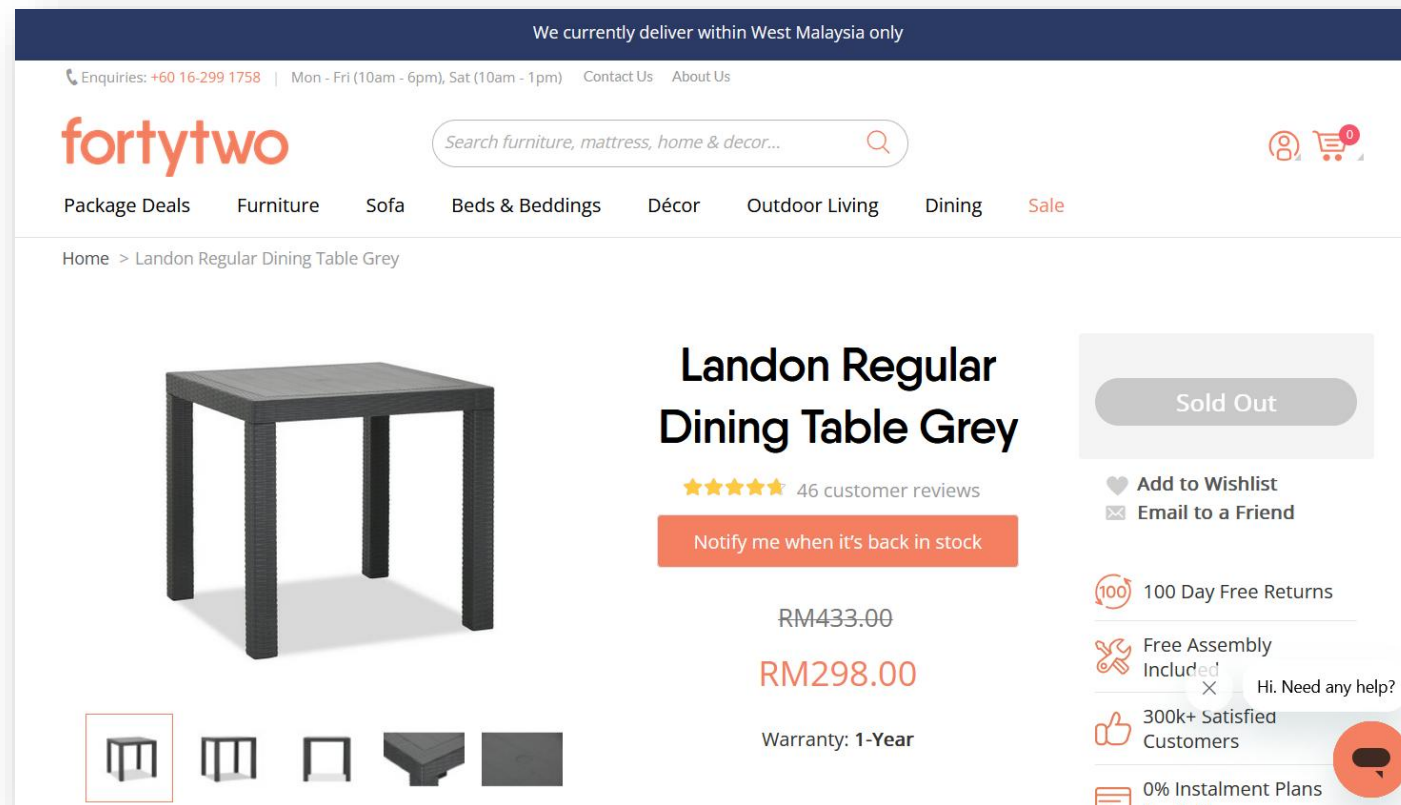
42EXPR Free Assembly Included



Connecting to the Web

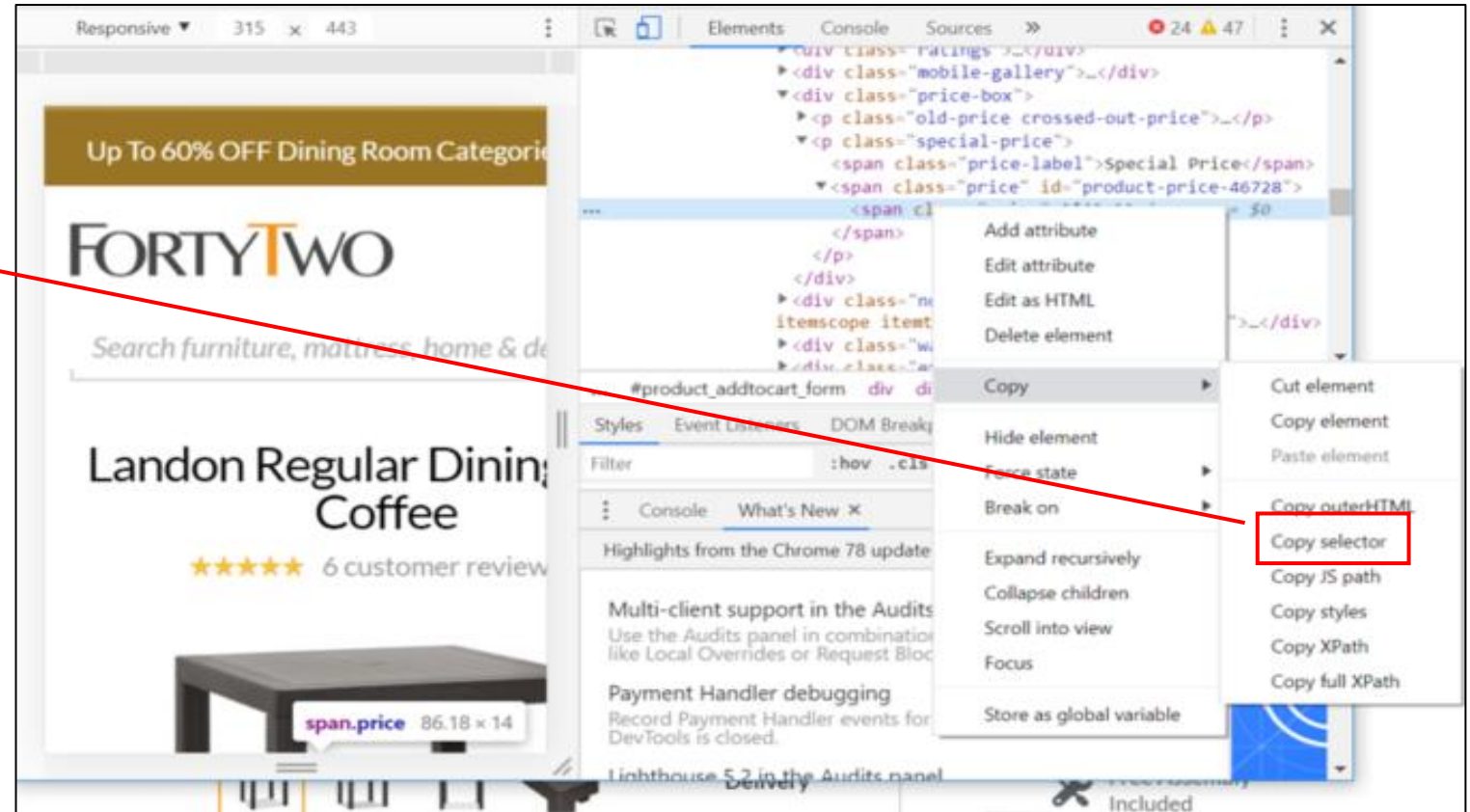
- Target -> What's the URL?

<https://https://www.fortytwo.my/landon-regular-dining-table-grey.html>



Connecting to the Web

- Get the url
- <https://www.fortytwo.my/landon-regular-dining-table-grey.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.my/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 > span:nth-child(1)") # RM298.00
print("Grabbed element elements : \n{}".format(elements))

price = elements[0].text
print("\nCurrent Price: " + price)

