

# Business Process Automation with VBA and Python

Mr. Eddie Chow / 23 November 2024



# Table Of Contents

- Introduction to business process automation
- Business Process automation with VBA
- Business process automation with Python
- Introduction to project management for business process automation
- Development and implementation of business process automation
- Final Group Presentation



## Python – Origin

- Released in 1991 (same as Visual Basic)
- Free and open-source



# Business Process Automation with Python

## Motivation – Soft Limits

- Protective measures not matching modern days

Feature	Maximum limit
Total number of rows and columns on a worksheet	1,048,576 rows by 16,384 columns
Column width	255 characters
Row height	409 points
Page breaks	1,026 horizontal and vertical

1. Microsoft. Excel Specifications and limits. <https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3>

# Business Process Automation with Python

## Motivation – Extensibility

- Python in Excel<sup>1</sup>

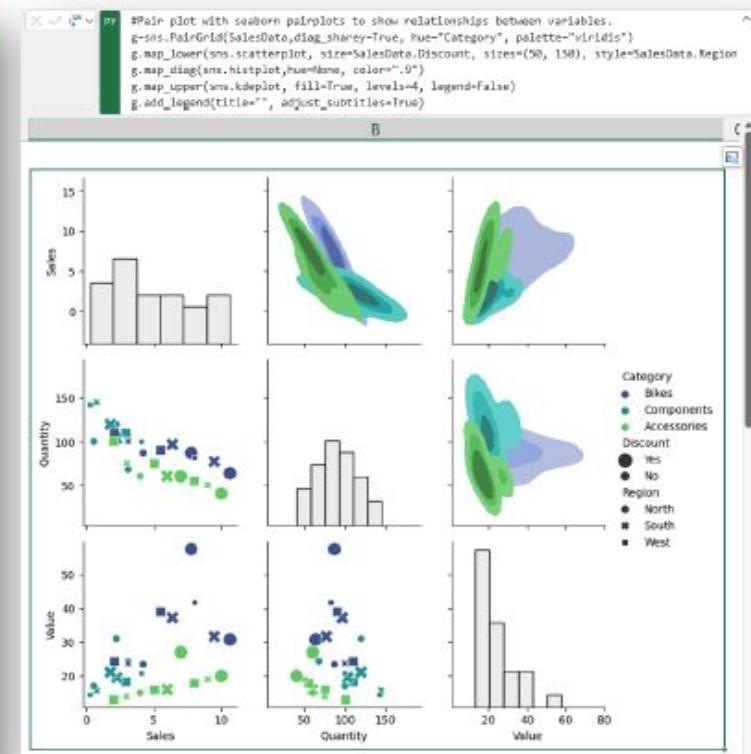
The screenshot shows a Microsoft Excel spreadsheet titled "Python in Excel.xlsx". In the formula bar, there is Python code:

```
#Announcing Python in Excel!
DataFrame=xl("A1:B10", headers=True)
DataFrame.groupby('Category').agg('mean')
```

The data in columns A and B is as follows:

	Category	\$
1	Components	\$ 20
2	Bikes	\$ 17
3	Accessories	\$ 9
4	Bikes	\$ 9
5	Clothing	\$ 8
6	Accessories	\$ 4
7	Clothing	\$ 4
8	Components	\$ 3
9	Components	\$ 1.
10		
11		
12		
13		

On the right side of the screen, there are two data visualization components. One is a bar chart titled "Image" showing sales values for Components, Clothing, Bikes, and Accessories. The other is a scatter plot showing the relationship between Sales and Quantity.



1. Announcing Python in Excel: Combining the power of Python and the flexibility of Excel. TECHCOMMUNITY.MICROSOFT.COM.  
<https://techcommunity.microsoft.com/t5/excel-blog/announcing-python-in-excel-combining-the-power-of-python-and-the/ba-p/3893439>

# Business Process Automation with Python

## Motivation – Scope

- How about outside of Microsoft Office?
- Program trading<sup>1</sup> / Web / AI / Business Intelligence

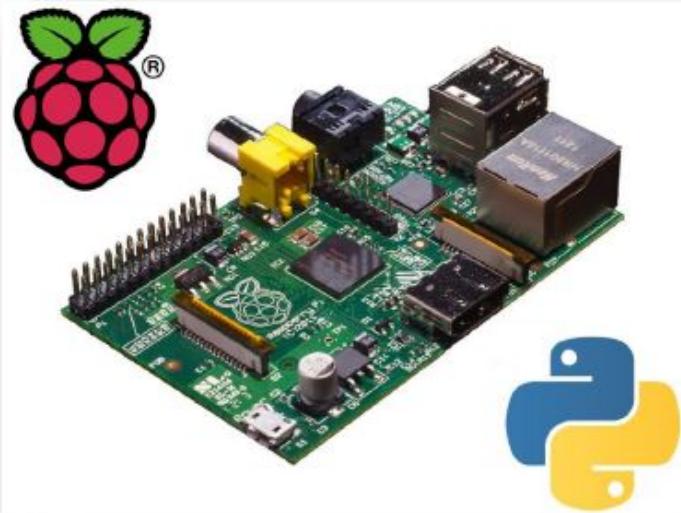


1. Nicholas Pongratz (2021). Sam Bankman Fried Explains His Arbitrage Techniques. Yahoo! Finance. <https://finance.yahoo.com/news/sam-bankman-fried-explains-arbitrage-132901181.html>

# Business Process Automation with Python

## Motivation – Scope

- How about outside of Microsoft Office?
- e.g., Drones / Security cameras / Firewalls



1. *Python and Drones Coding Course for High Schools. CODE4FUN: <https://www.youtube.com/watch?v=FyJrnFUgPA>*

# Business Process Automation with Python

## Motivation – Jobs

- Python → 1 of the top languages for work

### Top Programming Languages 2024

Click a button to see a differently weighted ranking



1. *The Top Programming Languages 2024.* (2024 August 22). IEEE Spectrum.  
<https://spectrum.ieee.org/top-programming-languages-2024>

# Business Process Automation with Python

## Motivation – Jobs

- Python → Recently added as part of CFA exams

### Level II

- Python Programming Fundamentals**

A fundamentals course to demonstrate the basics of Python and how to use Jupyter Notebook for developing, presenting, and sharing data science projects related to finance. (if not taken at Level I)

- Analyst Skills**

Focuses on the skills equity and credit analysts need using insights gained from hundreds of successful analysts.

- Python, Data Science & AI**

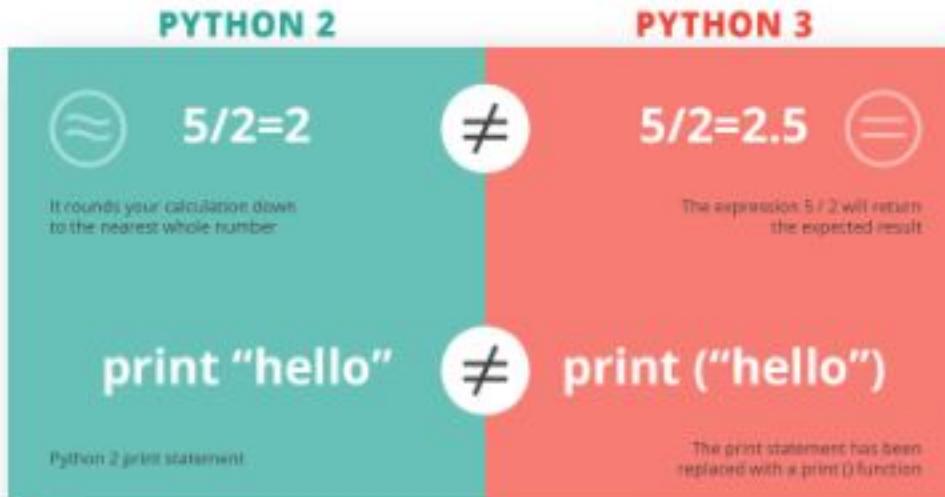
Introduces candidates to machine learning, artificial intelligence, and data science to understand financial statements, reporting, and analysis using Python.

1. *Practical Skills Module | CFA Program Evolution.* Practical Skills Module | CFA Program Evolution.  
<https://evolve.cfainstitute.org/practical-skills-modules.html>

# Business Process Automation with Python

## Python runtime

- The language and basic runtime environment
- Just like Apple iOS, it keeps updating (e.g., Python 3.12.2)
  - Python 2 was no longer supported from 2020<sup>1</sup>



1. Sunsetting Python 2. Python.org. <https://www.python.org/doc/sunset-python-2/>

# Business Process Automation with Python

## Installation – Python Runtime

- Official website
  - <https://www.python.org/downloads/>
- Installing in Windows → Tick “Add python.exe to PATH”

The screenshot shows the Python Downloads page. On the left, there's a large button for "Download the latest version for macOS". To its right, there are two main options: "Install Now" and "Customize installation". The "Install Now" section includes a path: C:\Users\Jacki\AppData\Local\Programs\Python\Python312. It also lists "Includes IDLE, pip and documentation" and "Creates shortcuts and file associations". The "Customize installation" section has a checkbox for "Use admin privileges when installing py.exe" (unchecked) and another for "Add python.exe to PATH" (checked). A yellow arrow points from the "Customize installation" text towards the checked checkbox.

python™

About Downloads Documentation Commu

Download the latest version for macOS

Download Python 3.11.3

→ Install Now  
C:\Users\Jacki\AppData\Local\Programs\Python\Python312

Includes IDLE, pip and documentation  
Creates shortcuts and file associations

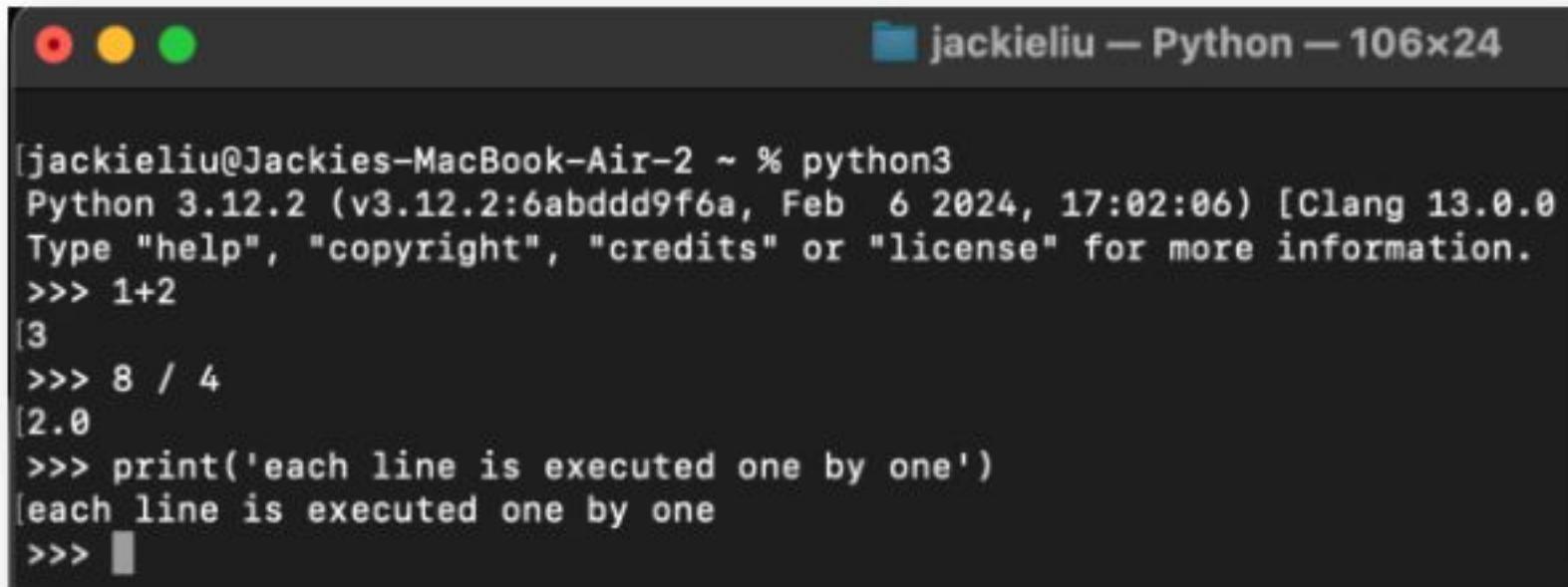
→ Customize installation  
Choose location and features

