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

Business Process Automation with VBA and Python

Mr. Eddie Chow / 18 January 2025

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

Table Of Contents

Agenda

1. Introduction to Python
2. Python Data Types and Method
3. Introduction of Web Scraping
4. Overview of Web Scraping
5. Scraping Tools and Technologies
6. Scraping Environment Setup
7. Introduction to HTML and CSS
8. Beautiful Soup for Web Scraping
9. Data Collection - Reading API
10. Combining several CSV File
11. Exploratory Data Analysis(ETL) and Visualization

Python – Origin

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

Motivation – Soft Limits

- Protective measures not matching modern days

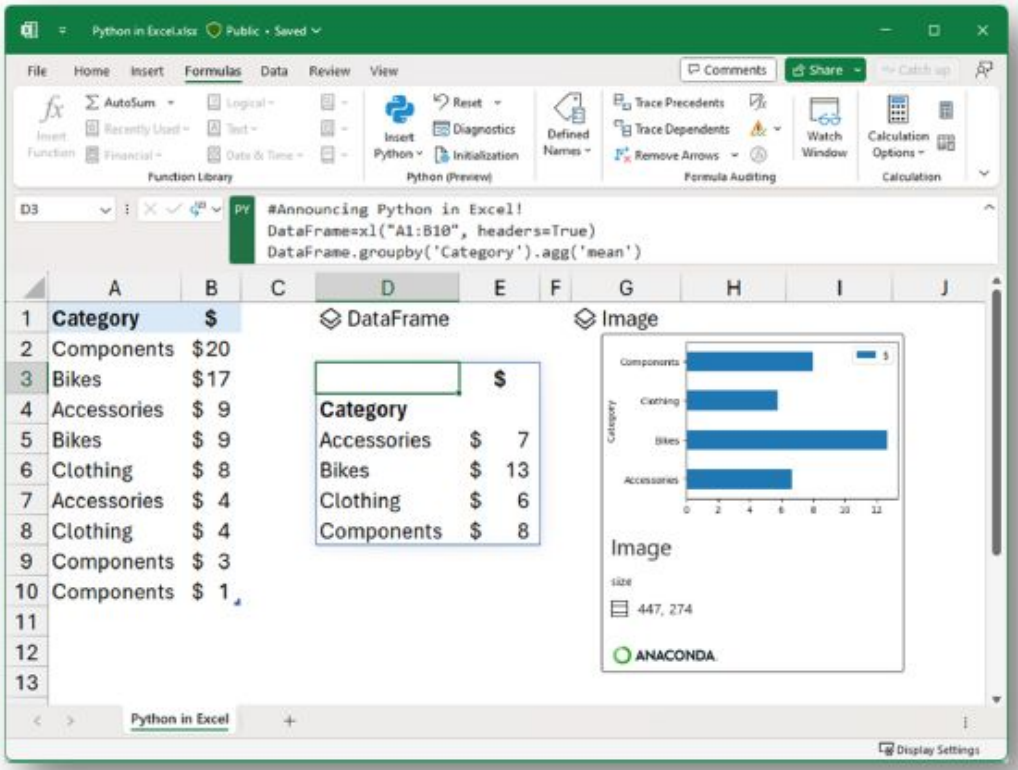
Feature	Maximum limit
Total number of rows and columns on a worksheet	1,048,576 rows by 16,384 columns
Column width	255 characters
Row height	409 points
Page breaks	1,026 horizontal and vertical

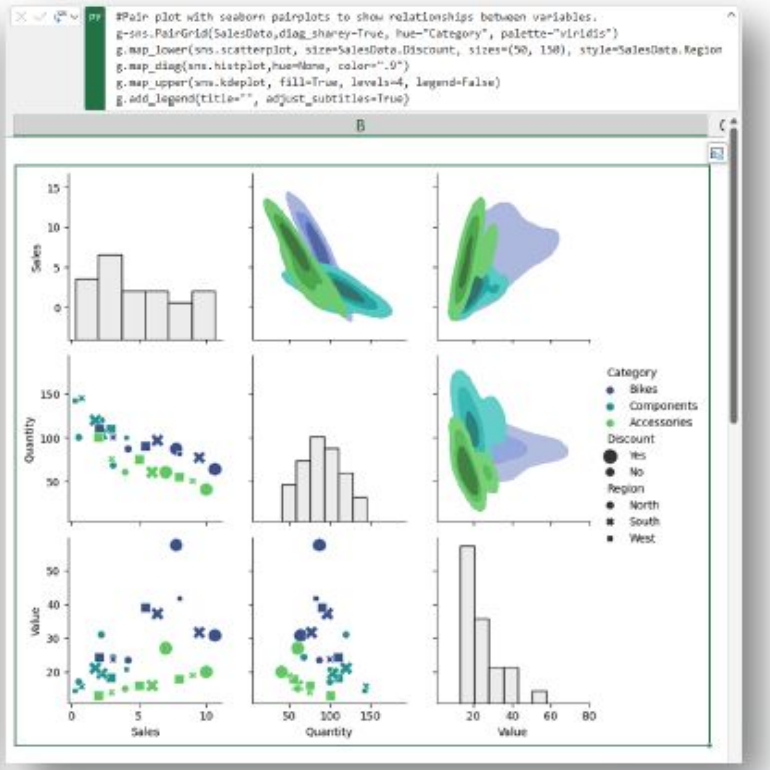
1. Microsoft. Excel Specifications and limits. https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3

Business Process Automation with Python

Motivation – Extensibility

- Python in Excel [1]





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

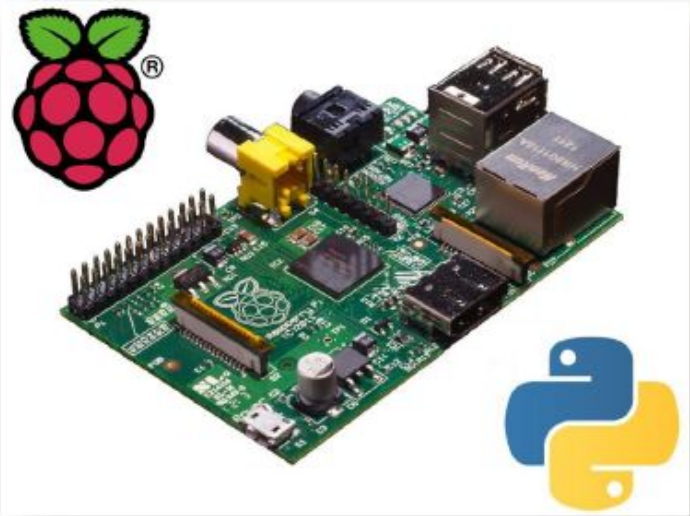
- How about outside of Microsoft Office?
- Program trading[1] / Web / AI / Business Intelligence

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=FyJrjnFUgPA

Business Process Automation with Python

Motivation – Jobs

- Python ➔ 1 of the top languages for work

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

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)

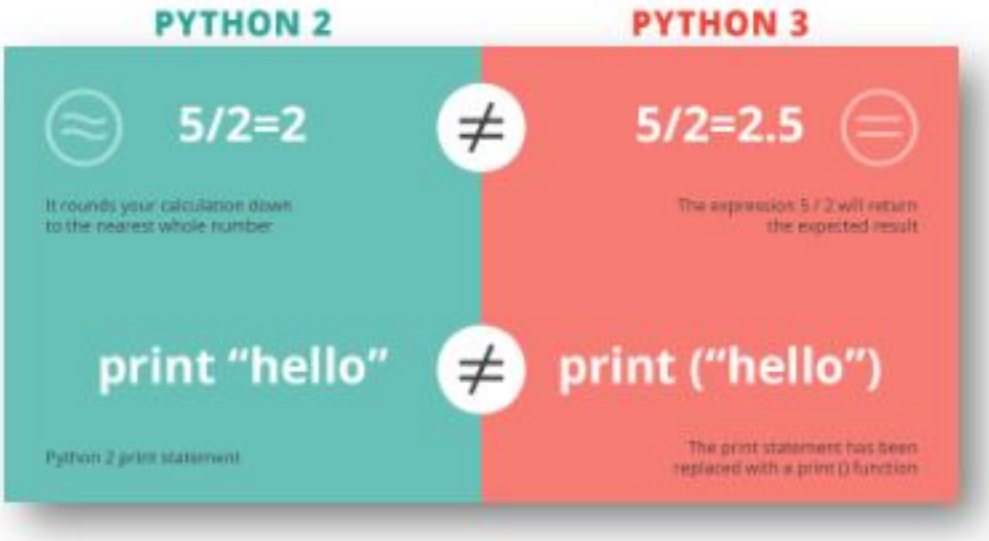
Analyst Skills
Focuses on the skills equity and credit analysts need using insights gained from hundreds of successful analysts.

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

Python runtime

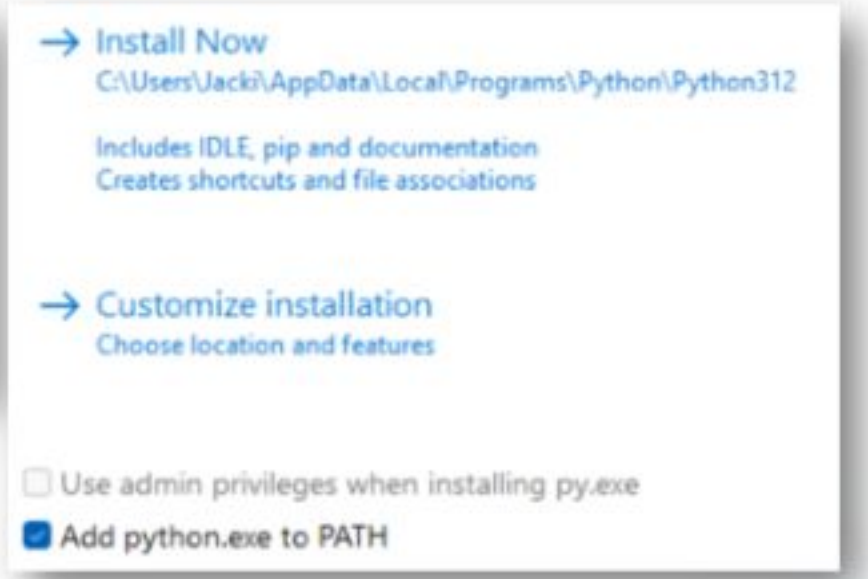
- The language and basic runtime environment
- Just like Apple iOS, it keeps updating (e.g., Python 3.12.2)
  - Python 2 was no longer supported from 2020[1]