elements = soup.select("#old-price-46728") # RM433.00
old_price = elements[0].text
print("\nOld Price: " + old_price)
```

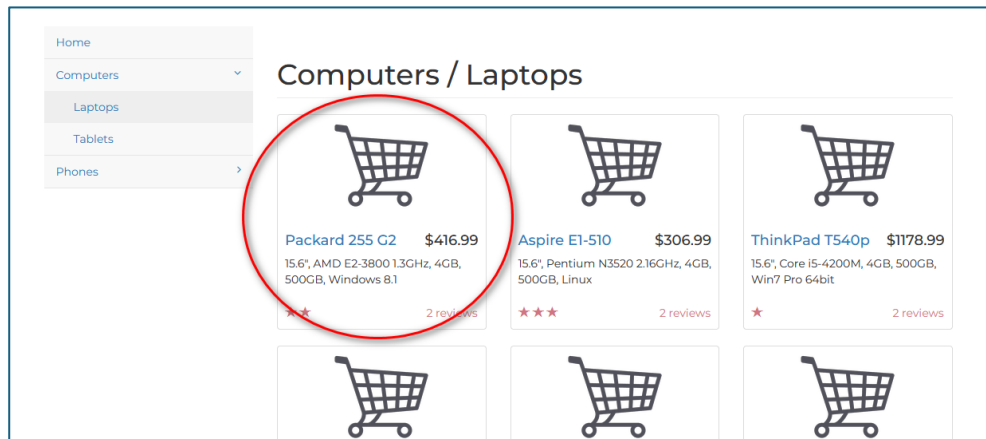
```
Grabbed element elements :
[<span class="price">RM298.00</span>]

Current Price: RM298.00

Old Price:  RM433.00
```

Getting item information via web scraping

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

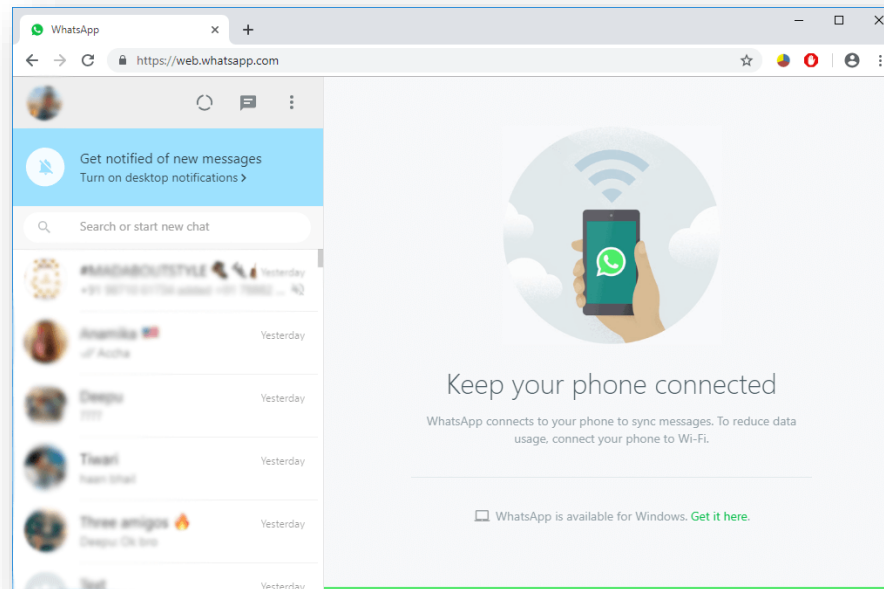