Use admin privileges when installing py.exe

Add python.exe to PATH

## How to run Python – In the old days

- Python interpreter
  - This is the core of Python
  - Input a line of codes → Press Enter → Run



The screenshot shows a terminal window titled "jackieliu — Python — 106x24". The window contains the following Python session:

```
[jackieliu@Jackies-MacBook-Air-2 ~ % python3
Python 3.12.2 (v3.12.2:6abddd9f6a, Feb 6 2024, 17:02:06) [Clang 13.0.0
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+2
[3
>>> 8 / 4
[2.0
>>> print('each line is executed one by one')
[each line is executed one by one
>>> ]
```

# Business Process Automation with Python

## How to run Python – Running as a file

- Python interpreter + Text editor
  - Write codes in a text editor (e.g., Notepad, Vim, Sublime Text)
  - Save codes as a “.py” file
  - Run all the lines in one go

The image shows two windows side-by-side. On the left is a text editor window titled 'test.py'. The code inside is:

```
print(f'1 + 2 = {1 + 2}')
```

On the right is a 'Command Prompt' window. The command entered is:

```
C:\Users\Jacki\Documents>python test.py
```

The output of the command is:

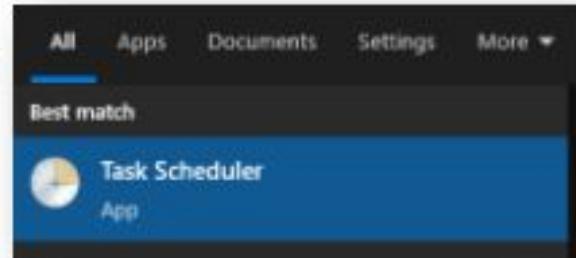
```
1 + 2 = 3
```

Below the output, the prompt 'C:\Users\Jacki\Documents>' is visible.

# Business Process Automation with Python

## How to run Python – Running as a file (bonus)

- Python interpreter + Text editor + Task Scheduler
  - Running a Python script daily/ hourly
    - e.g., Task Scheduler/ Cron / Autosys etc
  - Create a task pointing to the .py file

A screenshot of the Windows Task Scheduler application window. The window title is 'Task Scheduler'. The menu bar includes 'File', 'Action', 'View', and 'Help'. The toolbar contains icons for back, forward, search, and other functions. On the left, a navigation pane shows 'Task Scheduler (Locally)' and 'Task Scheduler'. The main pane displays a table of tasks. One task, 'Test Python', is listed with the following details:

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Auth...
Test Python	Running	At 1:12 PM every day	11/4/2023 1:12:00 AM	30/11/1999 12:00:00 AM	(0x41303)	HKU...

A context menu is open over the 'Test Python' task, listing options: 'Create Basic Task...', 'Create New Task...', 'Import Task...', and 'Refresh'. The 'Create New Task...' option is highlighted with a blue background and white text.

# Business Process Automation with Python

## How to run Python – IDE

- Integrated Development Environment (IDE)
  - A smart editor tailored for writing codes
    - e.g., Syntax highlighting, variable tracking, debugging mode
    - May support useful extensions/plugins (e.g., GitHub Copilot)
  - Common IDEs
    - e.g., Spyder, PyCharm, Visual Studio Code



# Business Process Automation with Python

## Example - PyCharm

The screenshot shows the PyCharm IDE interface. On the left, there's a file browser with files 'notebook.ipynb' and 'notebook.py'. The main area displays a Jupyter notebook cell containing Python code for data analysis. The code imports pandas, matplotlib.pyplot, and numpy, reads a CSV file, handles missing values, performs feature engineering by creating a 'AreaCategory' column, prints group statistics, does descriptive analysis, and creates a scatter plot of SalePrice vs Br-Liv-Area. To the right of the code, a scatter plot titled 'Scatter plot of SalePrice grouped by AreaCategory' is shown, with points colored by their 'AreaCategory'. Below the plot is a 'Data View' window showing a portion of the DataFrame.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv("/Users/Stanislav.Garkusha/Downloads/Shad_Python_81.2/Ames_dataset/AmesHousing.csv", na_values="?")

# Handle Missing Values
# Use apply function to apply a specific function across each column of the DataFrame
df = df.apply(lambda x: x.fillna(x.mean()) if x.dtype.kind in 'biufc' else x.fillna(x.mode()[0]))

# Feature Engineering
df["AreaCategory"] = pd.cut(df["Br-Liv-Area"], bins=[0, 1000, 2000, df["Br-Liv-Area"].max()], labels=["small", "medium", "large"], include_lowest=True)

print(df.groupby("AreaCategory")[
      "SalePrice"].mean()) # printing mean sales price for small, medium, and large living areas

# Statistical Analysis
print(df.describe()) # prints descriptive statistics of all numerical columns

# Data Visualization
fig, ax = plt.subplots()
ax.scatter(df["Br-Liv-Area"], df["SalePrice"], alpha=0.5)
ax.set_title('Scatter plot of Br-Liv-Area vs SalePrice')
ax.set_xlabel('Br-Liv-Area')
ax.set_ylabel('SalePrice')

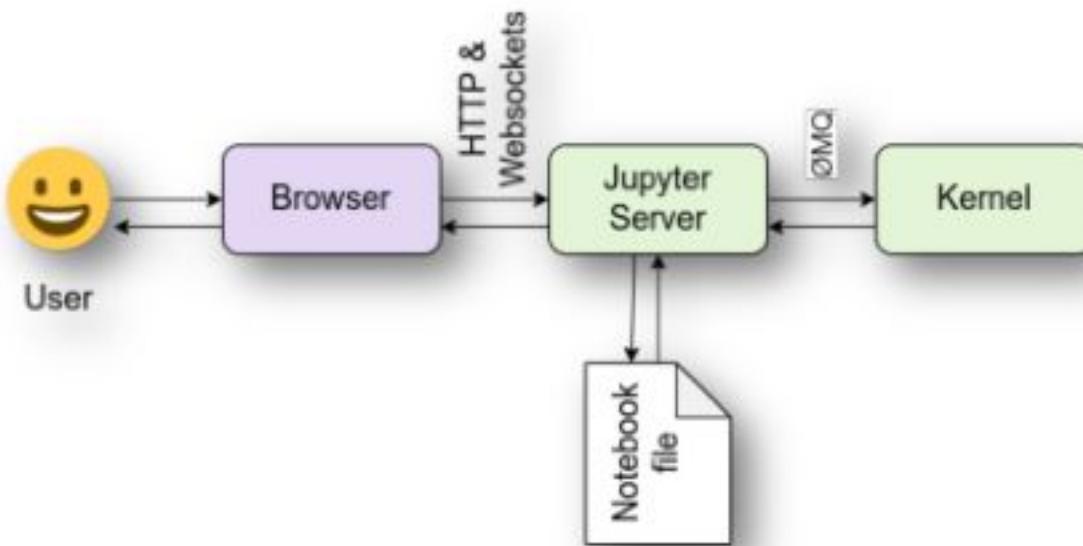
# Scatter plot instead of boxplot
fig, ax = plt.subplots()
area_categories = ['small', 'medium', 'large']
for category in area_categories:
    ax.scatter(df[df['AreaCategory'] == category]['Br-Liv-Area'], df[df['AreaCategory'] == category]['SalePrice'], alpha=0.5)
```

#	Order	P.D.	MS SubClass	M.S.
1	1553	1554	9102510...	20
2	2903	2904	923125...	20
3	942	943	9111030...	50
4	727	728	9024771...	30
5	726	727	9024771...	30
6	1557	1558	9112260...	30
7				C (all)

# Business Process Automation with Python

## How to run Python – Jupyter Notebook

- Web-based interactive platform
  - Accessible: Python runtime on a web server



1. Architecture — Jupyter Documentation 4.1.1 Alpha Documentation.  
<https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html>

## How to run Python – Jupyter Notebook

- Notebook
  - Self-explainable → Code & Text blocks
  - Cached results



Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#) using windowing, to reveal the frequency content of a sound signal.

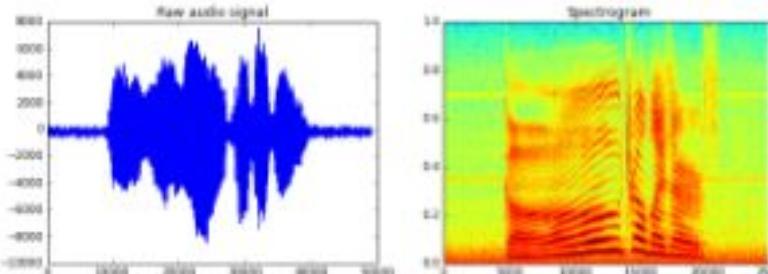
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j\pi}{N} kn} \quad k = 0, \dots, N-1$$

We begin by loading a dataset using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile  
rate, x = wavfile.read('test_wav.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline  
from matplotlib import pyplot as plt  
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 4))  
ax1.plot(x); ax1.set_title('Raw audio signal')  
ax2.specgram(x); ax2.set_title('Spectrogram')
```



# Business Process Automation with Python

## How to run Python – Jupyter Notebook

- Google Colaboratory (Colab)
  - Jupyter Notebook powered by Google Cloud <sup>1</sup>
    - <https://colab.research.google.com/>
  - Benefits <sup>2</sup>
    - “Zero configuration required”
    - Easy sharing
    - Free access: CPU & GPU