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

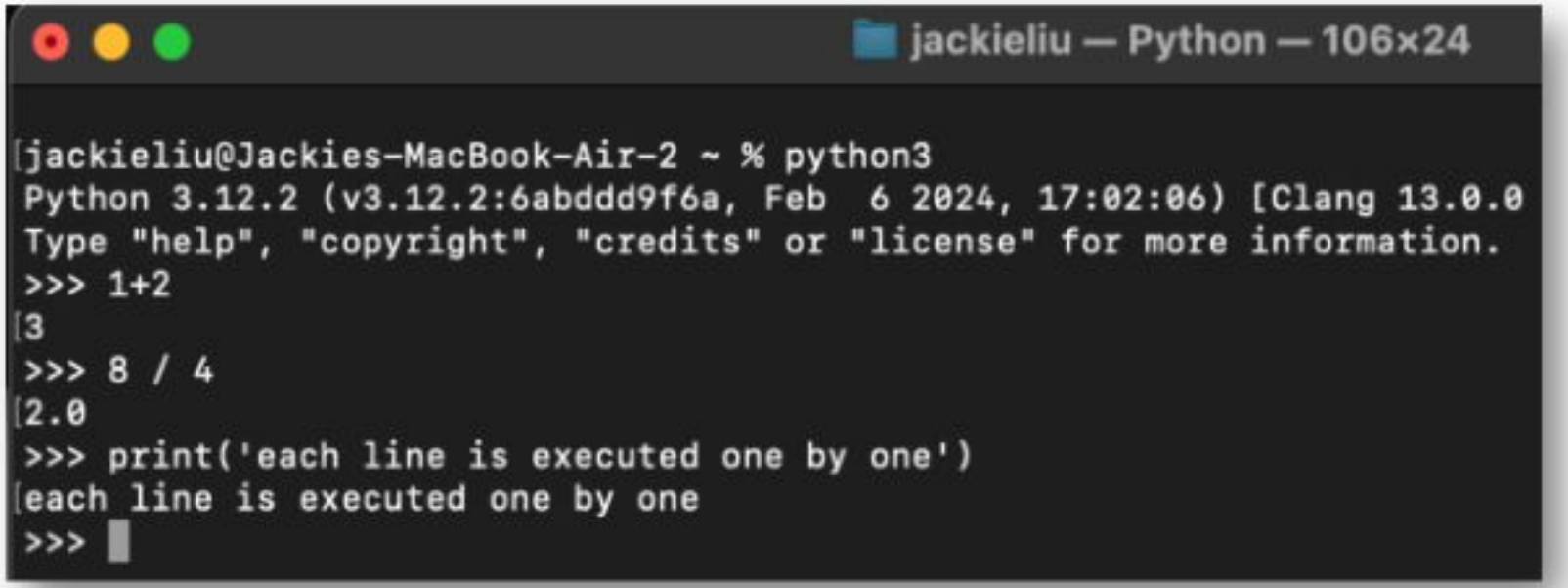
Installation – Python Runtime

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



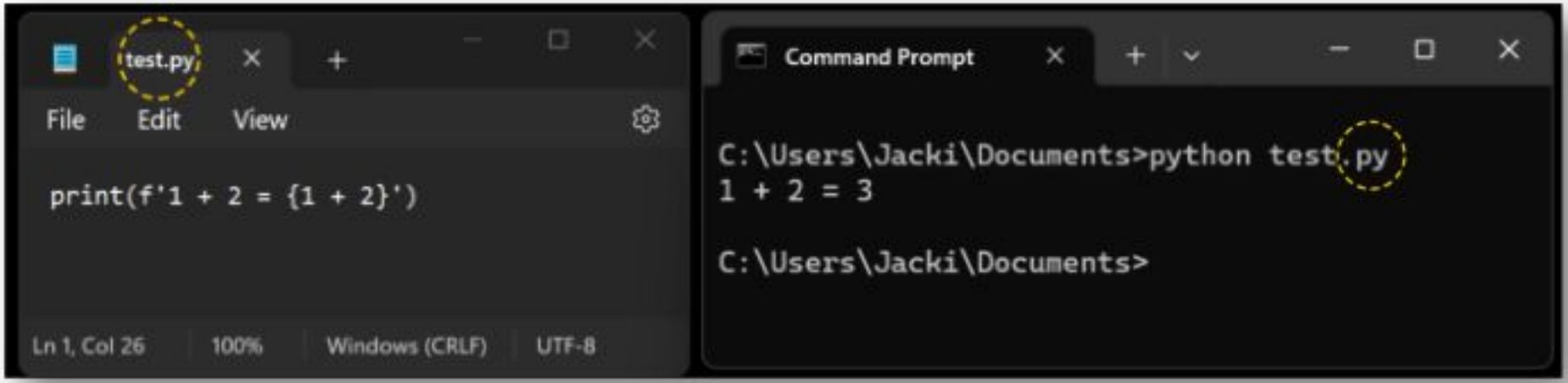
How to run Python – In the old days

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



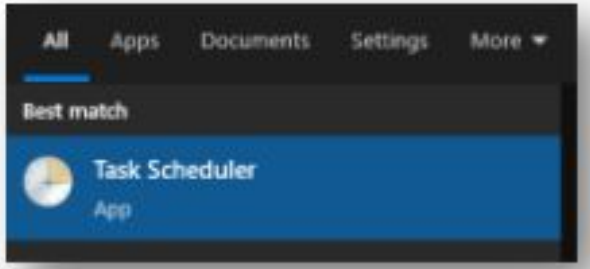
How to run Python – Running as a file

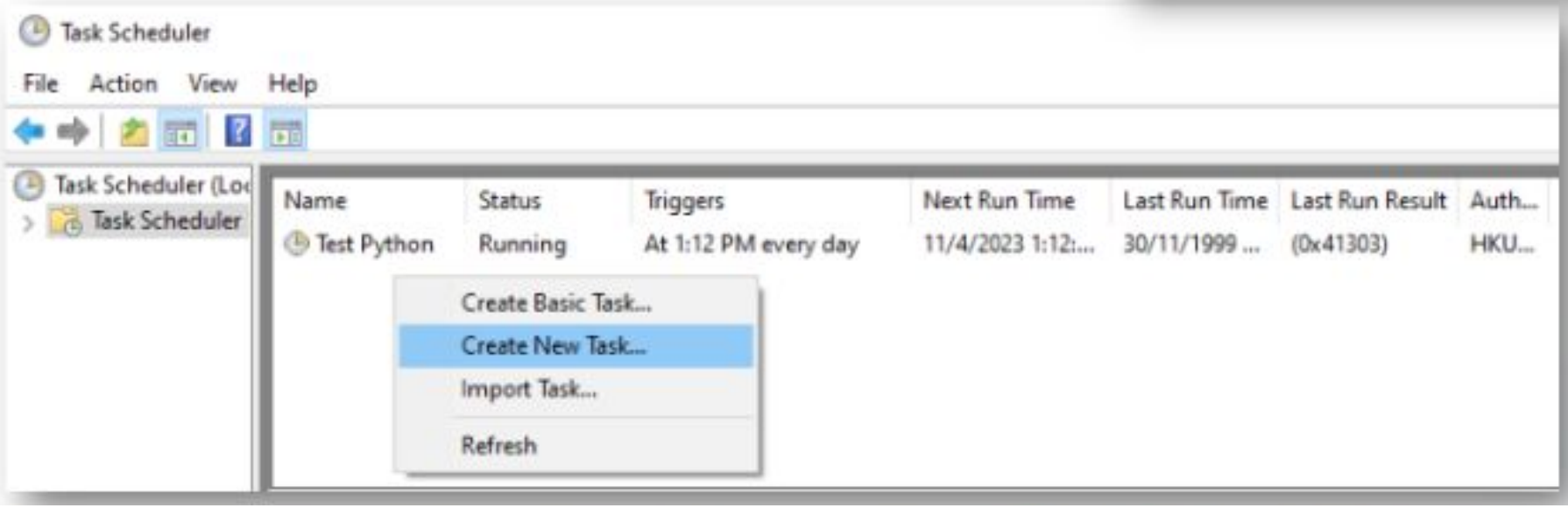
- Python interpreter + Text editor
  - Write codes in a text editor (e.g., Notepad, Vim, Sublime Text)
  - Save codes as a ".py" file
  - Run all the lines in one go