```
Item: Packard 255 G2  
Price: $416.99
```

- Try to fix it, could you?

Sharing other Use Cases

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



selenium python automation

Name

Your answer

Gender

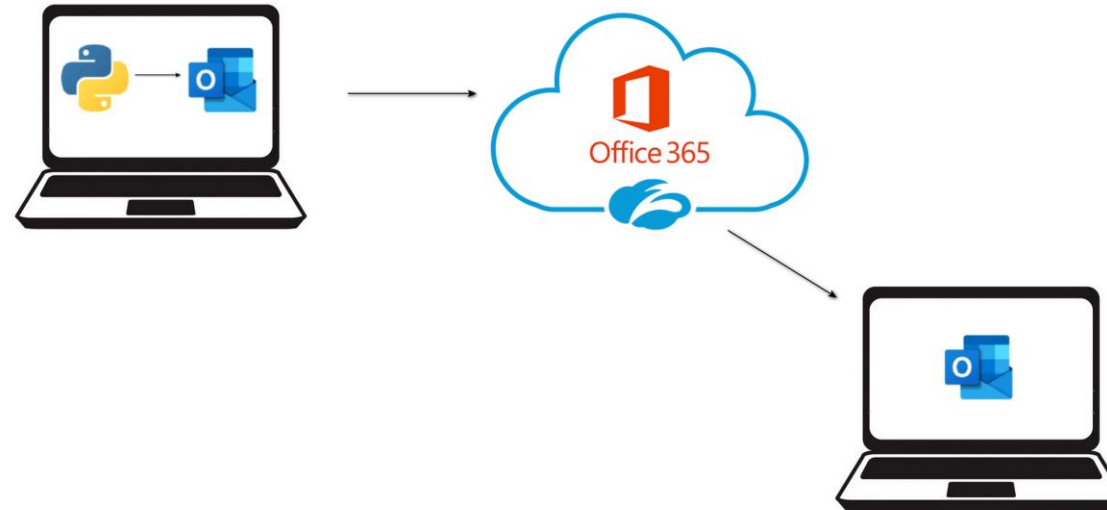
☐ Male

☐ Female



Email Automation with Python

pywin32 package



- This is applicable if 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 configured 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, formatted text in it)
- No credentials are needed for the Python program as Python uses MS Outlook

Create appointment using Outlook

```
import win32com.client

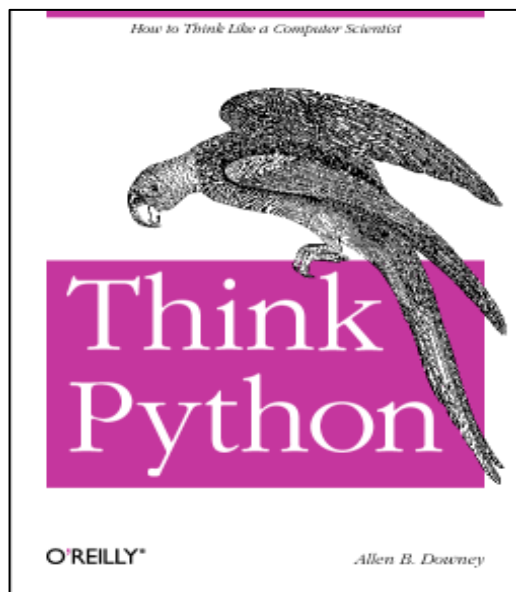
outlook = win32com.client.Dispatch("Outlook.Application")
appt = outlook.CreateItem(1)
appt.Start = "2021-10-08 15:30"
appt.Subject = "Meeting 1"
appt.Duration = 60
appt.Location = "Library"
appt.MeetingStatus = 0
appt.display()
appt.Save()
```

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

Use Case Sharing

- Sending Emails to Students
 - Python can be used to automate email sending via Outlook.
 - Customize emails with personalized content (e.g., names, grades).
 - Ideal for notifying students about announcements, results, or reminders
- Create Appointments using Outlook
 - Schedule meetings, reminders, or events for students directly through Python.
 - Automate appointment creation, including time, date, and location.

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](https://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>

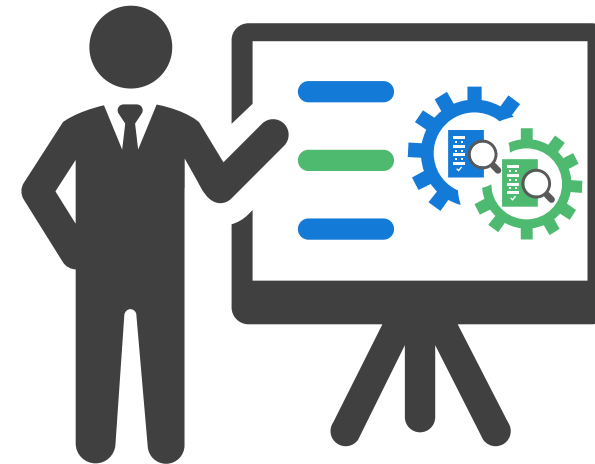
End of Course Survey



Course Run ID:

Please kindly complete this survey before leaving

Thank you



Learning material & source code:
<https://bit.ly/IPP-July2025>

Email us at:
jason_lim@rp.edu.sg