1. Google Colab FAQ. <https://research.google.com/colaboratory/faq.html>

2. Google Colaboratory. <https://colab.research.google.com>

## How to run Python – Summary

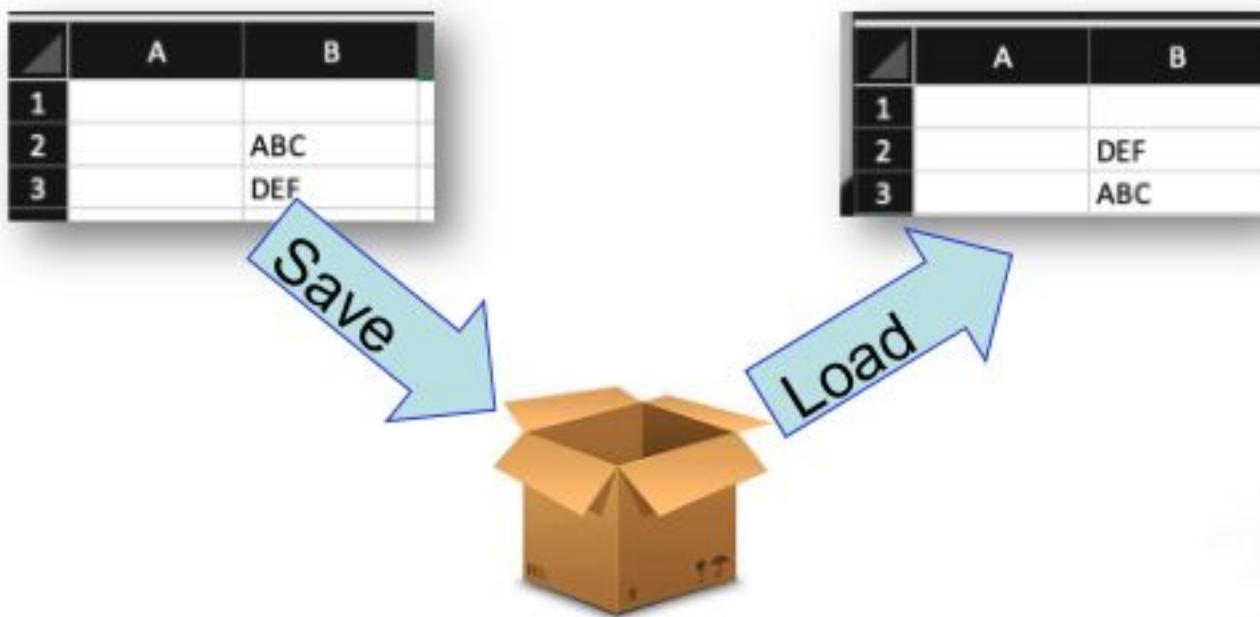
- Python interpreter
- Python (.py) files with
  - Text editors
  - IDEs
- Jupyter Notebook running
  - Locally
  - Remotely



# Business Process Automation with Python

## Variables

- From the previous lecture
  - Variables → Storage of values
  - Values → carry data types



## Variables

- Declaration
  - In VBA, declarations (“dim”) are recommended
    - Dim year As **Integer**
    - year = **2024**
- In Python, variables are created during **assignment**
  - year = **2024**
- To assign without an actual value
  - year = **None**



# Business Process Automation with Python

## Value Check

- VBA
  - Debug.Print()

```
Sub HighlightCell()
    If Range("B1").Value > 50 Then
        Range("B1").Interior.Color = vbYellow
    Else
        Range("B1").Interior.Color = vbGreen
    End If
    Debug.Print (Range("B1"))
End Sub
```

40

- Python
  - print()
  - Last line in a block

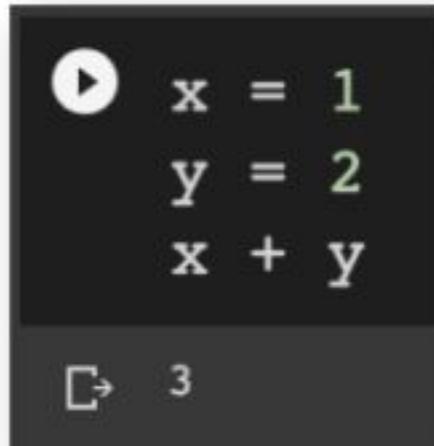
```
▶ print(1 + 2)          # This gives 3
  print(1 + 2 + 3)      # This gives 6
□ 3
  6
```

```
▶ 1 + 2          # This is not shown
  1 + 2 + 3      # This is shown
□ 6
```

# Business Process Automation with Python

## Primitive Data Types - Numeric

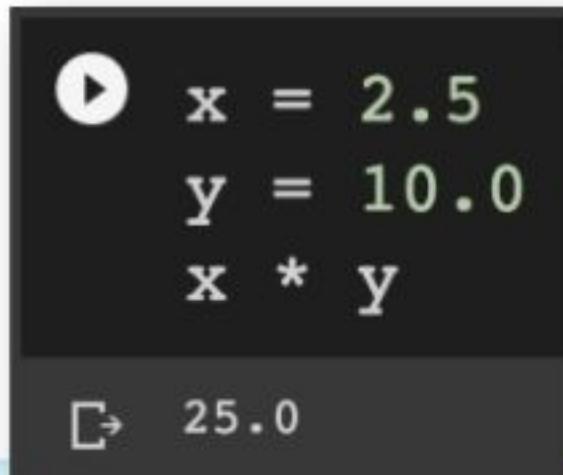
- Int



```
x = 1  
y = 2  
x + y  
3
```

A screenshot of a Jupyter Notebook cell. The cell contains Python code: 'x = 1', 'y = 2', and 'x + y'. The output below the code cell shows the result '3'.

- Float



```
x = 2.5  
y = 10.0  
x * y  
25.0
```

A screenshot of a Jupyter Notebook cell. The cell contains Python code: 'x = 2.5', 'y = 10.0', and 'x \* y'. The output below the code cell shows the result '25.0'.

## Primitive Data Types - Numeric

- Complex (For scientific calculations)

### THE QUADRATIC FORMULA

© CHILIMATH.COM

If  $ax^2 + bx + c = 0$  but  $a \neq 0$

then

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

DISCRIMINANT

- $b^2 - 4ac > 0$  two real solutions
- $b^2 - 4ac = 0$  one real solutions
- $b^2 - 4ac < 0$  zero real solutions

▶  $x = 3 + 4j$   
 $y = 2 + 2j$   
 $x - y$   
⇒  $(1+2j)$

$i = \sqrt{-1}$

## Primitive Data Types - Numeric

- Common Functions
  - `round()` / `pow()` / `abs()`

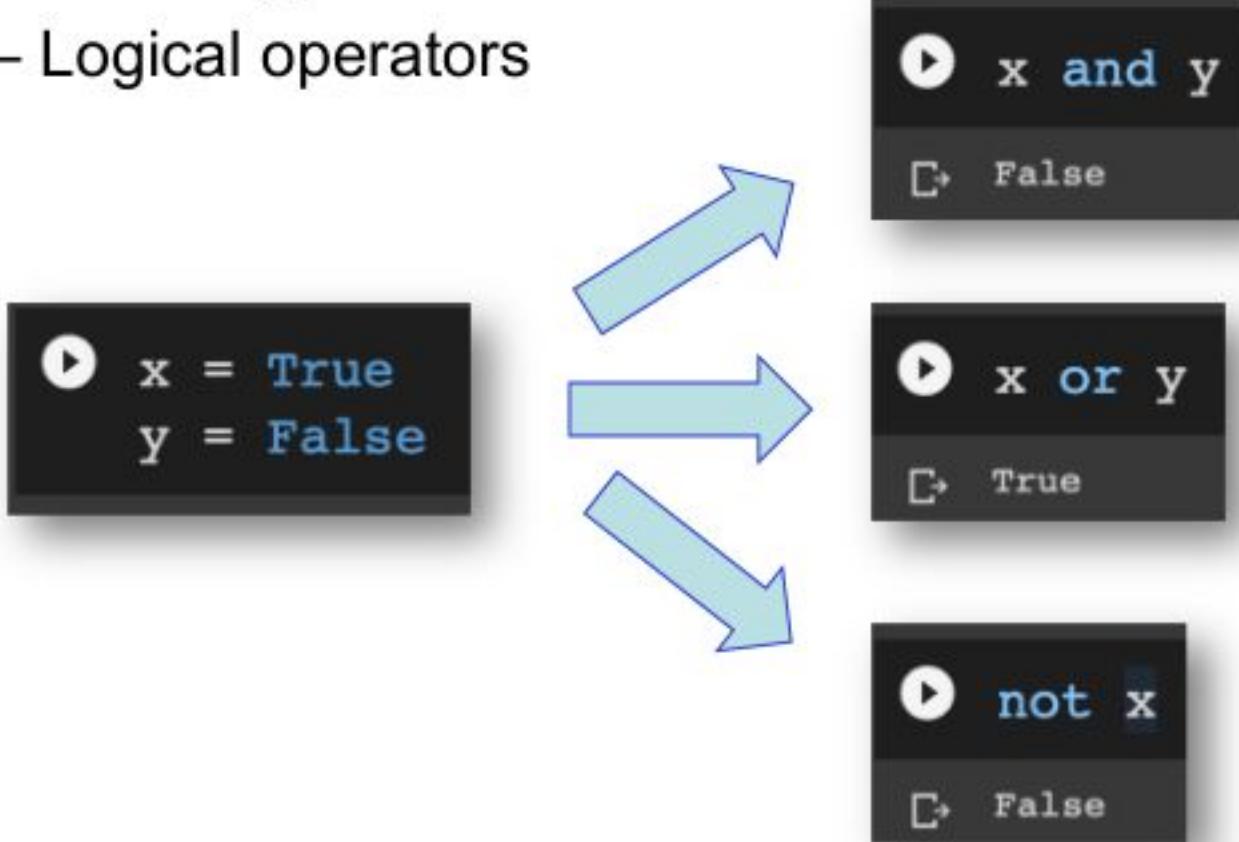
```
▶ print(round(9876.54321, 2))      # Round to 2 decimal places
    print(round(9876.54321, 0))      # Round to nearest integer
    print(pow(2, 3))                  # 2 * 2 * 2 = 8
    print(abs(-5))                   # Negative numbers become positive
```

```
▷ 9876.54
  9877.0
  8
  5
```

# Business Process Automation with Python

## Primitive Data Types - Boolean

- Bool – Logical operators



# Business Process Automation with Python

## Primitive Data Types - Boolean

- Bool – comparators
  - Generates True/ False

Operator	Name	Example
<code>==</code>	Equal	<code>5 == 5</code>
<code>!=</code>	Not equal	<code>26 != 3</code>
<code>&gt;</code>	Greater than	<code>100 &gt; 67</code>
<code>&lt;</code>	Less than	<code>89 &lt; 216</code>
<code>&gt;=</code>	Greater than or equal to	<code>90 &gt;= 54</code>
<code>&lt;=</code>	Less than or equal to	<code>23 &lt;= 77</code>