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



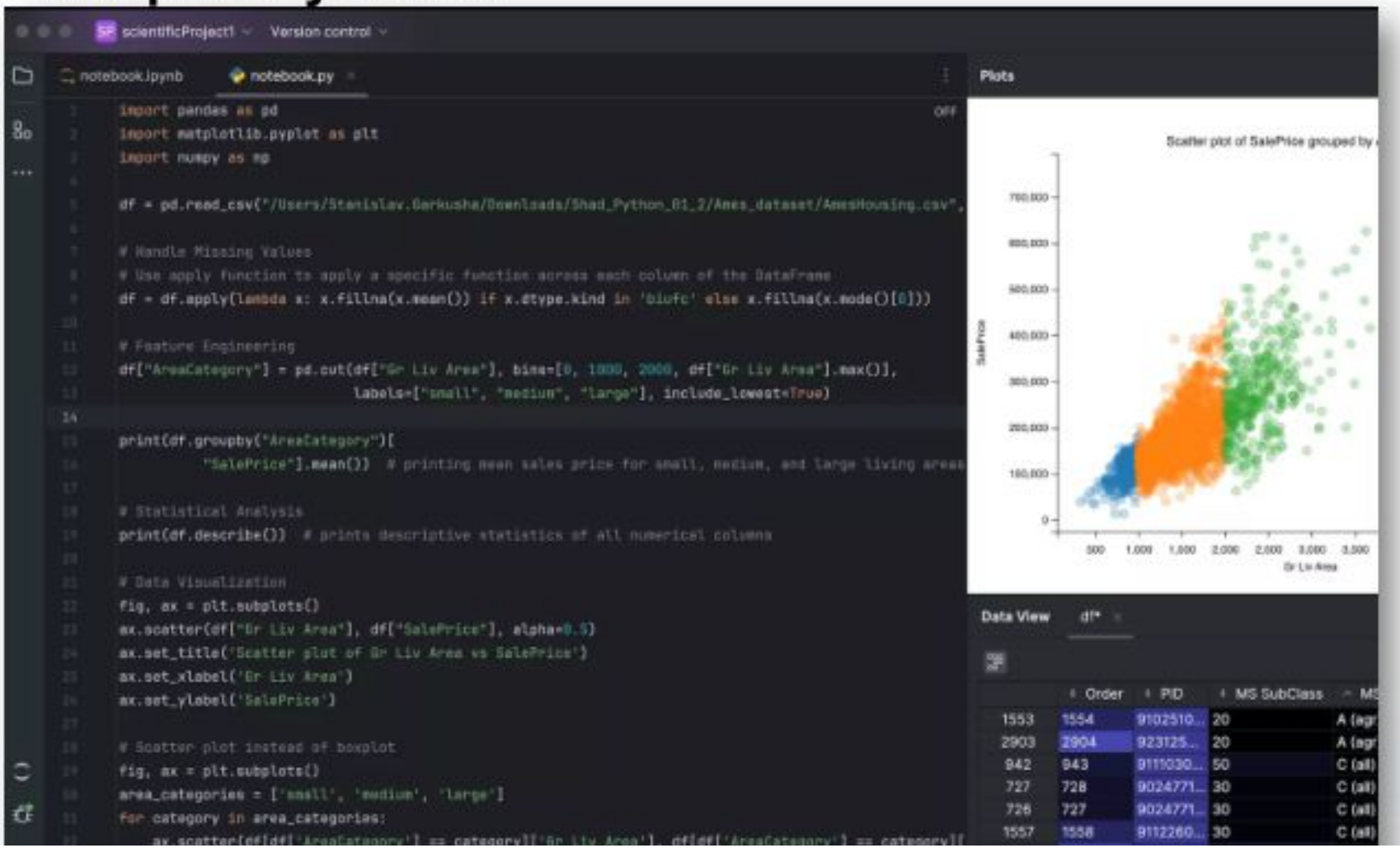


How to run Python – IDE

- 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

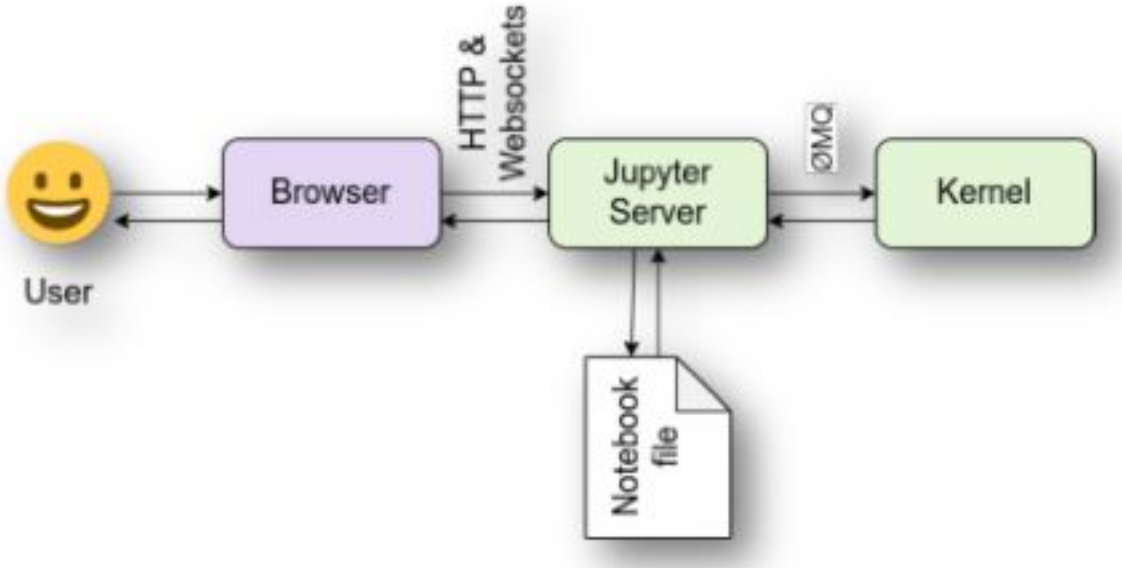
Business Process Automation with Python

Example - PyCharm



How to run Python – Jupyter Notebook

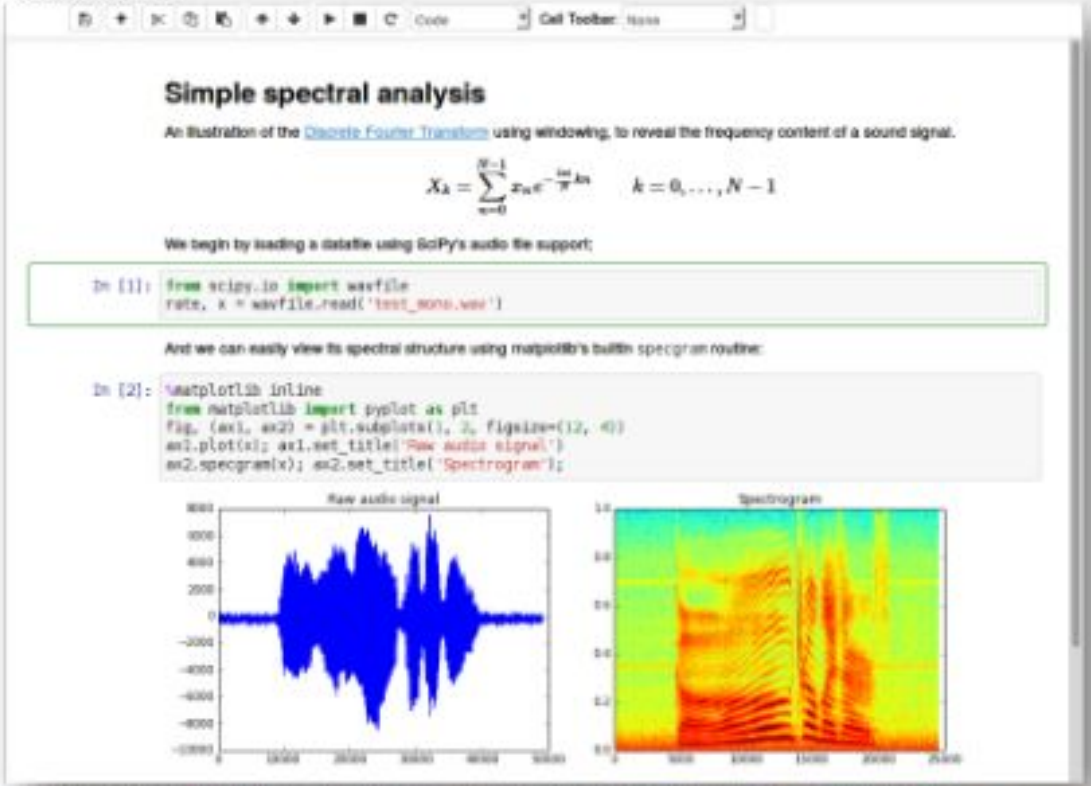
- Web-based interactive platform
  - Accessible: Python runtime on a web server



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



How to run Python – Jupyter Notebook

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

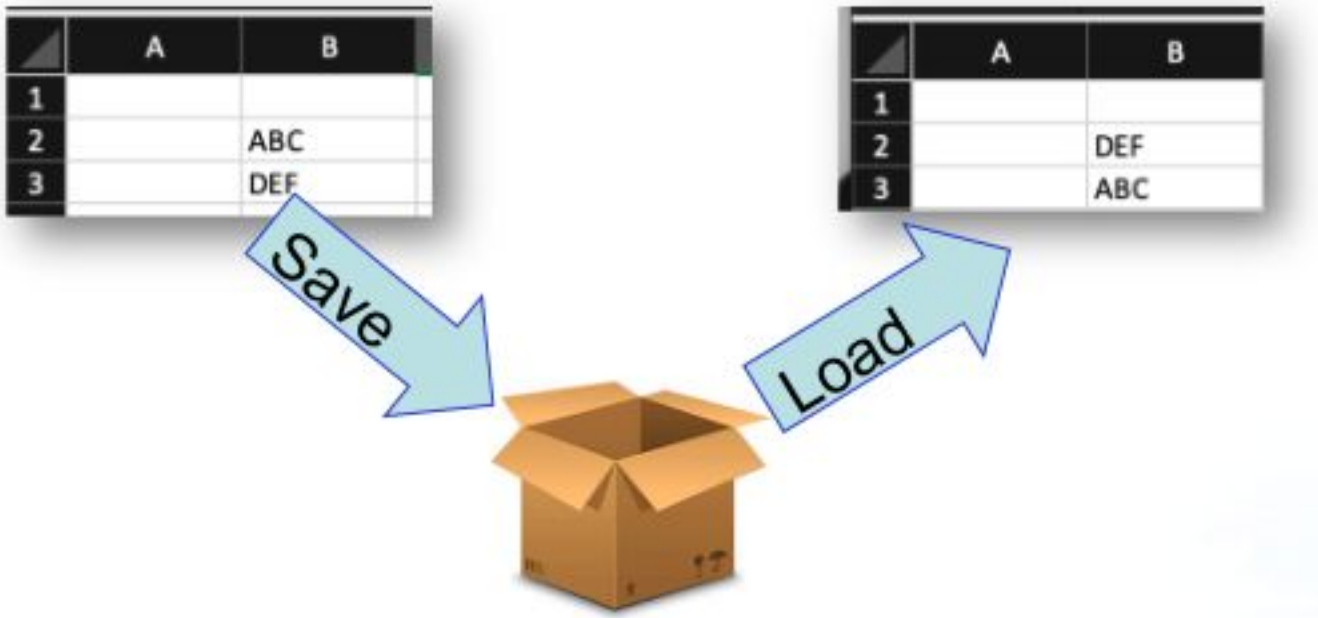
1. *Google Colab FAQ*. https://research.google.com/colaboratory/faq.html
2. *Google Colaboratory*. https://colab.research.google.com

How to run Python – Summary

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

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

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()

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

```
3
6
```

  - Last line in a block

```
1 + 2          # This is not shown
1 + 2 + 3      # This is shown
```

```
6
```

Primitive Data Types - Numeric

- Int

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

```
3
```

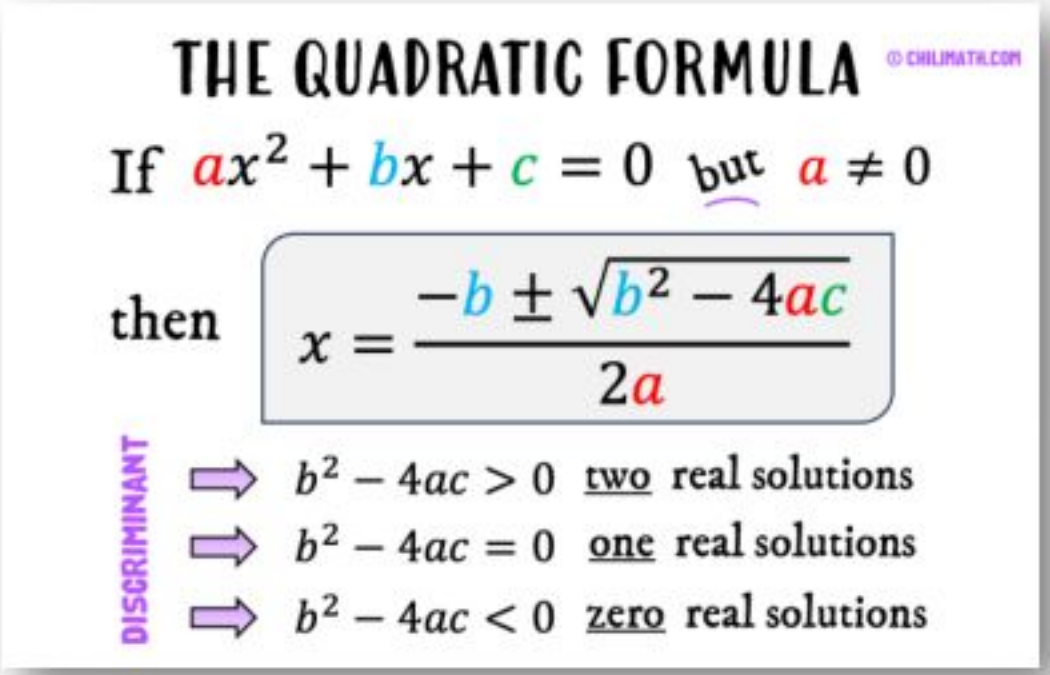
- Float

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

```
25.0
```

Primitive Data Types - Numeric

- Complex (For scientific calculations)




```
x = 3 + 4j
y = 2 + 2j
x - y
```

```
(1+2j)
```

$$i = \sqrt{-1}$$

Primitive Data Types - Numeric

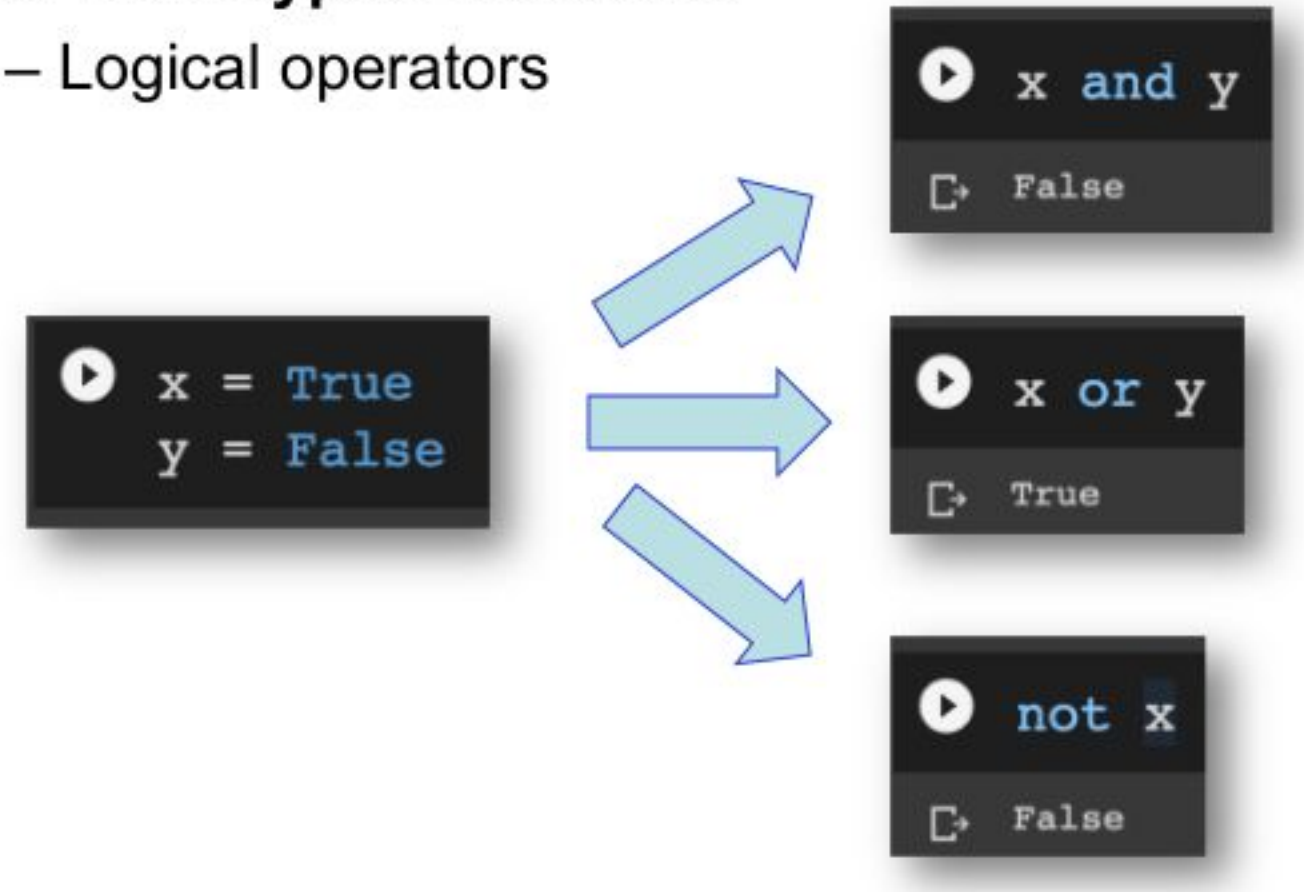
- Common Functions
  - round() / pow() / abs()

```
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
```

Primitive Data Types - Boolean

- Bool – Logical operators



Primitive Data Types - Boolean

- Bool – comparators
  - Generates True/ False

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

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
```

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
==========
```

1. 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%8d%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)

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

```
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!
```

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
```

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))
```

Practice 1 – Market Capitalization

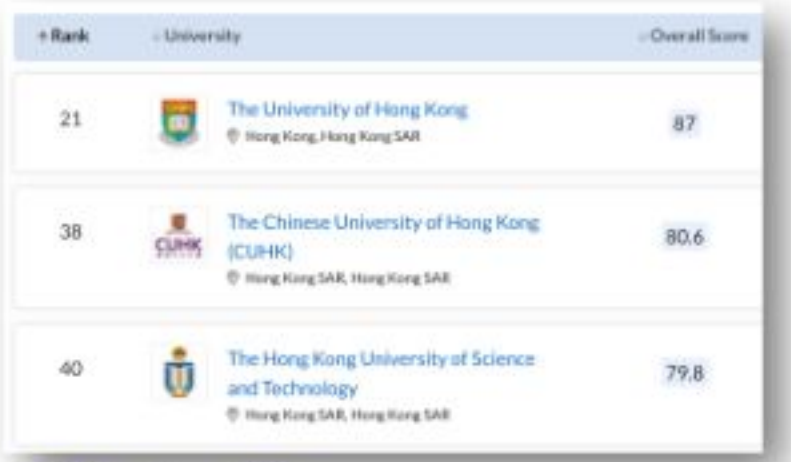
- Compute market capitalization of 2 companies
- Show the difference between the two
- Format the result up to 2 decimal places

- **Starting notebook:**

  https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice1_simple_calc_202409.ipynb

Sequential Data Types

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

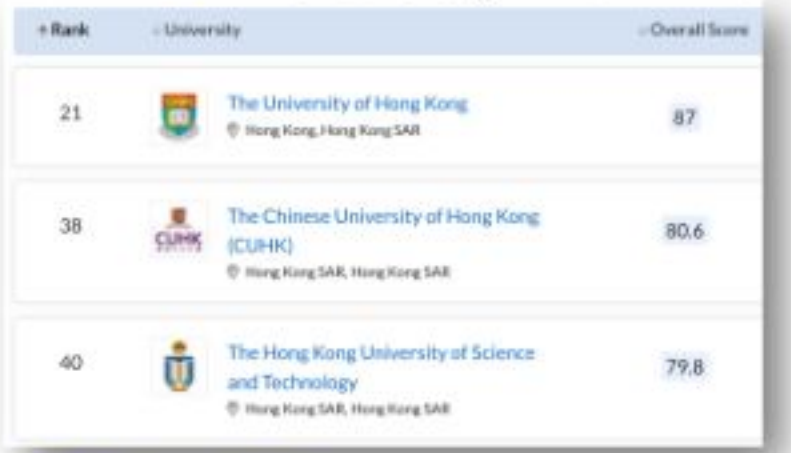


Rank	University	Overall Score
21	The University of Hong Kong Hong Kong, Hong Kong SAR	87
38	The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR	80.6
40	The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR	79.8

* QS Rankings 2023

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	87
38	The Chinese University of Hong Kong (CUHK)	80.6
40	The Hong Kong University of Science and Technology	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
```

Sequential Data Types

- **List**
  - Adding items

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

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

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

```
['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
```

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!']))
```

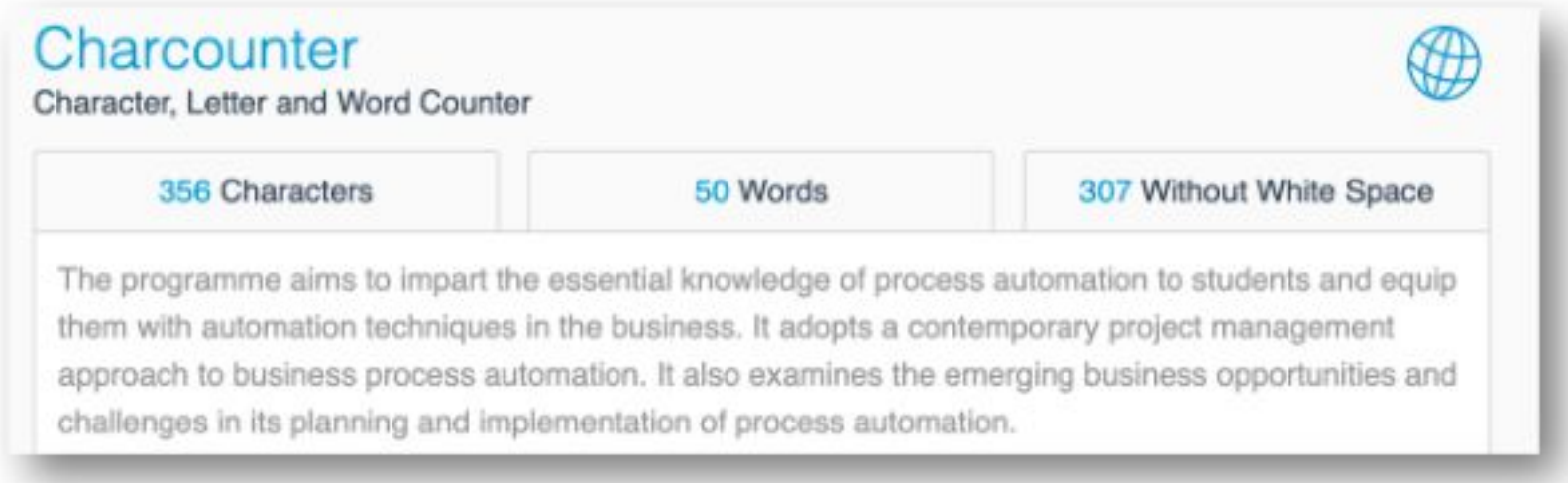
```
hello
Welcome,Jackie!
```

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

```
97
b
```

Practice 2 – Word Count

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



- Starting notebook:

https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

Data Type - Set

- **Motivation**
  - Problem ➔ Duplicate data are everywhere

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'>
```

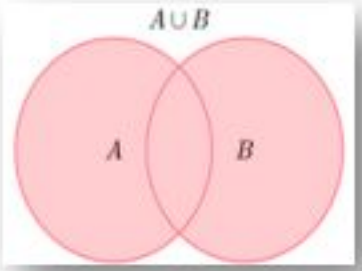
  - Can be converted between sequential data types

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

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

Data Type - Set

- **Set operations**
  - Union

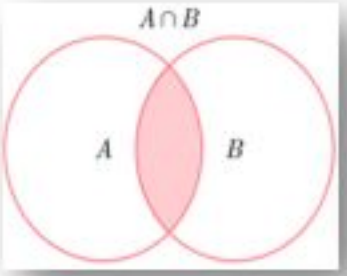



```
# Set operations
school_set_2022 = {'A', 'B', 'C'}
school_set_2023 = {     'B', 'C', 'D'}
```

```
# Union: items in any one of the sets
school_set_2022 | school_set_2023
```

```
{'A', 'B', 'C', 'D'}
```

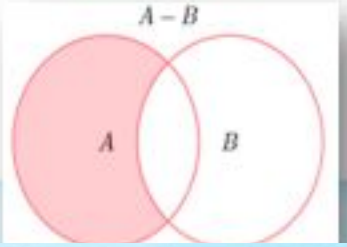
  - Intersection




```
# Intersection: items in both of the sets
school_set_2022 & school_set_2023
```

```
{'B', 'C'}
```

  - Difference




```
# Difference: Items in 1 set but not in the other
school_set_2023 - school_set_2022
```

```
{'D'}
```

Data Type - Dictionary

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

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

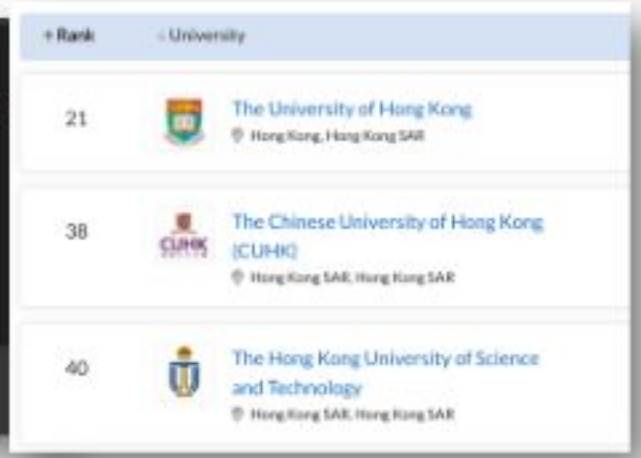
  - Looking up a keyword
    ➔ like finding words in **dictionaries**

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}
```



- 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}
```

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'values: {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'}
```