## Primitive Data Types - Text

- String
  - Representation

```
▶ x = 'This is a string'                      # Single quote
    y = "This is also a string"                # Double quote
    z = '''This is a very long string
spanning across more than 1 line'''          # Multi-line

    print(x)
    print(y)
    print(z)

⇒ This is a string
    This is also a string
    This is a very long string
        spanning across more than 1 line
```

## Primitive Data Types - Text

- String
  - Concatenation → “+” → glue 2 strings together

```
▶ x = 'This is a string'  
    y = "This is also a string"
```

```
▶ print(x + ' and ' + y) # Adding 2 strings together  
◀ This is a string and This is also a string
```

# Business Process Automation with Python

## Primitive Data Types - Text

- String
  - Concatenation → Number vs Text



```
lobster_price = 936
lobster_text = str(lobster_price)
print(2 * lobster_price) # 1872
print(2 * lobster_text) # 936936
```

1872  
936936

```
▶ print(10 * '=')
▶ print('WELCOME')
▶ print(10 * '=')
```

```
=====
WELCOME
=====
```

- 星島日報. (2023, May 6). 荃灣中菜館驚現「天價龍蝦」兩隻竟索價90萬元 酒樓親解釋. Singtaousa.com; 星島日報.  
<https://www.singtaousa.com/2023-05-06/%e8%bd%83%e7%81%a3%e4%b8%ad%e8%8f%9c%e9%a4%a8%e9%a9%9a%e7%8f%be%e3%80%8c%e5%a4%a9%e5%83%b9%e9%be%8d%e8%9d%a6%e3%80%8d-%e5%85%a9%e9%9a%bb%e7%ab%9f%e7%b4%a2%e5%83%b990%e8%90%ac%e5%85%83-%e9%85%92/4488305>

## Primitive Data Types - String formatting

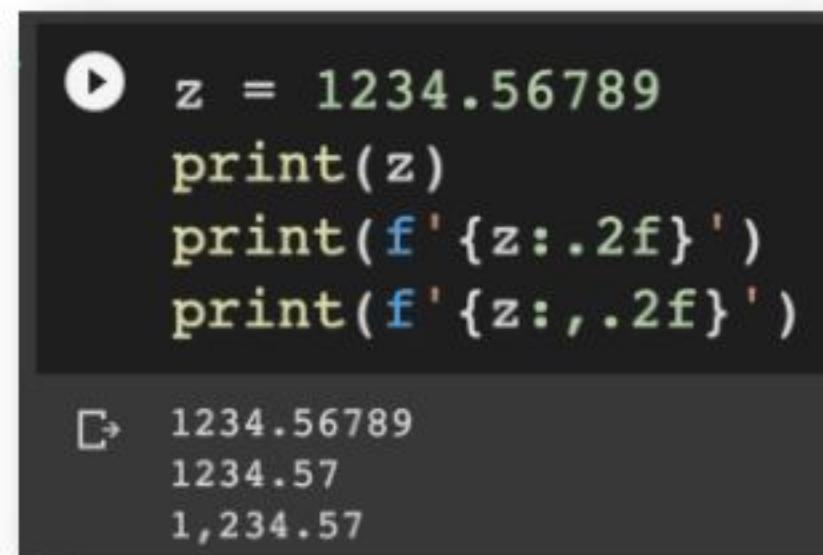
- F-String (Python 3.6 or newer)
  - A smart way to format strings

```
▶ x = 4
  y = 8.8
  f'x is {x} and y is {y}; y divided by x is {y / x}'
⇒ 'x is 4 and y is 8.8; y divided by x is 2.2'
```

1. *Input and Output.* Python Documentation. <https://docs.python.org/3/tutorial/inputoutput.html>

## Primitive Data Types - String formatting

- F-String (Python 3.6 or newer)
  - A smart way to format strings (advanced)



The screenshot shows a terminal window with a dark background. On the left, there is a play button icon. The code is displayed in white text:

```
z = 1234.56789
print(z)
print(f'{z:.2f}')
print(f'{z:,.2f}')
```

On the right, the output is shown in white text:

```
1234.56789
1234.57
1,234.57
```

1. *Input and Output*. Python Documentation. <https://docs.python.org/3/tutorial/inputoutput.html>

## Primitive Data Types - String

- Common Functions
  - upper() / lower() / replace()

```
▶ # Common functions
    test_string = 'Hello Hong Kong!'
    print(test_string.upper())
    print(test_string.lower())
    print(test_string.replace('Hello', 'Bello'))
```

```
⇨ HELLO HONG KONG!
    hello hong kong!
    Bello Hong Kong!
```

# Business Process Automation with Python

## Primitive Data Types - Conversion

- **VBA**
  - CInt() / CDbl() / Format()
- **Python**
  - int() / float() / str() / bool()

```
▶ int(4.5)
◀ 4
```

```
▶ float(5)
◀ 5.0
```

```
▶ str(5.55)
◀ '5.55'
```

❗

```
▶ bool('Some values')
◀ True
```

```
▶ bool('')
◀ False
```

# Business Process Automation with Python

## Methods

- VBA
  - Sub-routine: reusable code blocks
  - Function: sub-routine which gives an output
- Python
  - Methods: Use “`return`” if output is required

VBA

```
Function AddNumbers(x As Double, y As Double) As Double
    AddNumbers = x + y
End Function
```

```
?AddNumbers(1,2)
3
```

Python

```
def add_numbers(x, y):
    return x + y
```

```
def add_numbers(x, y):
    return x + y

print(add_numbers(1,2))
```

# Business Process Automation with Python

## Practice 1 – Market Capitalization

- Compute market capitalization of 2 companies
- Show the difference between the two
- Format the result up to 2 decimal places



- **Starting notebook:**

[https://github.com/innoviai/ipa\\_courses/blob/main/Lecture%202/practice1\\_simple\\_calc\\_202409.ipynb](https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice1_simple_calc_202409.ipynb)

# Business Process Automation with Python

## Sequential Data Types

- **Motivation**

- How to represent the ordering of schools?
  - e.g., HKU > CUHK > HKUST

+ Rank	University	Overall Score
21	 The University of Hong Kong Hong Kong, Hong Kong SAR	87
38	 The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR	80.6
40	 The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR	79.8

\* QS Rankings 2023

# Business Process Automation with Python

## Sequential Data Types

- Motivation

- How to represent the ordering of schools?
  - e.g., HKU > CUHK > HKUST
  - Preferably with 1 variable...

+ Rank	University	Overall Score
21	The University of Hong Kong Hong Kong, Hong Kong SAR	87
38	The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR	80.6
40	The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR	79.8

\* QS Rankings 2023

```
▶ school_1 = 'hku'  
school_2 = 'cuhk'  
school_3 = 'hkust'  
print(school_1)  
print(school_2)  
print(school_3)
```

⇨ hku  
cuhk  
hkust

## Sequential Data Types

- Tuple
  - Representation

```
[55] school_tuple = ('hku', 'cuhk', 'hkust')
      print(school_tuple)
      print(type(school_tuple))

('hku', 'cuhk', 'hkust')
<class 'tuple'>
```

- Indexing (zero-based)

```
▶ print(school_tuple[0])
print(school_tuple[1])
print(school_tuple[2])

↪ hku
cuhk
hkust
```

## Sequential Data Types

- **Tuple**

```
school_tuple = ('hku', 'cuhk', 'hkust')
```

- Useful functions
  - Get the first occurrence

```
▶ school_tuple.index('cuhk')
```

```
▷ 1
```

- Number of items

```
▶ len(school_tuple)
```

```
▷ 3
```

- Number of specific element

```
▶ ('male', 'male', 'female').count('male')
```

```
▷ 2
```

## Sequential Data Types

- **List**

- “Mutable” object (something you can change)
- Representation

```
▶ school_list = ['hku', 'cuhk', 'hkust']
    print(school_list)
    print(type(school_list))

◀ ['hku', 'cuhk', 'hkust']
<class 'list'>
```

- Indexing

```
▶ print(school_list[0])      # First 1 item
    print(school_list[0:2])    # First 2 items
    print(school_list[-1])     # Last one

◀ hku
['hku', 'cuhk']
polyu
```

# Business Process Automation with Python

## Sequential Data Types

- **List**

- Adding items

```
➊ school_list = ['hku', 'cuhk', 'hkust'] + ['cityu', 'polyu']
school_list
```

```
➋ ['hku', 'cuhk', 'hkust', 'cityu', 'polyu']
```

```
➊ school_list = ['hku', 'cuhk', 'hkust', 'cityu']
school_list.append('polyu')
school_list
```

```
➋ ['hku', 'cuhk', 'hkust', 'cityu', 'polyu']
```

## Sequential Data Types

- **List**

- Removing an item
  - By index → `pop()`

```
▶ school_list.pop(1)
school_list
[ 'hku', 'hkust', 'cityu', 'polyu' ]
```

- By element → `remove()`

```
▶ school_list.remove('cityu')
school_list
[ 'hku', 'hkust', 'polyu' ]
```

## Sequential Data Types

- Common operations

- Membership Check → “in”

```
▶ # Membership check
    school_list = ['hku', 'cuhk', 'hkust', 'cityu', 'polyu']
    print('hku' in school_list)
    print('hkbu' in school_list)

◀ True
False
```

## Sequential Data Types

- Common operations

- Ordering → max() / min() / reversed() / sorted()

```
▶ number_list = [1, 5, 6, 2, 3]
    print(max(number_list))                      # Max
    print(min(number_list))                      # Min
    print(list(reversed(number_list)))          # Reverse order
    print(sorted(number_list))                  # Sort by ascending order

▷ 6
1
[3, 2, 6, 5, 1]
[1, 2, 3, 5, 6]
```

## Sequential Data Types

- String (again)
  - string = sequential!



```
# Index:          0123456789
school_name = 'HKU School of Professional and Continuing Education'

print(len(school_name))      # There is a total of 51 letters
print(school_name.index('c'))  # First c is the 6th letter (counting 1 space)
print(school_name.count('o'))  # There are 7 O's
print('HKU' in school_name)   # Substring matching
print(school_name[4:10])       # Substring extraction
```

```
51
5
7
True
School
```

# Business Process Automation with Python

## Sequential Data Types

- String (again)
  - Conversions

```
▶ # Conversion: String -> list
  print(list('hello'))
```

▷ ['h', 'e', 'l', 'l', 'o']

```
▶ # Conversion: List -> String
  print(''.join(['h', 'e', 'l', 'l', 'o']))
  print(','.join(['Welcome', 'Jackie!']))
```

▷ hello  
Welcome,Jackie!

```
▶ # Conversion: String <-> int?
  print(ord('a'))
  print(chr(98))
```

▷ 97  
b

# Business Process Automation with Python

## Practice 2 – Word Count

- Given an input text → Show below 3 fields
  - Reference: <https://charcounter.com/en/>

**Charcounter**  
Character, Letter and Word Counter 

356 Characters	50 Words	307 Without White Space
----------------	----------	-------------------------

The programme aims to impart the essential knowledge of process automation to students and equip them with automation techniques in the business. It adopts a contemporary project management approach to business process automation. It also examines the emerging business opportunities and challenges in its planning and implementation of process automation.