Data Type - Dictionary

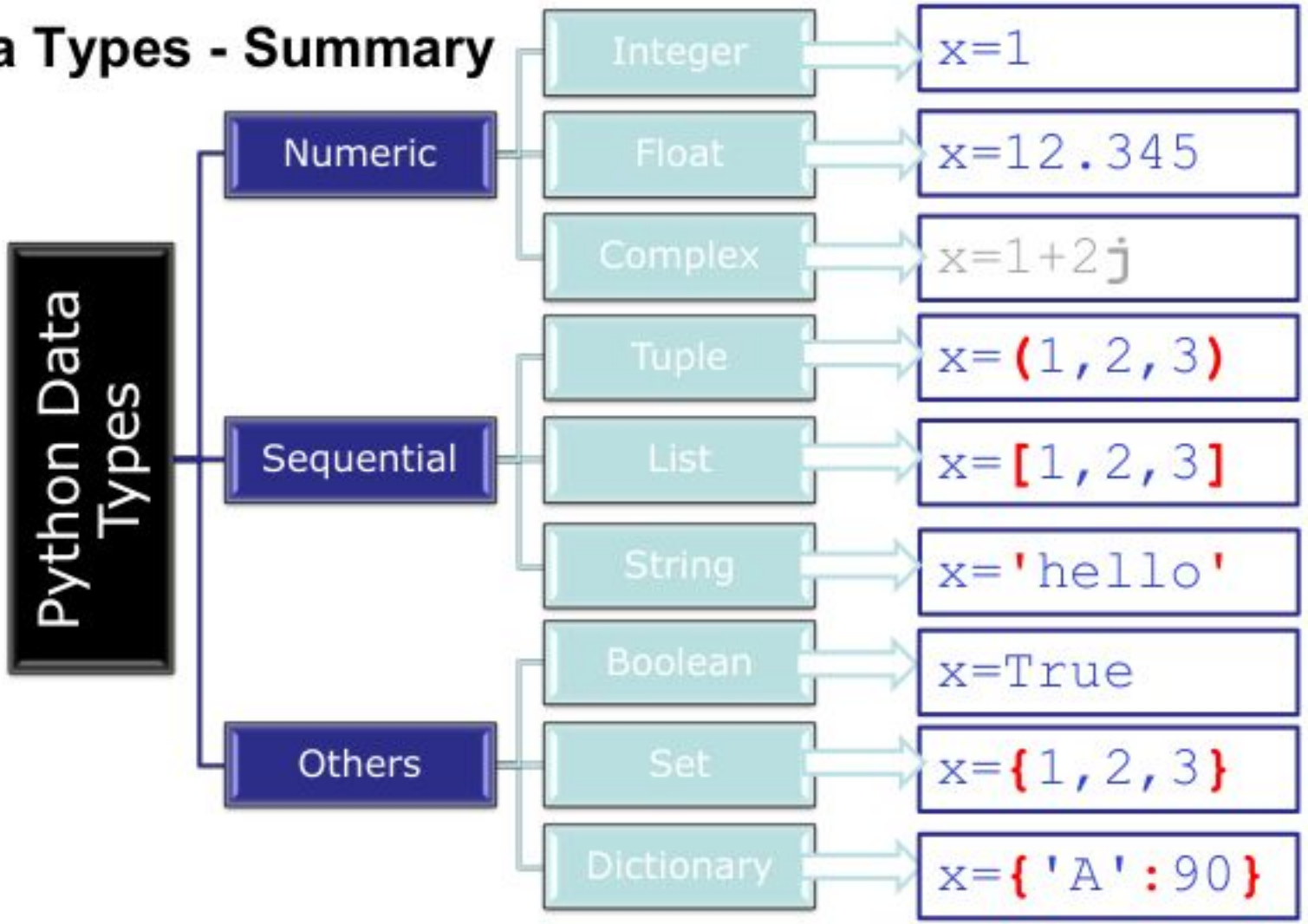
- **Conversion**
  - zip() can be used to create tuples

	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



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')
```

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)
```

```
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)
```

```
B
```

```
mark = 89
grade = None

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

print(grade)
```

```
B
```

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
```

Conditional Statements

- If-elif-else
  - Unlike VBA, scope is NOT defined with "If ➔ End If"
  - Python tracks the scope using indentation (spaces)

VBA

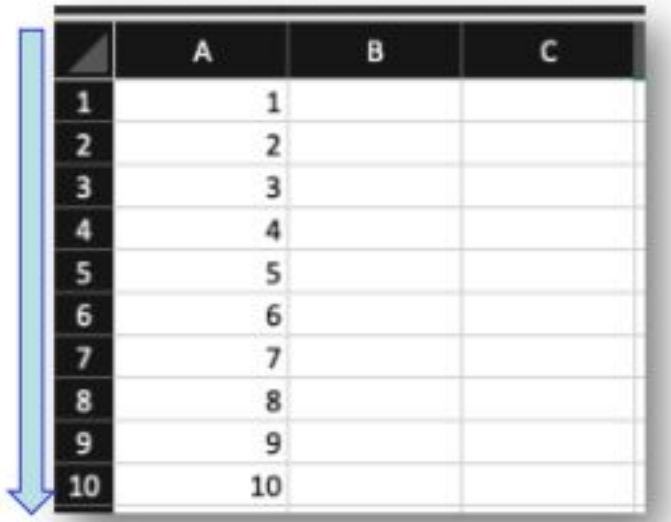
```
Dim speed As Integer
speed = 70

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')
```

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

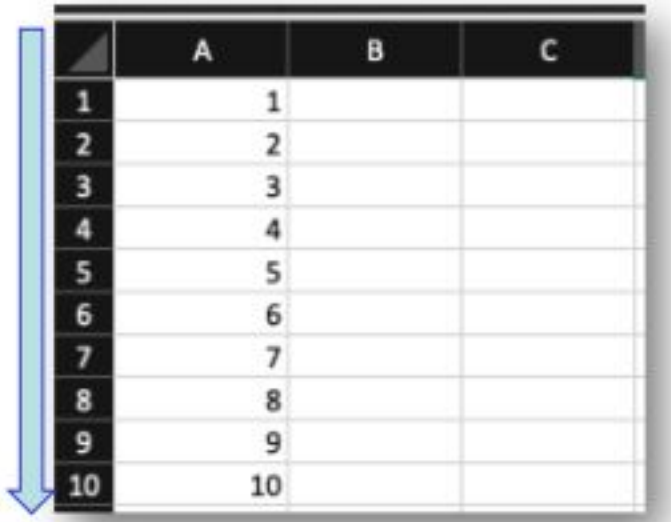
```
For i = 1 To 10
        Debug.Print (i)
Next i
```

Python

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

Loops – Using "for"

- Different ways to specify ranges



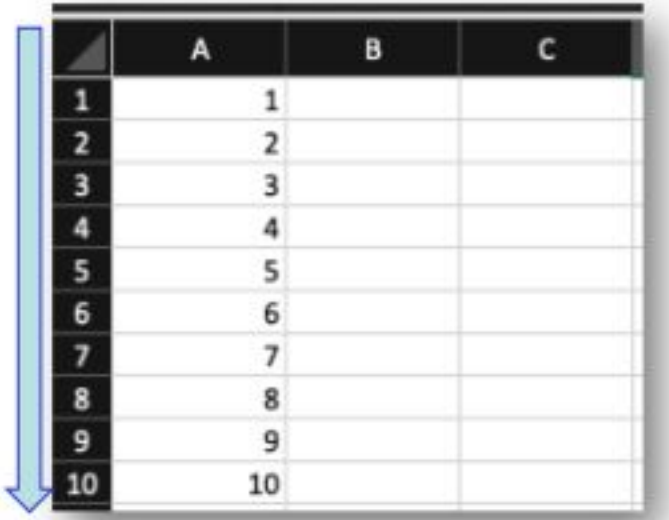
	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
[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)
```

```
list(range(10))  # In Python2, range() returns a list directly

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

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
```

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
```

Loops – Using "for"

- Looping through a dictionary

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

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

```
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
```

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 = list()
for i in range(11):
    # Use the modulus operator to check for even number (Can be divded by 2)
    if i % 2 == 0:
        even_number_list.append(i)
print(even_number_list)
```

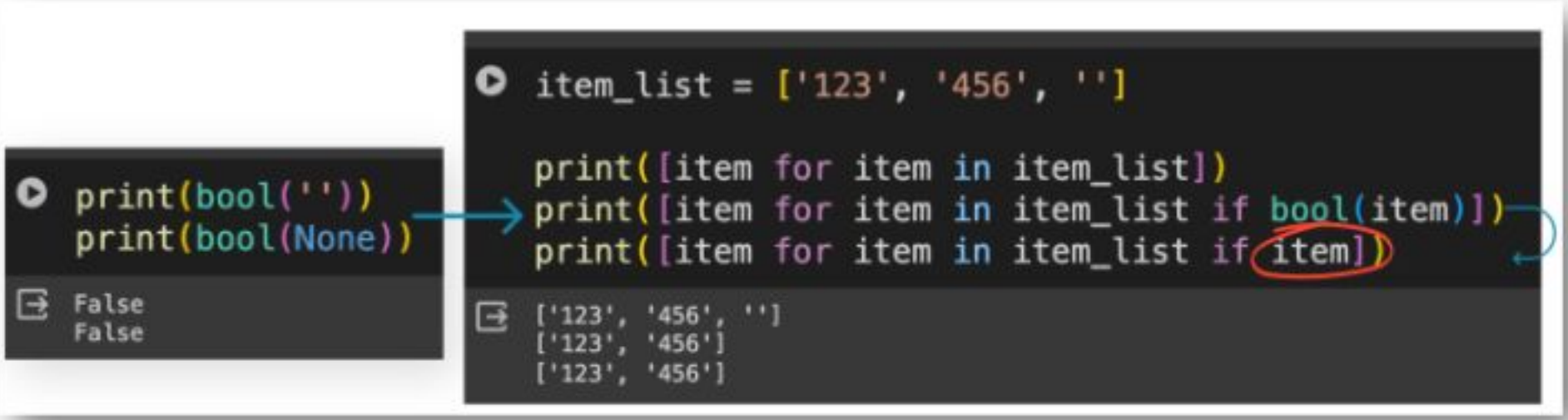
```
[0, 2, 4, 6, 8, 10]
```

```
# 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]
```

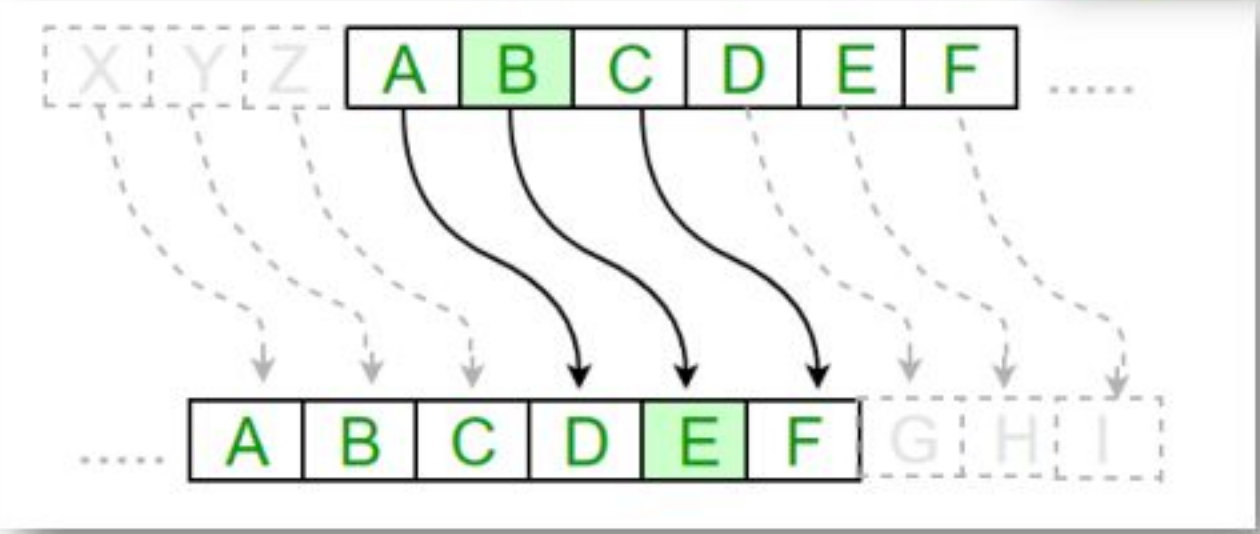
Loops – Using "for"