- Starting notebook:**

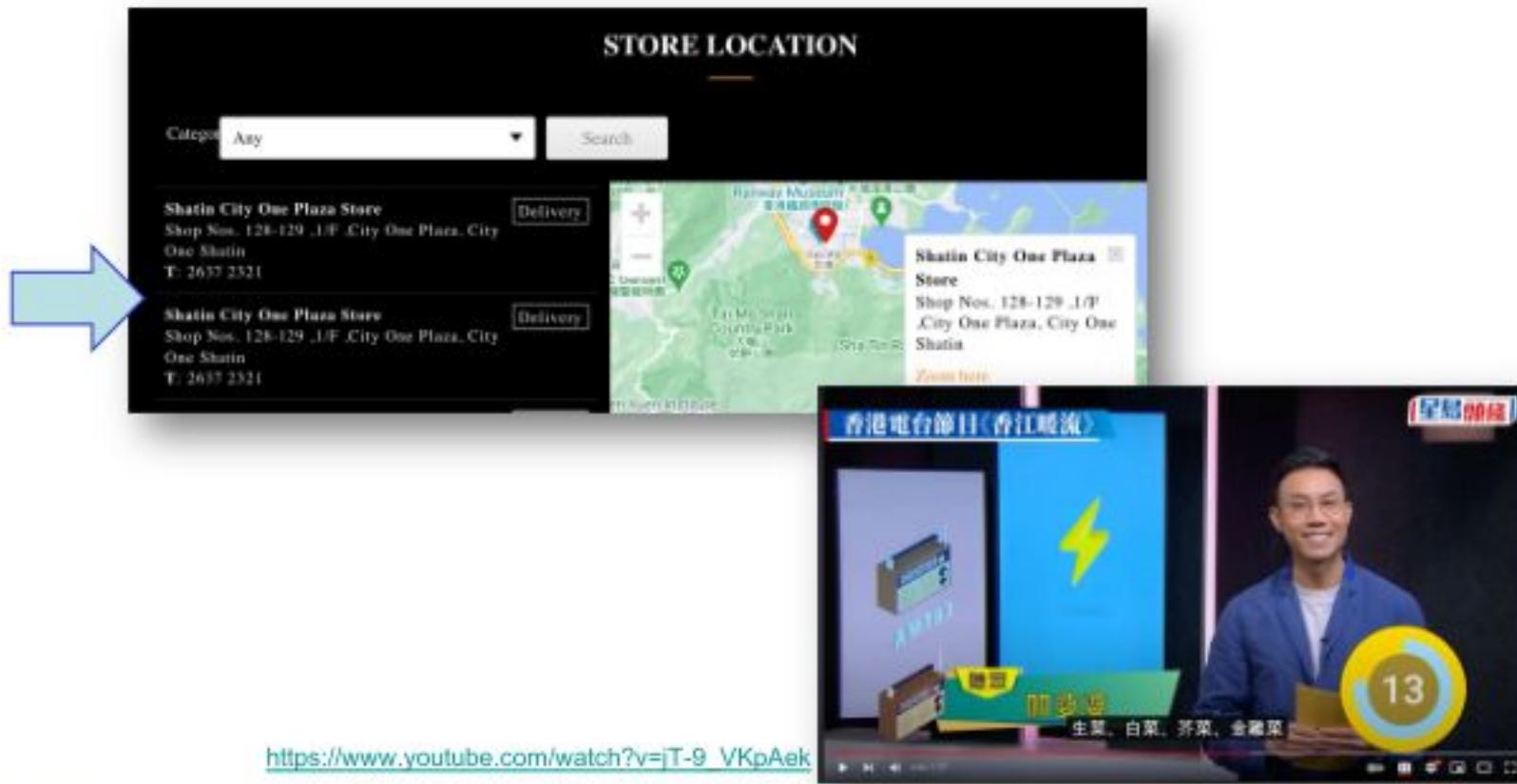
[https://github.com/innoviai/ipa\\_courses/blob/main/Lecture%202/practice2\\_word\\_count\\_202409.ipynb](https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb)

# Business Process Automation with Python

## Data Type - Set

- Motivation

- Problem → Duplicate data are everywhere



[https://www.youtube.com/watch?v=jT-9\\_VKpAek](https://www.youtube.com/watch?v=jT-9_VKpAek)

## Data Type - Set

- **Set**
  - A way to maintain unique values

```
▶ duplicate_list = ['Shatin City One', 'Shatin City One', 'TKO Gateway']
      set(duplicate_list)
[▶ ('Shatin City One', 'TKO Gateway')
```

## Data Type - Set

- Set
  - Representation

```
▶ school_set = {'A', 'B', 'B', 'C'}
    print(school_set)
    print(type(school_set))

⇒ {'B', 'C', 'A'}
<class 'set'>
```

- Can be converted between sequential data types

```
▶ print(set(['A', 'B', 'B', 'C'])) # List -> Set
    print(list({'A', 'B', 'C'}))      # Set -> List

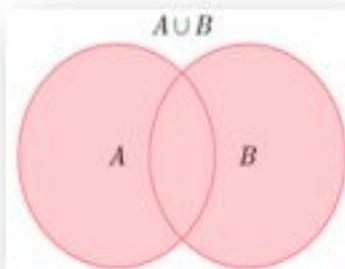
⇒ {'C', 'B', 'A'}
['B', 'C', 'A']
```

# Business Process Automation with Python

## Data Type - Set

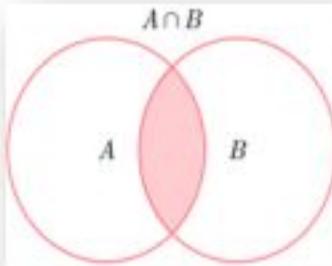
- Set operations

- Union



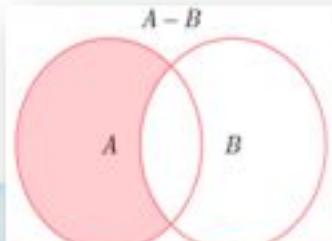
```
❶ # Set operations
school_set_2022 = {'A', 'B', 'C'}
school_set_2023 = {      'B', 'C', 'D'}
```

- Intersection



```
❶ # Union: items in any one of the sets
school_set_2022 | school_set_2023
❷ {'A', 'B', 'C', 'D'}
```

- Difference



```
❶ # Intersection: items in both of the sets
school_set_2022 & school_set_2023
❷ {'B', 'C'}
```

```
❶ # Difference: Items in 1 set but not in the other
school_set_2023 - school_set_2022
❷ {'D'}
```

# Business Process Automation with Python

## Data Type - Dictionary

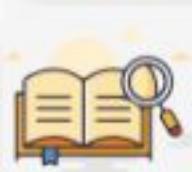
- Motivation

- Business problems often involve key-value pairs
- e.g., Employee ID → Full Name / Job Title



A	B	C	D
Employee ID	Full Name	Job Title	Department
E02002	Kai Le	Controls Engineer	Engineering
E02003	Robert Patel	Analyst	Sales
E02004	Cameron Lo	Network Administrator	IT
E02005	Harper Castillo	IT Systems Architect	IT
E02006	Harper Dominguez	Director	Engineering
E02007	Ezra Vu	Network Administrator	IT
E02008	Jade Hu	Sr. Analyst	Accounting
E02009	Miles Chang	Analyst II	Finance
E02010	Gianna Holmes	System Administrator	IT
E02011	Jameson Thomas	Manager	Finance
E02012	Jameson Pena	Systems Analyst	IT
E02013	Bella Wu	Sr. Analyst	Finance

- Looking up a keyword  
→ like finding words in **ictionaries**



# Business Process Automation with Python

## Data Type - Dictionary

- Representation

```
# Representation: option 1
school_rank_dict = {'hku': 21, 'cuhk': 38, 'hkust': 40}
print(type(school_rank_dict))

# Representation: option 2
school_rank_dict = dict(hku=21, cuhk=38, hkust=40)
print(school_rank_dict)

<class 'dict'>
{'hku': 21, 'cuhk': 38, 'hkust': 40}
```

+ Rank	- University
21	 The University of Hong Kong Hong Kong, Hong Kong SAR
38	 The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR
40	 The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR

- Lookup → get()

```
# Lookup
print(school_rank_dict['cuhk'])           # Direct lookup
print(school_rank_dict.get('hkust'))       # get(): Success
print(school_rank_dict.get('test'))         # get(): Failed
print(school_rank_dict.get('test', 'N/A'))  # get(): Error handling
```

38  
40  
None  
N/A

## Data Type - Dictionary

- Adding an item

```
▶ # Adding an item
school_rank_dict = {'hku': 21, 'cuhk': 38, 'hkust': 40}
school_rank_dict['cityu'] = 54
print(school_rank_dict)

▶ {'hku': 21, 'cuhk': 38, 'hkust': 40, 'cityu': 54}
```

- Removing an item

```
▶ # Removing an item
school_rank_dict = {'hku': 21, 'cuhk': 38, 'hkust': 40, 'cityu': 54}
school_rank_dict.pop('cityu')
print(school_rank_dict)

▶ {'hku': 21, 'cuhk': 38, 'hkust': 40}
```

# Business Process Automation with Python

## Data Type - Dictionary

- Listing out the details

```
▶ # Show all the keys
print(f'keys: {list(school_rank_dict.keys())}')
```

```
▶ # Show all the values
print(f'velues: {list(school_rank_dict.values())}')
```

```
▶ # Show all the items
print(f'items: {list(school_rank_dict.items())}')
```

```
↳ keys: ['hku', 'cuhk', 'hkust']
values: [21, 38, 40]
items: [('hku', 21), ('cuhk', 38), ('hkust', 40)]
```

- Membership Check

```
▶ # Membership Check
print('hku' in school_rank_dict)
```

```
↳ True
```

## Data Type - Dictionary

- Conversion

- Cast from a list of tuples

	A	B	C
1	Employee ID	Full Name	Job Title
2	E02002	Kai Le	Controls Engineer
3	E02003	Robert Patel	Analyst
4	E02004	Cameron Lo	Network Administrator

```
# Conversion: list => dict
employee_name_list = [('E02002', 'Kai Le'), ('E02003', 'Robert Patel')]
print(dict(employee_name_list))

{'E02002': 'Kai Le', 'E02003': 'Robert Patel'}
```