- List comprehension + Filtering



Practice 3 – Caesar Cipher

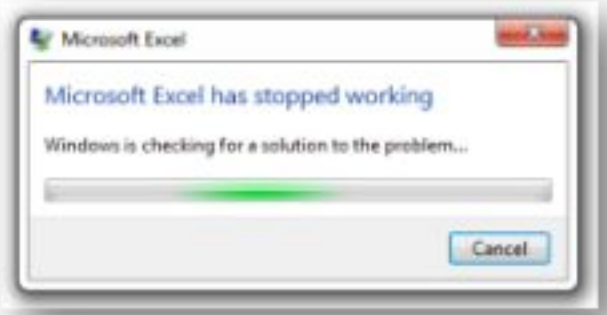
- **Encryption: Shift each letter by 3**
  - Reference: https://cryptii.com/pipes/caesar-cipher



- **Starting notebook:**
  https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

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')

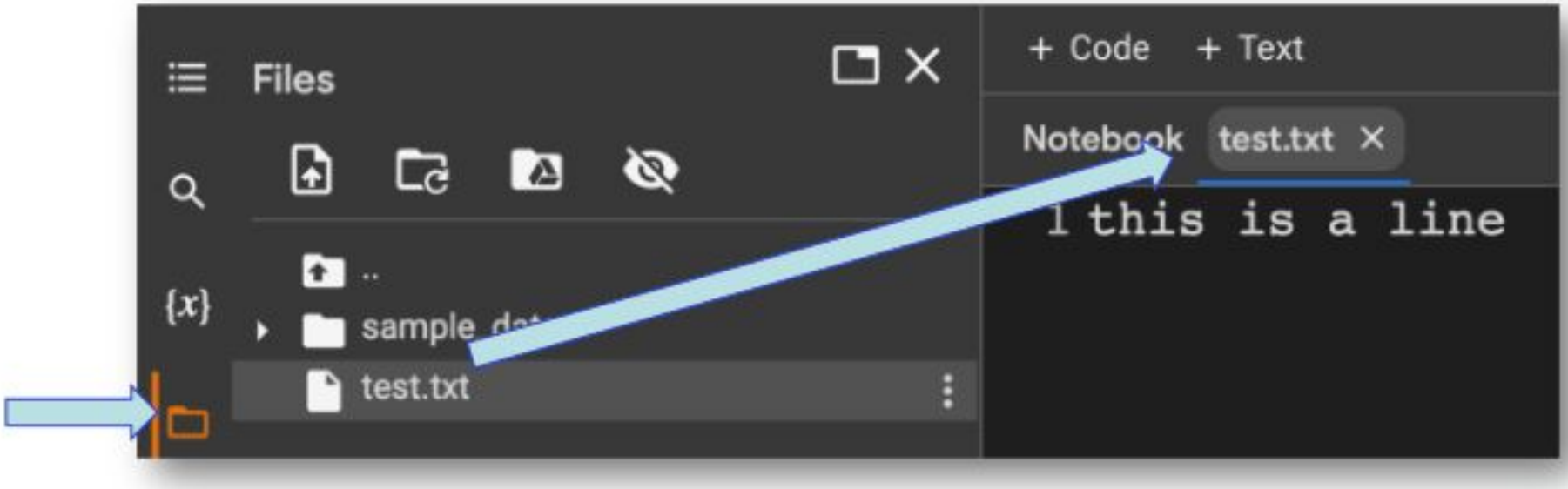
File Input / Output

- **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()
```

File Input / Output

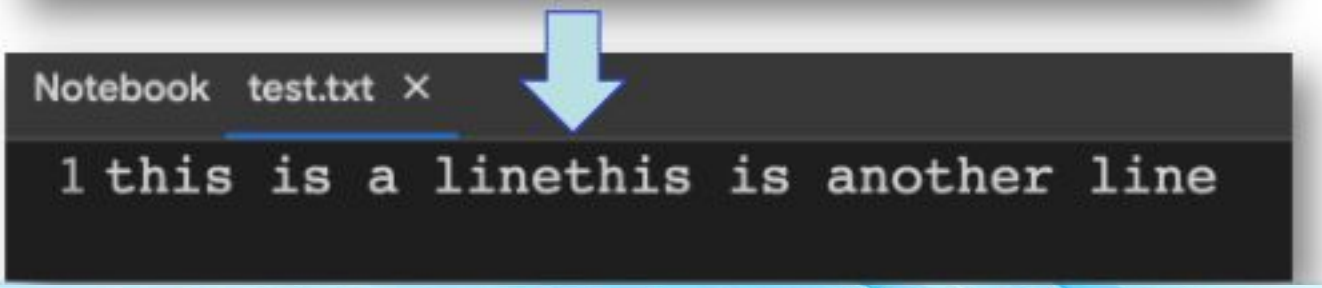
- **Writing a text file**
  - Viewing the result



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()
```



File Input / Output

- **Writing a text file**
  - Multiple lines – the solution
  - ➔ The next-line character "\n"

Notebook test.txt ×

```
1 this is a line
2 this is another line
```

```
# 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

```
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
```

```
'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']
```

Web Crawler

Definition

- a web crawler is a computer program that browses the World Wide Web in a methodical, automated manner. (Wikipedia)

Utilities:

- Gather pages from the Web.
- Support a search engine, perform data mining and so on.

Object:

- Text, video, image and so on.
- Link structure

Web Features of a Crawler

Should provide:
- Distributed
- Scalable
- Performance and efficiency
- Quality
- Freshness
- Extensible

Some scraping knowledge

1. HTTP: the communication protocol
2. HTML: the language in which web pages are defined
3. JS: javascript (code executing in the browser)
4. CSS: style sheets, how web pages are styled. Important, but does not contain data.
5. JPG, PNG, BMP: images, usually not interesting
6. CSV / TXT / JSON / XML: data, interesting !!!

Obtaining Data

Data can come from:

- You curate it
- Someone else provides it, all pre-packaged for you (e.g., files)
- Someone else provides an API
- Someone else has available content, and you try to take it (web scraping)

Obtaining Data – Web Scraping

Web Scraping

- Using programs to get data from online
- Often much faster than manually copying data!
- Transfer the data into a form that is compatible with your code

Obtaining Data – Web Scraping

Why scraping the web?

- Vast source of information; can combine with multiple datasets
- Companies have not provided APIs
- Automate tasks
- Keep up with sites / real-time data
- Fun!

Obtaining Data – Web Scraping

Robots.txt

- Specified by web site owner
- Gives instructions to web robots (e.g., your code)
- Located at the top-level directory of the web server
  - E.g., http://google.com/robots.txt

Robots.txt

- Protocol for giving spiders ("robots") limited assess to a website

  www.robotstxt.org/wc/norobots.html

- Website announced its request on what can(not) be crawled

- For a server, create a file /robots.txt

- This file specifies access restrictions

Robots.txt

- Protocol for giving spiders ("robots") limited assess to a website

  www.robotstxt.org/wc/norobots.html

- Website announced its request on what can(not) be crawled

- For a server, create a file /robots.txt

- This file specifies access restrictions

An example of robots.txt

- No rebot should visit any URL starting with "/yoursite/temp/", except the rebot called "searchengine":
- User-agent: *
- Disallow: /yoursite/temp/
- User-agent: searchengine
- Disallow:

Obtaining Data – Web Scraping

Web Servers

- A server maintains a long-running process (also called a daemon), which listens on a pre-specified port
- It responds to requests, which is sent using a protocol called HTTP (HTTPS is secure)
- Our browser sends these requests and downloads the content, then displays it
- 2– request was successful, 4– client error, often `page not found`; 5– server error (often that your request was incorrectly formed)

Obtaining Data – Web Scraping

HTML

- Tags are denoted by angled brackets
- Almost all tags are in pairs e.g., Hello
- Some tags do not have a closing tag e.g.,

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ttle</title>
  </head>
  <body>
    <h1>Body Title</h1>
    <p>Body Content</p>
  </body>
</html>
```

Obtaining Data – Web Scraping

HTML

- <html>, indicates the start of an html page
- <body>, contains the items on the actual webpage (text, links, images, etc)
- <p>, the paragraph tag. Can contain text and links
- <a>, the link tag. Contains a link url, and possibly a description of the link
- <input>, a form input tag. Used for text boxes, and other user input
- <form>, a form start tag, to indicate the start of a form
- <img>, an image tag containing the link to an image

Obtaining Data – Web Scraping

How to Web scrape:

1. Get the webpage content
    - Requests (Python library) gets a webpage for you
2. Parse the webpage content
    - (e.g., find all the text or all the links on a page)
    - BeautifulSoup (Python library) helps you parse the webpage.
    - Documentation: http://crummy.com/software/BeautifulSoup

The Big Picture Recap

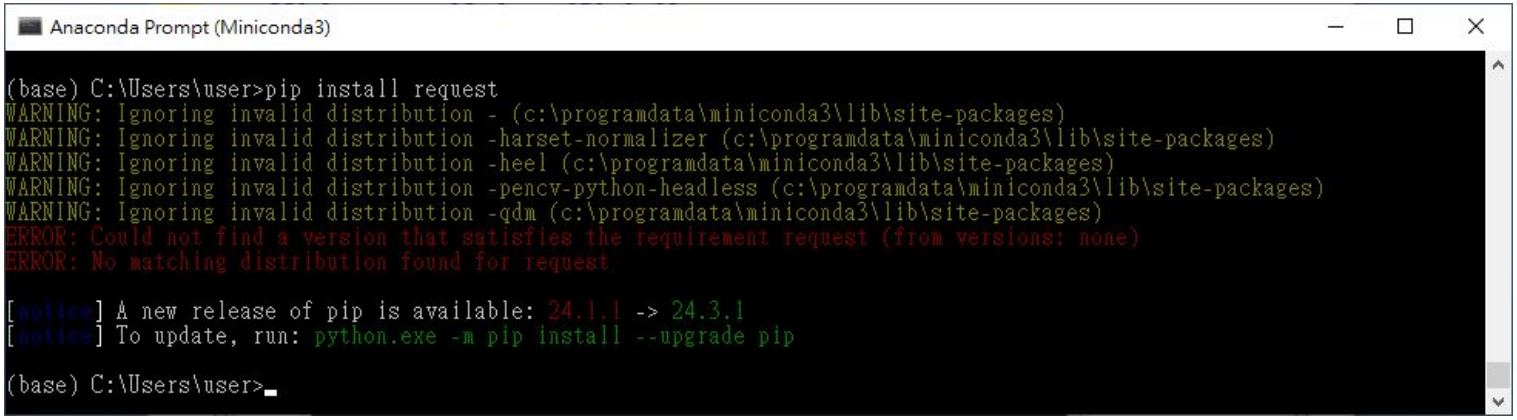
Data Sources	Files, APIs, Webpages (via Requests)
Data Parsing	Regular Expressions, Beautiful Soup
Data Structures/Storage	Traditional lists/dictionaries, PANDAS
Models	Linear Regression, Logistic Regression, kNN, etc

BeautifulSoup only concerns webpage data

Obtaining Data – Web Scraping

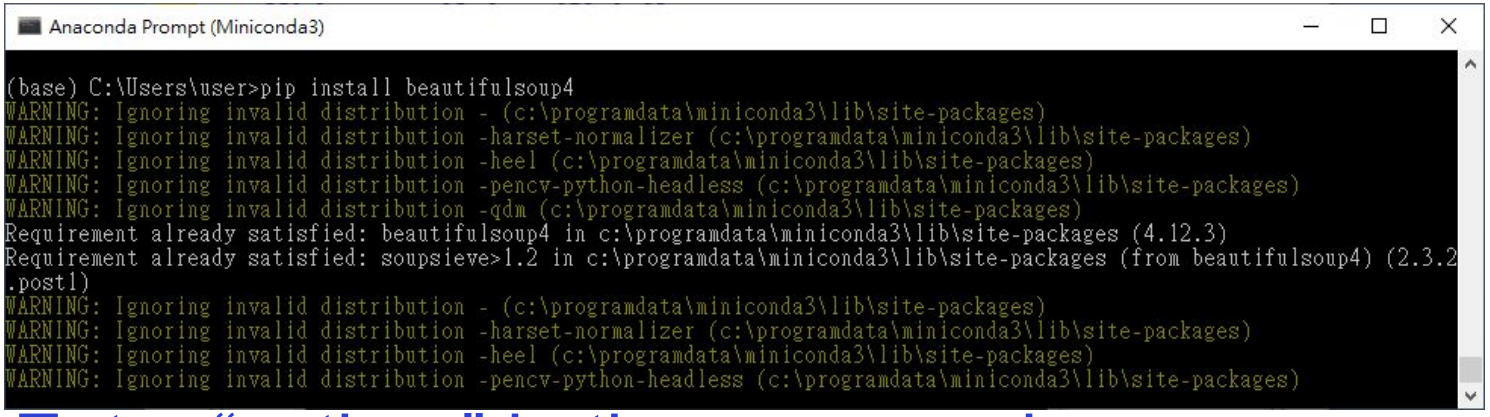
1. Open tools “Anaconda Prompt” which will direct you to the command prompt.