# Business Process Automation with Python

## Data Type - Dictionary

- Conversion
  - zip() can be used to create tuples

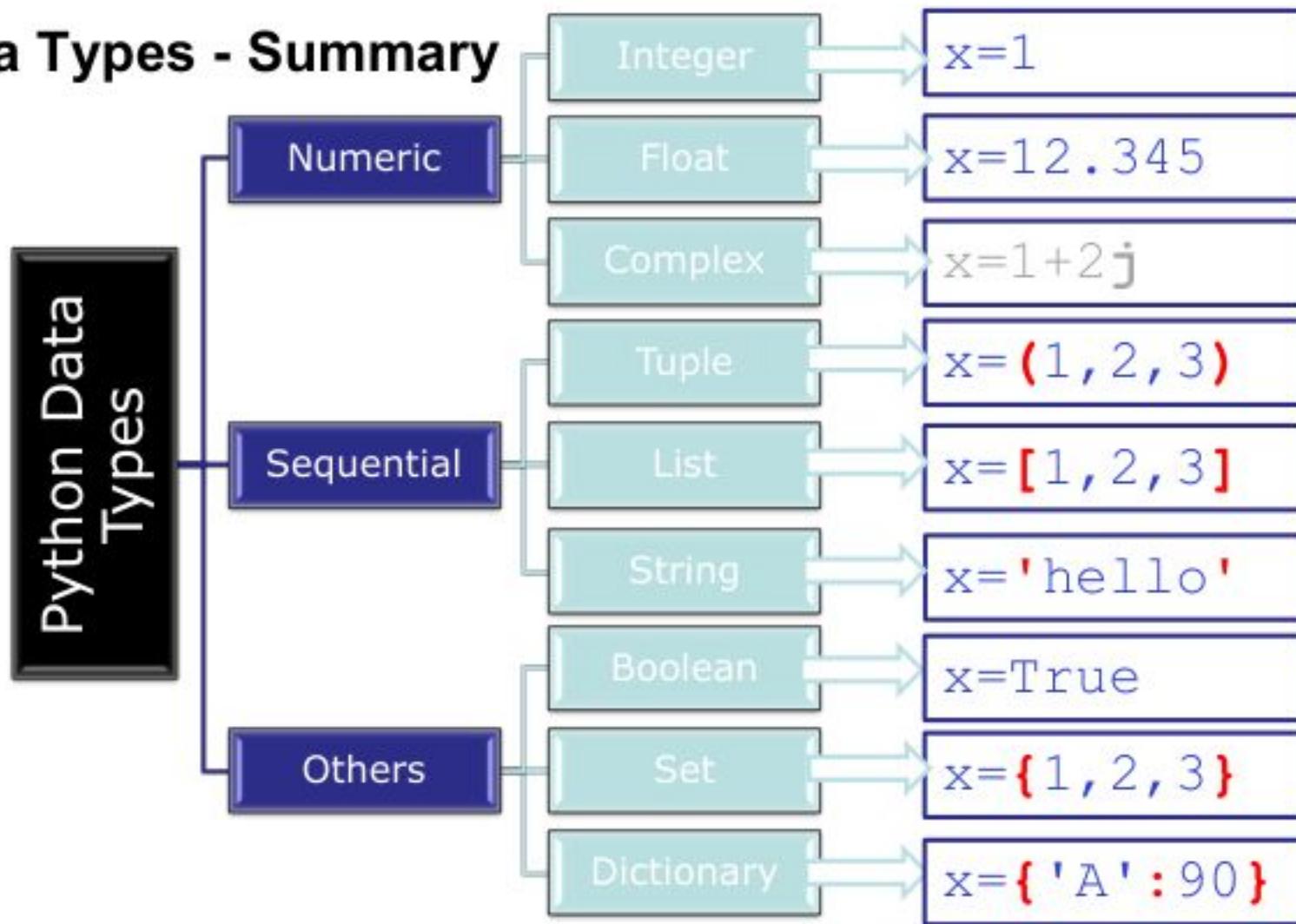
	A	B	C
1	Employee ID	Full Name	Job Title
2	E02002	Kai Le	Controls Engineer
3	E02003	Robert Patel	Analyst
4	E02004	Cameron Lo	Network Administrator

```
employee_ids = ['E02002', 'E02003']
full_names = ['Kai Le', 'Robert Patel']
job_titles = ['Controls Engineer', 'Analyst']

print( list(zip(employee_ids, full_names)) )
print( dict(zip(employee_ids, full_names)) )
print( dict(zip(employee_ids, job_titles)) )

[('E02002', 'Kai Le'), ('E02003', 'Robert Patel')]
{'E02002': 'Kai Le', 'E02003': 'Robert Patel'}
{'E02002': 'Controls Engineer', 'E02003': 'Analyst'}
```

## Data Types - Summary



# Business Process Automation with Python

## Conditional Statements

- If-else
  - Python tracks the scope using **indentation** (spaces)

### VBA

```
Dim speed As Integer  
speed = 70  
  
If speed > 50 Then  
    Debug.Print ("Exceeded limit")  
Else  
    Debug.Print ("OK")  
End If
```

### Python

```
speed = 70  
  
if speed > 50:  
    print('Exceeded limit')  
else:  
    print('OK')
```



# Business Process Automation with Python

## Conditional Statements

- If-else
  - Value assignment can be conditional too!

### Multiline

```
speed = 70

if speed > 50:
    print('Exceeded limit')
else:
    print('OK')
```

### 1 line

```
speed = 70

print ('Exceeded limit' if speed > 50 else 'OK')
```

## Conditional Statements

- If-elif-else
  - elif → “Else If”

```
mark = 89
grade = None

if mark >= 90:
    grade = 'A'
elif mark >= 80:
    grade = 'B'
elif mark >= 70:
    grade = 'C'
else:
    grade = 'F'

print(grade)
```

D. B

## Conditional Statements

- If-elif-else
  - elif → “Else If”
  - May be broken down to if-else

```
mark = 89
grade = None

if mark >= 90:
    grade = 'A'
elif mark >= 80:
    grade = 'B'
elif mark >= 70:
    grade = 'C'
else:
    grade = 'F'

print(grade)
```

D B

```
mark = 89
grade = None

grade = 'A' if mark >= 90 else ('B' if mark >= 80 else ('C' if mark >= 70 else 'F'))

print(grade)
```

D B

# Business Process Automation with Python

## Conditional Statements

- If
  - “else” is optional

```
▶ salary = 10000
    salary_growth = 1.2
    years_of_service = 3

    # No increment for first year
    if years_of_service > 1:
        salary = salary * salary_growth
    print(salary)

□ 12000.0
```

# Business Process Automation with Python

## Conditional Statements

- If-elif-else
  - Unlike VBA, scope is NOT defined with “If → End If”
  - Python tracks the scope using **indentation** (spaces)

### VBA

```
Dim speed As Integer  
speed = 70  
  
If speed > 50 Then  
    Debug.Print ("Exceeded limit")  
Else  
    Debug.Print ("Speed is okay")  
End If
```

### Python

```
speed = 70  
  
if speed > 50:  
    print('Exceeded limit')  
else:  
    print('Speed is okay')
```

# Business Process Automation with Python

## Loops – Using “for”

- Iterating through a range

	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

### VBA

```
For i = 1 To 10  
    Debug.Print (i)  
Next i
```

### Python

```
for i in range(1, 11):  
    print(i)
```

## Loops – Using “for”

- Different ways to specify ranges



	A	B	C
1		1	
2		2	
3		3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	

```
[11] for i in range(1, 11):
    print(i) # 1, 2, ..., 10
1
2
3
4
5
6
7
8
9
10
```



```
▶ for i in range(10):
    print(i + 1) # i = 0, 1, ..., 9
1
2
3
4
5
6
7
8
9
10
```

## Loops – Using “for”

- Under the hood
  - Iterating through a sequence object

```
[21] print(range(10))  
range(0, 10)
```



	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
▶ list(range(10)) # In Python2, range() returns a list directly  
⇒ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



# Business Process Automation with Python

## Loops – Using “for”

- Be careful...

```
▶ item_prices = [200, 100]
    total = 0
    # Case 1: Apply $50 discount once
    for price in item_prices:
        total = total + price
        if total >= 200:
            total = total - 50
    print(total)
```

✉ 250

```
▶ item_prices = [200, 100]
    total = 0
    # Case 2: Apply $50 discount twice (accidentally..)
    for price in item_prices:
        total = total + price
        if total >= 200:
            total = total - 50
    print(total)
```

✉ 200

# Business Process Automation with Python

## Loops – Using “for”

- Looping through a list

```
▶ school_list = ['hku', 'cuhk', 'hkust']

for school in school_list:
    print(f'{school} is a good school in Hong Kong')

▶ hku is a good school in Hong Kong
    cuhk is a good school in Hong Kong
        hkust is a good school in Hong Kong
```

# Business Process Automation with Python

## Loops – Using “for”

- Looping through a dictionary

```
➊ school_rank_dict = {'hku': 21, 'cuhk': 38, 'hkust': 40}

➋ for school, rank in school_rank_dict.items():
    | print(f'{school} is ranked {rank} in the world!')

➌ hku is ranked 21 in the world!
    cuhk is ranked 38 in the world!
        hkust is ranked 40 in the world!
```

## Loops – Using “for”

- Looping through a string
- Q: Why use “elif” instead of “else”?

```
▶ # looping through a string
    uppercase_letter_count = 0
    lowercase_letter_count = 0

    for letter in 'See you in Hong Kong':
        if letter.isupper():
            uppercase_letter_count = uppercase_letter_count + 1
        elif letter.islower():
            lowercase_letter_count = lowercase_letter_count + 1

    print(f'Uppercase letters: {uppercase_letter_count}')
    print(f'Lowercase letters: {lowercase_letter_count}')

□ Uppercase letters: 3
Lowercase letters: 13
```

# Business Process Automation with Python

## Loops – Using “for”

- List comprehension  
→ compressing a loop in 1 line

```
▶ print(f'8 % 2 = {8%2}')
print(f'9 % 2 = {9%2}')

⇒ 8 % 2 = 0
9 % 2 = 1
```

```
▶ # Traditional way
even_number_list = []
for i in range(11):
    # Use the modulus operator to check for even number (Can be divided by 2)
    if i % 2 == 0:
        even_number_list.append(i)
print(even_number_list)

⇒ [0, 2, 4, 6, 8, 10]
```

```
▶ # List comprehension
even_number_list = [i for i in range(11) if i % 2 == 0]
print(even_number_list)

⇒ [0, 2, 4, 6, 8, 10]
```

# Business Process Automation with Python

## Loops – Using “for”

- List comprehension + Filtering

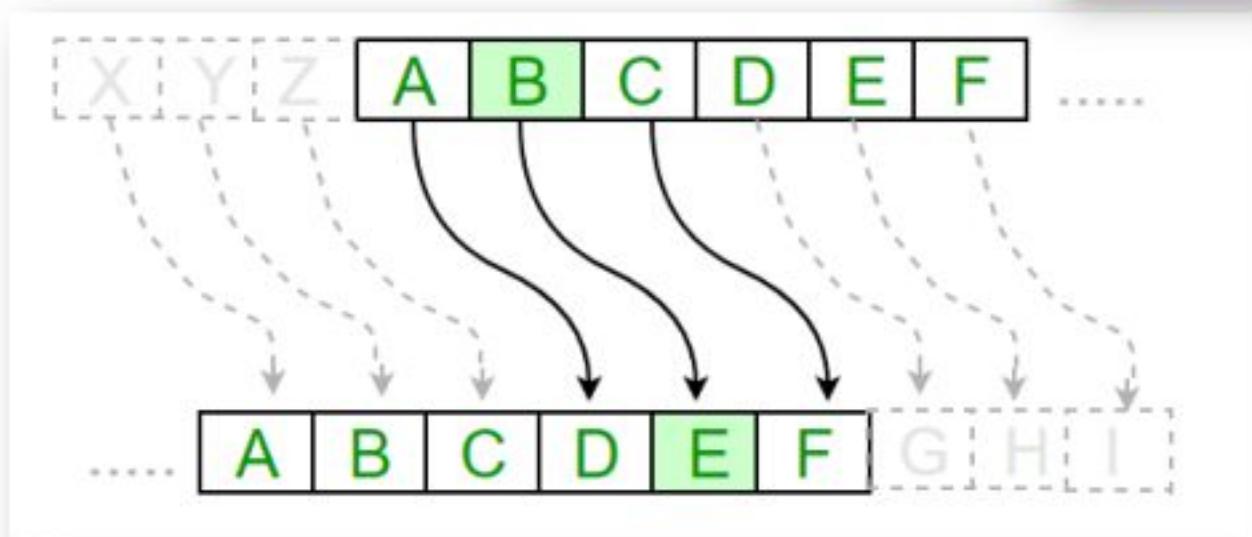
```
▶ print(bool(''))  
print(bool(None))
```

```
▶ item_list = ['123', '456', '']  
  
▶ print([item for item in item_list])  
print([item for item in item_list if bool(item)])  
print([item for item in item_list if item])  
  
◀ ['123', '456', '']  
['123', '456']  
['123', '456']
```

## Practice 3 – Caesar Cipher

- **Encryption: Shift each letter by 3**

- Reference: <https://cryptii.com/pipes/caesar-cipher>



- **Starting notebook:**

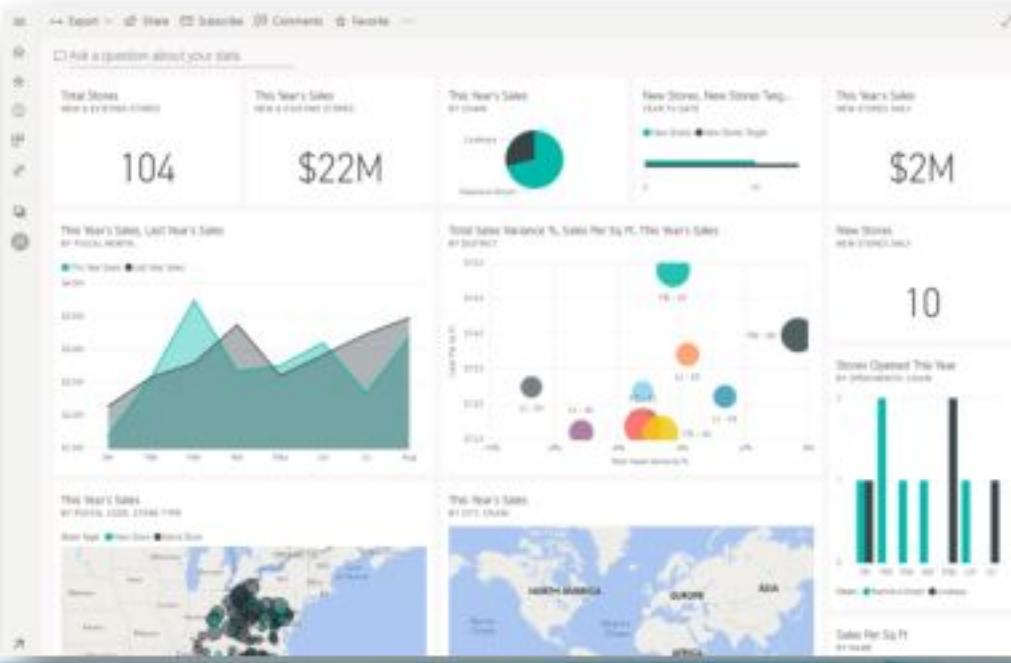
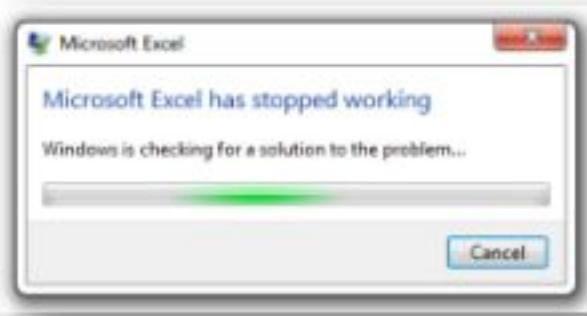
[https://github.com/innoviai/ipa\\_courses/blob/main/Lecture%202/practice2\\_word\\_count\\_202409.ipynb](https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb)

# Business Process Automation with Python

## File Input / Output

- Motivation

- Storage → Saving progress
- Interaction with other systems



## File Input / Output

- **Creating a file object**

- `open(file_name, file_mode)`
- **file\_mode**
  - **r** → read (read an existing file)
  - **w** → write (create a new file)
  - **a** → append (add to an existing file)
  - **b** → binary mode
    - e.g., Chinese characters / PDF files
    - adds to another mode (e.g., 'rb' instead 'r')

## File Input / Output

- **Writing a text file**

- `open()` → with 'w' mode
- `write()` → insert a string
- `close()` → save the file (as a last step)

```
▶ # Create a handler
    output_file_stream = open('test.txt', 'w')

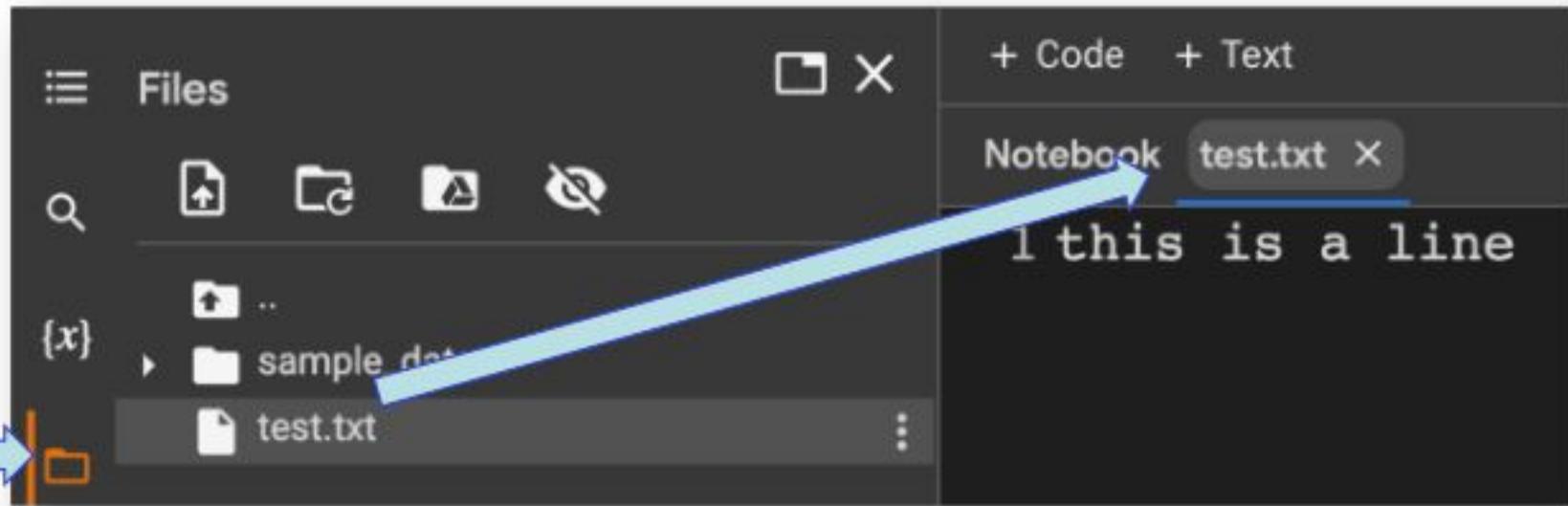
    # Write a string into it
    output_file_stream.write('this is a line')

    # Save the file
    output_file_stream.close()
```

# Business Process Automation with Python

## File Input / Output

- Writing a text file
  - Viewing the result



# Business Process Automation with Python

## File Input / Output

- Writing a text file
  - Multiple lines – the problem

```
▶ # Create a handler
    output_file_stream = open('test.txt', 'w')

    # Write a string
    output_file_stream.write('this is a line')
    # Write another one
    output_file_stream.write('this is another line')

    # Save the file
    output_file_stream.close()
```

Notebook test.txt X



1 this is a linethis is another line

## File Input / Output

- **Writing a text file**

- Multiple lines – the solution
- → The next-line character “\n”

```
Notebook test.txt X
1 this is a line
2 this is another line
```

```
❶ # Create a handler
output_file_stream = open('test.txt', 'w')

# Write a string
output_file_stream.write('this is a line\n') ←
# Write another one
output_file_stream.write('this is another line')

# Save the file
output_file_stream.close()
```

## File Input / Output

- **The “With” statement**

- Better readability (in terms of scoping)
- Automatically calling close() at the end

```
▶ # Create a handler
    with open('test.txt', 'w') as output_file_stream:
        # Write a string
        output_file_stream.write('this is a line')
```

## File Input / Output

- **Reading a text file**

- `open()` with 'r' mode
- `read()` → extract all the content from the file

```
➊ file_text = None
    # Create a handler
    with open('test.txt', 'r') as input_file_stream:
        # Read the file
        file_text = input_file_stream.read()
file_text

➋ 'this is a line\nthis is another line'
```

## File Input / Output

- **Reading a text file**

- Handling separators
  - `splitlines()` → useful against line separators '\n'
  - `split()` → flexible, can handle commas in CSV files

```
▶ print( file_text.split('\n') )  
print( file_text.splitlines() )  
  
↳ ['this is a line', 'this is another line']  
['this is a line', 'this is another line']
```

# Business Process Automation with Python

## Practice 4 – File Processing with DATA.GOV.HK data

- Read the CSV file
- Loop through the data and do some counting



A	B	C	D	E
Account	Expenditure Group	Subhead	Item	Actual(\$)
1 RECURRENT	PERSONNEL EMOLUMENTS	SALARIES	SALARIES	151,760,978
2 RECURRENT	PERSONNEL EMOLUMENTS	ALLOWANCES	OVERTIME	507,492
4 RECURRENT	PERSONNEL EMOLUMENTS	ALLOWANCES	ACTING ALLOWANCES	5,636,585
5 RECURRENT	PERSONNEL RELATED EXPENSES	MANDATORY PROVIDENT FUND CONTRIBUTION	CONTRIBUTION FOR RECRUITS UNDER THE NEW TERMS BEFORE CONFIRMATION	265,855
6 RECURRENT	PERSONNEL RELATED EXPENSES	CIVIL SERVICE PROVIDENT FUND CONTRIBUTION	MANDATORY CONTRIBUTION UNDER THE MANDATORY PROVIDENT FUND SCHEMES ORDINANCE	1,468,649
7 RECURRENT	PERSONNEL RELATED EXPENSES	CIVIL SERVICE PROVIDENT FUND CONTRIBUTION	GOVERNMENT VOLUNTARY CONTRIBUTION	9,534,636
8 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	CLOTHES AND UNIFORMS	21,940
9 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	CLEANING MATERIALS	107,532
10 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	OFFICE STATIONERY AND MATERIALS	672,661
11 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	GENERAL PUBLICATIONS, PERIODICALS AND JOURNALS	123,429
12 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	OFFICE FURNITURE AND EQUIPMENT (INCLUDING RENTAL)	445,709
13 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	PAPERS	53,221
14 RECURRENT	DEPARTMENTAL EXPENSES	STORES AND EQUIPMENT	INFORMATION AND COMMUNICATIONS TECHNOLOGY	3,299,560
15 RECURRENT	DEPARTMENTAL EXPENSES	LIGHT AND POWER	ELECTRICITY	422,305