2. Install python package requests by entering “pip install request”

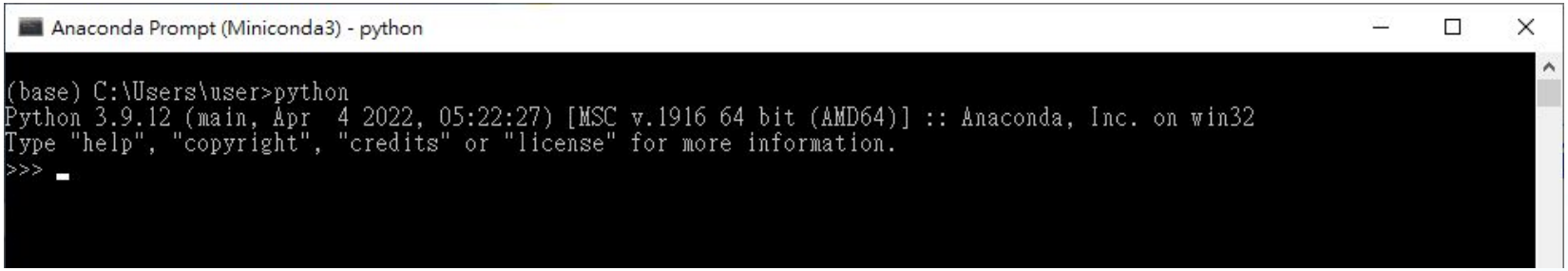


Obtaining Data – Web Scraping

1. Install python package beautifulsoup4 by entering “pip install beautifulsoup4”



2. Enter “python” in the command



Obtaining Data – Web Scraping

1. Get the webpage content

Requests (Python library) gets a webpage for you

```
import requests

url =
"https://www.nytimes.com/internationa
l/"

page = requests.get(url)

page.status_code

page.content
```

Obtaining Data – Web Scraping

1. Get the webpage content

Requests (Python library) gets a webpage for you

```
import requests

url =
"https://www.nytimes.com/
l/"

page = requests.get(url)
page.status_code
page.content
```

Gets the status from the webpage request. 200 means success. 404 means page not found.

Obtaining Data – Web Scraping

1. Get the webpage content

Requests (Python library) gets a webpage for you

```
import requests

url =
"https://www.nytimes.com/internationa
l/"

page = requests.get(url)

page.status_code

page.content
```

Returns the content of the response, in bytes.

Obtaining Data – Web Scraping

2. Parse the webpage content

BeautifulSoup (Python library) helps you parse a webpage

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Obtaining Data – Web Scraping

2. Parse the webpage content

BeautifulSoup (Python library) helps you parse a webpage

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Returns the full context, including the title tag. e.g.,

```
<title data-rh="true">The New York Times - Breaking News</title>
```

Obtaining Data – Web Scraping

2. Parse the webpage content

BeautifulSoup (Python library) helps you parse a webpage

```
soup = BeautifulSoup(page.content, "html.parser")
soup.title
soup.title.text
```

Returns the text part of the title tag. e.g.,

```
The New York Times - Breaking News
```

Obtaining Data – Web Scraping

BeautifulSoup

- Helps make messy HTML digestible
- Provides functions for quickly accessing certain sections of HTML content

Exampl

```
import bs4
## get bs4 object
soup = bs4.BeautifulSoup(source)
## all a tags
soup.findAll('a')
## first a
soup.find('a')
## get all links in the page
link_list = [l.get('href') for l in soup.findAll('a')]
```

Obtaining Data – Web Scraping

HTML is a tree

- You don't have to access the HTML as a tree, though;
- Can immediately search for tags/content of interest (a la previous slide)

Exampl

```
tree = bs4.BeautifulSoup(source)

## get html root node
root_node = tree.html

## get head from root using contents
head = root_node.contents[0]

## get body from root
body = root_node.contents[1]

## could directly access body
tree.body
```

Data Collection - Reading API

Case Study - Read Weather Forcasating from HKO API

Step 1. Import the required libraries:

```
#import necessary library
import requests
import pandas as pd
```

Step 2: Load the weather forecasting data from 'https://data.weather.gov.hk/weatherAPI/opendata/'

```
# data dict is saved in https://www.hko.gov.hk/en/weatherAPI/doc/files/HKO_Open_Data_API_Documentation.pdf
# Parameters for the API
datatype = 'fnd'
lang = 'en'
apiurl = f'https://data.weather.gov.hk/weatherAPI/opendata/weather.php?dataType={datatype}&lang={lang}'

# Fetching the JSON data
response = requests.get(apiurl)
response.raise_for_status()  # Check for request errors

# Converting JSON to a DataFrame
json_data = response.json()

# Convert JSON data to DataFrame
weatherforcast_df = pd.DataFrame(json_data['weatherForecast'])

# Display the DataFrame
print(weatherforcast_df.head())
```

```
  forecastDate       week                                        forecastWind  \
0     20250115  Wednesday                                    North force 4 to 5.
1     20250116   Thursday      North to northeast force 4, force 5 at first.
2     20250117     Friday             Northeast force 4, east force 5 later.
3     20250118   Saturday  East to northeast force 4 to 5, occasionally f...
4     20250119     Sunday                  East to northeast force 2 to 3.

                                 forecastWeather  \
0  Mainly cloudy with a few light rain patches at...
1  Fine. Rather cool in the morning. Very dry dur...
2  Fine. Rather cool in the morning. Very dry dur...
3                          Dry with sunny periods.
4                            Mainly fine and dry.

             forecastMaxtemp             forecastMintemp  \
0  {'value': 20, 'unit': 'C'}  {'value': 15, 'unit': 'C'}
1  {'value': 19, 'unit': 'C'}  {'value': 13, 'unit': 'C'}
2  {'value': 19, 'unit': 'C'}  {'value': 13, 'unit': 'C'}
3  {'value': 20, 'unit': 'C'}  {'value': 15, 'unit': 'C'}
4  {'value': 21, 'unit': 'C'}  {'value': 16, 'unit': 'C'}

                    forecastMaxrh                    forecastMinrh  \
0  {'value': 80, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}
1  {'value': 65, 'unit': 'percent'}  {'value': 35, 'unit': 'percent'}
2  {'value': 65, 'unit': 'percent'}  {'value': 30, 'unit': 'percent'}
3  {'value': 70, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}
4  {'value': 70, 'unit': 'percent'}  {'value': 40, 'unit': 'percent'}

   ForecastIcon  PSR
0            51  Low
1            92  Low
2            50  Low
3            51  Low
4            50  Low
```

Data Collection - Reading API

#Continue from previous page

Step 3. Saving Weather Forecasting DataFrame to a CSV

filecsv_file = 'weatherForecast.csv' # Desired output CSV file
nameweatherforcast_df.to_csv(csv_file, index=False)

print(f"Data has been successfully saved to {csv_file}")

A	B	C	D	E	F	G	H	I	J
forecastDate	week	forecastWind	forecastWeather	forecastMaxtemp	forecastMintemp	forecastMaxrh	forecastMinrh	ForecastIcon	PSR
20250115	Wednesday	North force 4 to 5.	Mainly cloudy with a fe	{'value': 20, 'unit':	{'value': 15, 'unit':	{'value': 80, 'un	{'value': 40, 'un	51	Low
20250116	Thursday	North to northeast force	Fine. Rather cool in the r	{'value': 19, 'unit':	{'value': 13, 'unit':	{'value': 65, 'un	{'value': 35, 'un	92	Low
20250117	Friday	Northeast force 4, east	Fine. Rather cool in the r	{'value': 19, 'unit':	{'value': 13, 'unit':	{'value': 65, 'un	{'value': 30, 'un	50	Low
20250118	Saturday	East to northeast force	Dry with sunny periods.	{'value': 20, 'unit':	{'value': 15, 'unit':	{'value': 70, 'un	{'value': 40, 'un	51	Low
20250119	Sunday	East to northeast force	Mainly fine and dry.	{'value': 21, 'unit':	{'value': 16, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250120	Monday	East to northeast force	Fine and dry.	{'value': 22, 'unit':	{'value': 16, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250121	Tuesday	East to northeast force	Fine and dry.	{'value': 22, 'unit':	{'value': 17, 'unit':	{'value': 70, 'un	{'value': 40, 'un	50	Low
20250122	Wednesday	East force 4, occasiona	Sunny periods.	{'value': 21, 'unit':	{'value': 17, 'unit':	{'value': 75, 'un	{'value': 50, 'un	51	Low
20250123	Thursday	East force 4, occasiona	Sunny intervals.	{'value': 21, 'unit':	{'value': 17, 'unit':	{'value': 75, 'un	{'value': 50, 'un	52	Low

Combining several CSV File

This Tasks is to combining Building Information and School Net Numbers

Step 1. Import the required libraries:

#Loading orginal CSV file containing property data:

```
hkbuildingdata = pd.read_csv('hkbuildinginfo.csv', encoding='ISO-8859-1')
hkbuildingdata.head(5)
```

Step 2: #Clean up the data

```
# Check for missing values
print("\nMissing values in each column:")
print(hkbuildingdata.isnull().sum())

#Handle missing values by Dropping rows with missing values
# For columns with a high percentage of missing values, consider dropping or filling them
# Here we will drop 'block' and 'state' columns due to excessive missing values
hkbuildingdata.drop(columns=['withpre', 'rootid', 'fatherid', 'catid', 'source', 'block', 'state', 'date_dm'],
        inplace=True, errors='ignore')

# Assuming 'floor' is the column causing the error
hkbuildingdata['floor'] = hkbuildingdata['floor'].astype(str)  # Convert to string type

# Now you can safely use the .str accessor
hkbuildingdata['floor'] = hkbuildingdata['floor'].str.replace(r'[^\d]', '', regex=True)  # Remove
        non-numeric characters

# Convert back to numeric if needed
hkbuildingdata['floor'] = pd.to_numeric(hkbuildingdata['floor'], errors='coerce')  # Convert to numeric,
        coercing errors to NaN
```

```
Missing values in each column:
date                  0
withpre           23564
id                    0
rootid                0
fatherid              0
catid                 0
catname               0
catfathername         0
url_father            0
url_cat               0
source            11556
contract              0
memo                  0
price                 0
price_value           0
holddate              0
winloss_flag          0
winloss               0
act_area              0
area                  0
arearaw               4
sq_price              0
sq_price_value        0
sq_actprice           0
sq_actprice_value     0
month                 0
day                   0
date_dm               0
date_y                0
block             32308
state             77003
floor                32
room                258
addr                  0
Unnamed: 34       80000
dtype: int64
```

Combining several CSV File

#Continue from previous page

```
# Continue with your existing code for filling missing values and other operations
hkbuildingdata['floor'] = hkbuildingdata['floor'].fillna(hkbuildingdata['floor'].median())  # Fill with median for numeric data

# Step 4: Fill missing values for 'floor' and 'room'
hkbuildingdata['floor'] = hkbuildingdata['floor'].fillna(hkbuildingdata['floor'].median())  # Fill with median for numeric data
hkbuildingdata['room'] = hkbuildingdata['room'].fillna(hkbuildingdata['room'].mode()[0])   # Fill with mode for categorical data

# Step 5: Fill missing values for 'arearaw'
hkbuildingdata['arearaw'] = hkbuildingdata['arearaw'].fillna(hkbuildingdata['arearaw'].mean())

# Step 6: Rename columns to more meaningful names
hkbuildingdata.rename(columns={'floor': 'Floor', 'arearaw': 'House Area', 'sq_price_value': 'Price Per Square', 'catname': 'Building
        Name', 'catfathername': 'District'}, inplace=True)

# Step 7: Display the cleaned DataFrame and check for remaining missing values
print("\nCleaned DataFrame:")
print(hkbuildingdata.head())

print("\nRemaining Missing Values:")
print(hkbuildingdata.isnull().sum())
```

```
Cleaned DataFrame:
         date      id       Building Name       District  \
0  2020-11-27  683333     Bel Air Heights   Diamond Hill
1  2020-11-27  683332       Fa Yuen Plaza       Mong Kok
2  2020-11-27  683331      Caldecott Hill  Yau Yat Tsuen
3  2020-11-27  683330    Pang Ching Court   Wong Tai Sin
4  2020-11-27  683329  Metro Harbour View   Tai Kok Tsui

                                  url_father  \
0   https://data.28hse.com/en/kl/diamond-hill
1       https://data.28hse.com/en/kl/mong-kok
2  https://data.28hse.com/en/kl/yau-yat-tsuen
3   https://data.28hse.com/en/kl/wong-tai-sin
4   https://data.28hse.com/en/kl/tai-kok-tsui

                                        url_cat   contract  \
0  https://data.28hse.com/en/kl/diamond-hill/2058...  Agreement
1  https://data.28hse.com/en/kl/mong-kok/2712_fa-...  Agreement
2  https://data.28hse.com/en/kl/yau-yat-tsuen/309...  Agreement
3  https://data.28hse.com/en/kl/wong-tai-sin/5035...  Agreement
4  https://data.28hse.com/en/kl/tai-kok-tsui/2564...  Agreement

            memo   price  price_value  ... Price Per Square  sq_actprice  \
0  20112702480015    $10M     10000000  ...         12106.54       $15221
1  20112702470044  $4.28M      4280000  ...         12334.29       $20000
2  20112702440018    $11M     11000000  ...          8580.34       $11100
3  20112702410017   $4.9M      4900000  ...          8404.80           --
4  20112702400032  $7.15M      7150000  ...         12016.81       $17354

  sq_actprice_value month day  date_y Floor  room                addr  \
0          15220.70   Nov  27    2020   2.0     E  BLOCK 2 2#/F Room E
1          20000.00   Nov  27    2020   1.0     C          1#/F Room C
2          11099.90   Nov  27    2020   7.0     E   TOWER 1 7/F Room E
3              0.00   Nov  27    2020   1.0    22         1#/F Room 22
4          17354.37   Nov  27    2020   1.0     B  BLOCK 8 1#/F Room B
```

Combining Building Information and School Net Numbers

#Combine Original Data with School District Data

```
schoolnetnumber_df = pd.read_csv("Primary Sch_Net Numbers By District.csv")

# Normalize the 'District' column in both DataFrames to lowercase
hkbuildingdata['District'] = hkbuildingdata['District'].str.lower()
schoolnetnumber_df['District'] = schoolnetnumber_df['District Areas'].str.lower()

# Merge schooldistrict_df with districtcode_df on the normalized 'District' column
merged_df = hkbuildingdata.merge(schoolnetnumber_df[['District', 'Sch_Net_No']], on='District', how='left')

# Create the 'District_Code' column based on the logic provided
hkbuildingdata['School_Netnumber'] = merged_df['Sch_Net_No'].fillna(0).astype(int)
hkbuildingdata
```

[27]:

	date	id	Building Name	District	url_father	url_cat	contract	memo	price	price_va
0	2020-11-27	683333	Bel Air Heights	diamond hill	https://data.28hse.com/en/kl/diamond-hill	https://data.28hse.com/en/kl/diamond-hill/2058...	Agreement	20112702480015	$10M	10000(
1	2020-11-27	683332	Fa Yuen Plaza	mong kok	https://data.28hse.com/en/kl/mong-kok	https://data.28hse.com/en/kl/mong-kok/2712_fa-...	Agreement	20112702470044	$4.28M	4280(
2	2020-11-27	683331	Caldecott Hill	yau yat tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen/309...	Agreement	20112702440018	$11M	11000(
3	2020-11-27	683330	Pang Ching Court	wong tai sin	https://data.28hse.com/en/kl/wong-tai-sin	https://data.28hse.com/en/kl/wong-tai-sin/5035...	Agreement	20112702410017	$4.9M	4900(
4	2020-11-27	683329	Metro Harbour View	tai kok tsui	https://data.28hse.com/en/kl/tai-kok-tsui	https://data.28hse.com/en/kl/tai-kok-tsui/2564...	Agreement	20112702400032	$7.15M	7150(

5 rows × 29 columns

Exploratory Data Analysis(ETL) and Data Visualization

Exploratory Data Analysis and Visualization

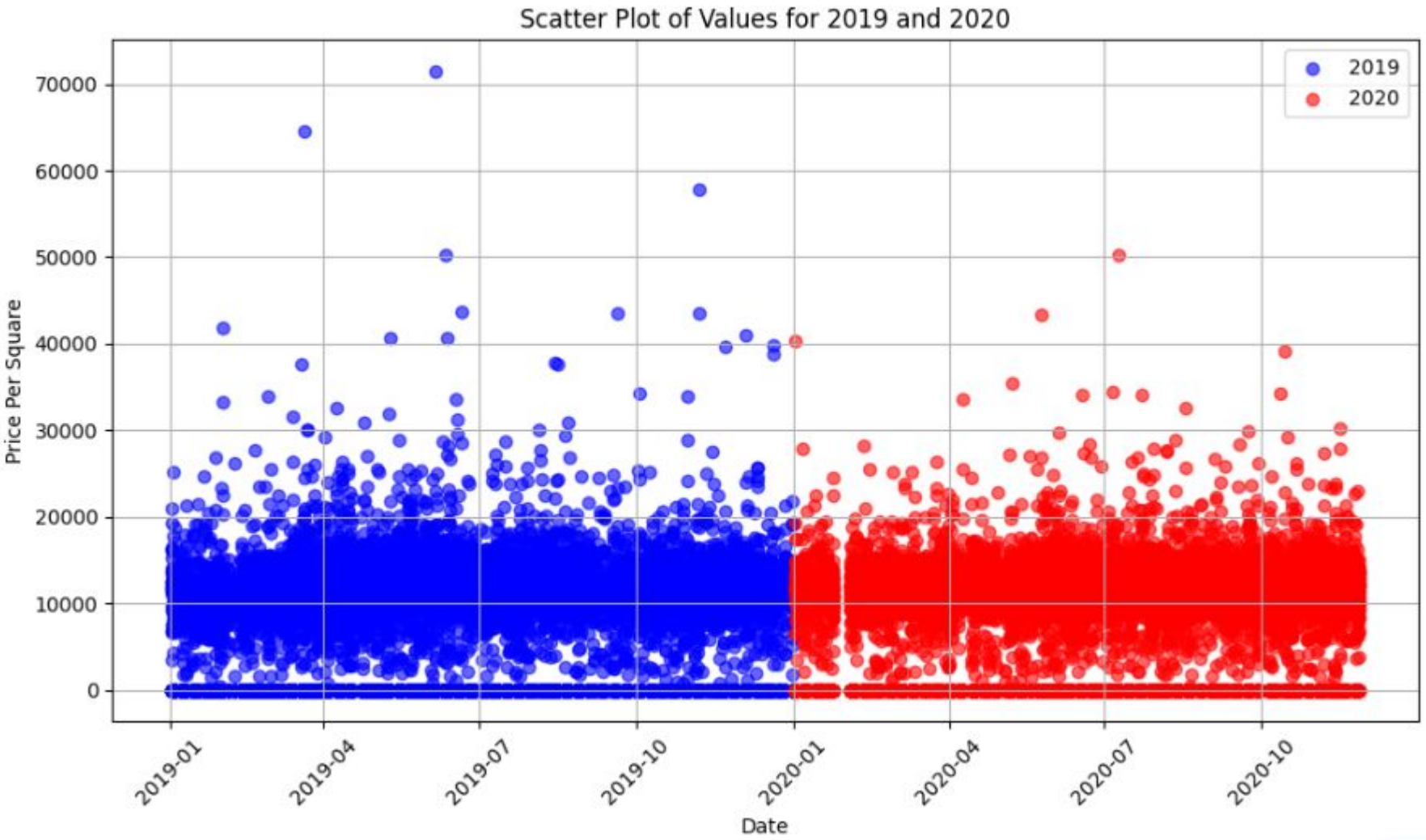
```
#filter the data with period "2020-1-1" to "2020-12-31"
# Create a full date column from 'month', 'day', and 'date_y'
hkbuildingdata['full_date'] = pd.to_datetime(hkbuildingdata['month'] + ' ' + hkbuildingdata['day'].astype(str) + ' ' +
      hkbuildingdata['date_y'].astype(str))

# Filter the DataFrame for the date range "2020-01-01" to "2020-12-31"
start_date = '2020-01-01'
end_date = '2020-12-31'
covidfiltered_data = hkbuildingdata[(hkbuildingdata['full_date'] >= start_date) & (hkbuildingdata['full_date'] <= end_date)]

# Display the first few rows of the filtered DataFrame
print(covidfiltered_data.head(5))

# Filter the DataFrame for the date range "2019-
start_date = '2019-01-01'
end_date = '2019-12-31'
filtered_data = hkbuildingdata[(hkbuildingdata['fu

# Display the first few rows of the filtered DataFr
print(filtered_data.head(5))
```

	date	id	Building Name	District	url_father	url_cat	contract
0	2020-11-27	683333	Bel Air Heights	diamond hill	https://data.28hse.com/en/kl/diamond-hill	https://data.28hse.com/en/kl/diamond-hill/2058...	Agreement
1	2020-11-27	683332	Fa Yuen Plaza	mong kok	https://data.28hse.com/en/kl/mong-kok	https://data.28hse.com/en/kl/mong-kok/2712_fa-...	Agreement
2	2020-11-27	683331	Caldecott Hill	yau yat tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen	https://data.28hse.com/en/kl/yau-yat-tsuen/309...	Agreement
3	2020-11-27	683330	Pang Ching Court	wong tai sin	https://data.28hse.com/en/kl/wong-tai-sin	https://data.28hse.com/en/kl/wong-tai-sin/5035...	Agreement
4	2020-11-27	683329	Metro Harbour View	tai kok tsui	https://data.28hse.com/en/kl/tai-kok-tsui	https://data.28hse.com/en/kl/tai-kok-tsui/2564...	Agreement

Exploratory Data Analysis(ETL) and Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatter plot
plt.figure(figsize=(10, 6))

# Plot for 2019
plt.scatter(filtered_data['full_date'], filtered_data['Price Per Square'], color='blue', label='2019', alpha=0.6)

# Plot for 2020
plt.scatter(covidfiltered_data['full_date'], covidfiltered_data['Price Per Square'], color='red', label='2020', alpha=0.6)

# Adding titles and labels
plt.title('Scatter Plot of Values for 2019 and 2020')
plt.xlabel('Date')
plt.ylabel('Price Per Square')
plt.xticks(rotation=45)
plt.legend()
plt.grid()

# Show the plot
plt.tight_layout()
plt.show()
```

Exploratory Data Analysis(ETL) and Visualization



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

THANK YOU