- Starting notebook:

[https://github.com/innoviai/ipa\\_courses/blob/main/Lecture%202/practice4\\_file\\_IO\\_gov\\_hk\\_202409.ipynb](https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice4_file_IO_gov_hk_202409.ipynb)

# Business Process Automation with Python

## Notebook Summary

[https://github.com/innovai/iqa\\_courses/blob/main/Lecture%202/practice2\\_word\\_count\\_202409.ipynb](https://github.com/innovai/iqa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb)

The screenshot shows a Jupyter Notebook interface with the following structure:

- Table of contents**:
  - Python Introduction
    - 0) Displaying values
      - The last line in a code block is evaluated and displayed
      - You can also use print() method to display the values
    - 1) Primitive Data types
      - Numeric - Integer
      - Numeric - Float (with decimal points)
      - Numeric - Complex (For scientific calculations)

**Common functions**:
  - Text - String
    - Boolean (True or False)
    - Simple conversion
    - Simple string formatting
  - 2) Sequential Data Types
    - Motivation - 1 variable to store many values
      - Tuples (multiple values)
      - List - You can change the values
      - String - as a sequential data type
      - Set - Unique values
    - 3) Dictionary - Key and value pairs lookup
      - If-else
    - 4) Control Flows
    - 5) File I/O

# Business Process Automation with Python

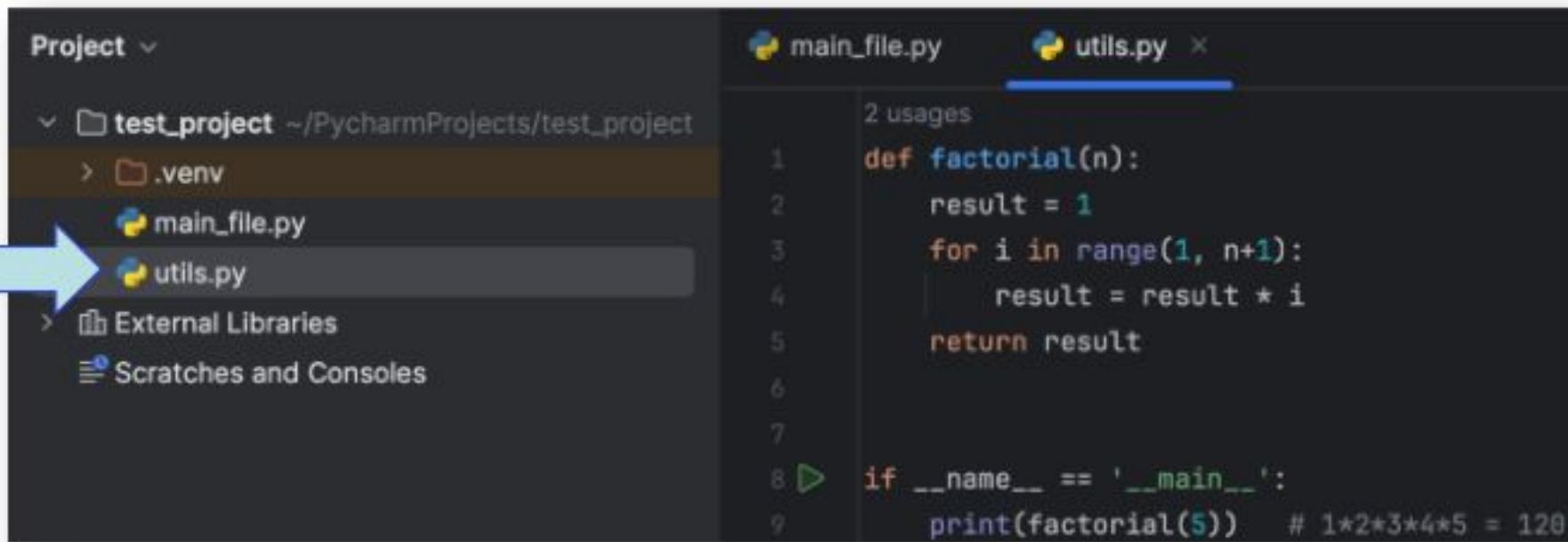
## Practice 4 - Review

- It takes so much effort to read CSV files
  - Q: But this should be a common task?
  - Any instant noodles for me to “add hot water & eat”?



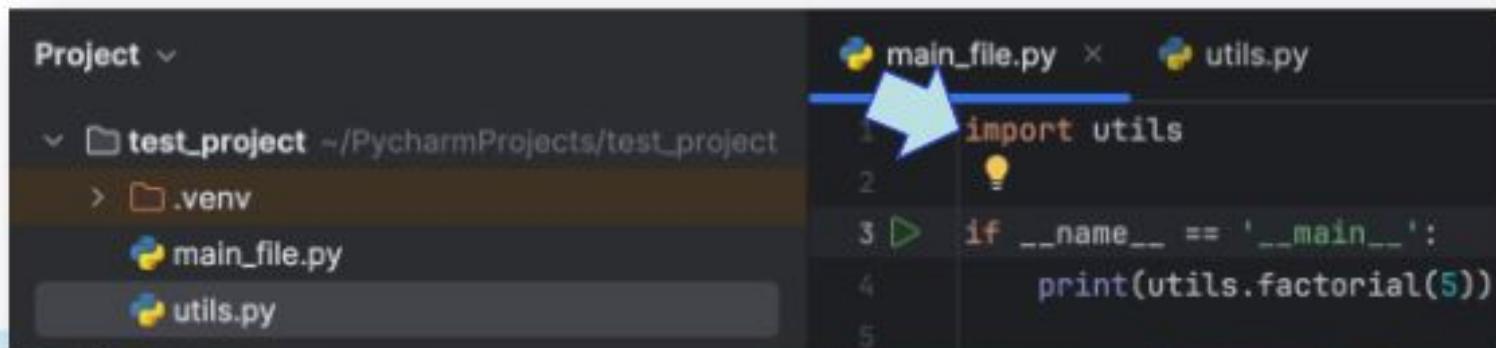
# Business Process Automation with Python

## Importing Modules



The screenshot shows the PyCharm interface. On the left, the Project tool window displays a project named 'test\_project' containing files '.venv', 'main\_file.py', and 'utils.py'. A large blue arrow points from the 'utils.py' file in the project tree towards the code editor. The code editor on the right shows the contents of 'main\_file.py':

```
1  def factorial(n):
2      result = 1
3      for i in range(1, n+1):
4          result = result * i
5      return result
6
7
8  if __name__ == '__main__':
9      print(factorial(5)) # 1*2*3*4*5 = 120
```



The screenshot shows the PyCharm interface with the same project structure. A blue arrow points from the 'utils.py' file in the project tree towards the code editor. The code editor now shows the modified 'main\_file.py' with an import statement:

```
1  import utils
2
3  if __name__ == '__main__':
4      print(utils.factorial(5))
```

# Business Process Automation with Python

## Python Architecture (Python runtime + Packages)

- Packages

- Built-in / published codes which can be imported to use
- e.g., **datetime** module → **date** class → **strftime** function

```
▶ import datetime  
  
dt = datetime.date(2024, 12, 25)  
print(dt.strftime('%Y-%m-%d'))  
print(dt.strftime('%d %B %Y (%A)'))
```

```
→ 2024-12-25  
25 December 2024 (Wednesday)
```



Python



Packages

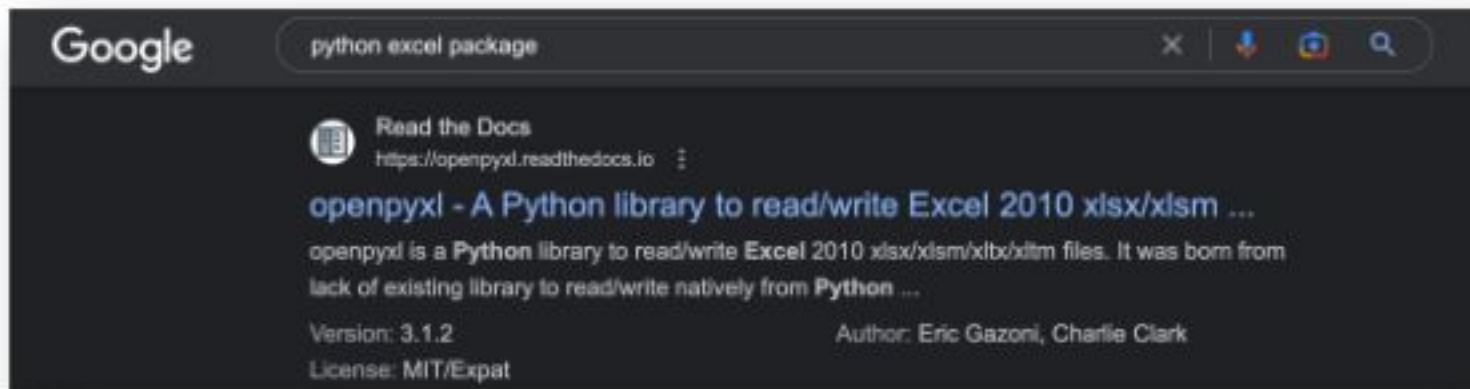


Easy programs!

# Business Process Automation with Python

## Packages - Installation

- Default installer – pip (Python 3.4 or later)
  - Search for “Python” + (something you need) + “Package”
  - Run “pip install xxx” in command prompt / terminal



## Installation

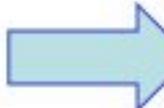
Install openpyxl using pip. It is advisable to do this in a Python virtualenv without system packages:

```
$ pip install openpyxl
```

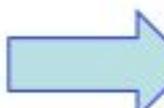
# Business Process Automation with Python

## Packages - Installation

- In Jupyter Notebooks
  - Add “!” → Install system-wide
  - Add “%” → Install within notebook environment



```
● !pip install pandas
C: Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.5.3)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```



```
● %pip install pandas
C: Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.5.3)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```



**HKUSPACE**  
香港大學專業進修學院  
HKU School of Professional and Continuing Education

# THANK YOU

