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 / 23 November 2024

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

Table Of Contents

Python – Origin

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

Motivation – Soft Limits

- Protective measures not matching modern days

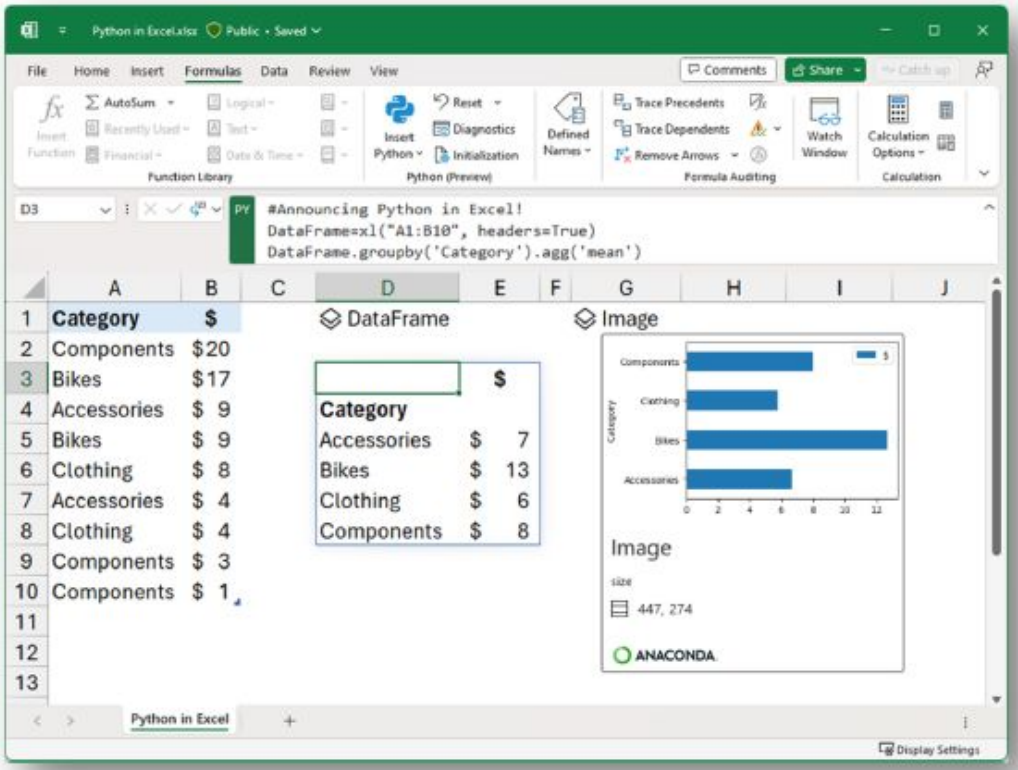
Feature	Maximum limit
Total number of rows and columns on a worksheet	1,048,576 rows by 16,384 columns
Column width	255 characters
Row height	409 points
Page breaks	1,026 horizontal and vertical

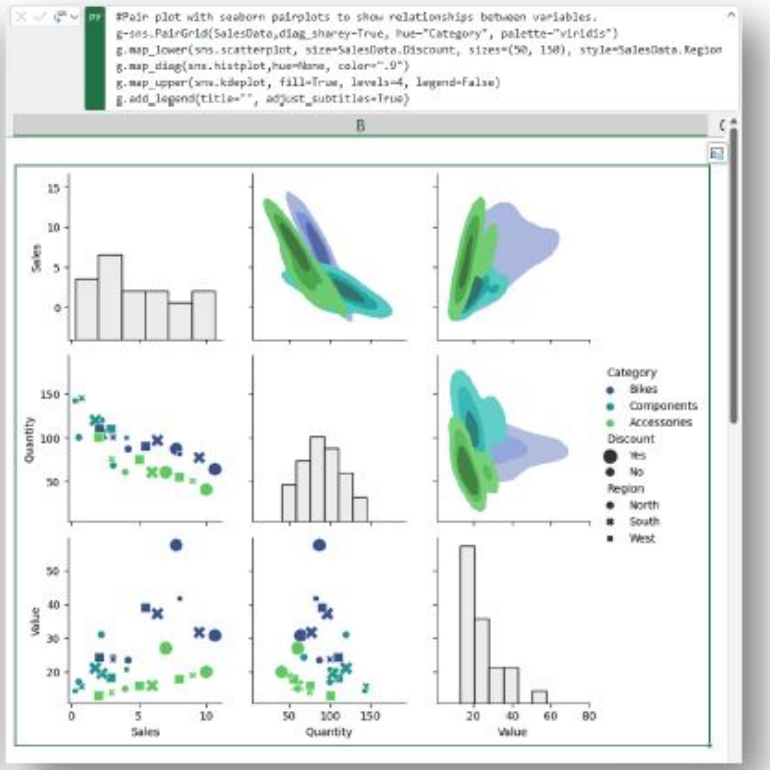
1. Microsoft. Excel Specifications and limits. https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3

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

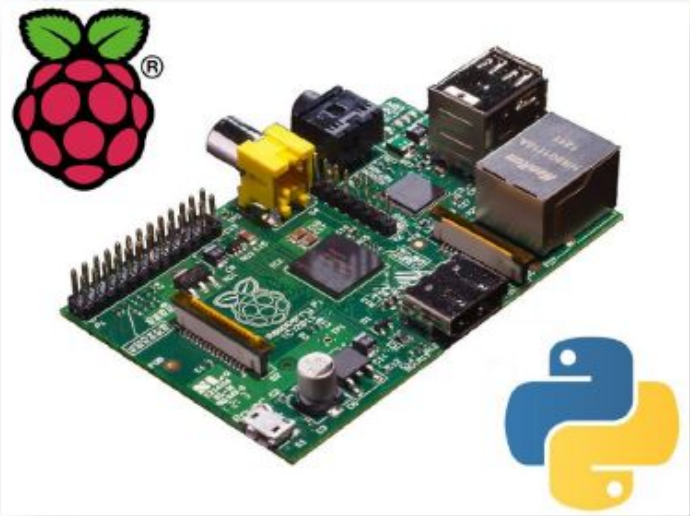
Motivation – Scope

- 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

Motivation – Scope

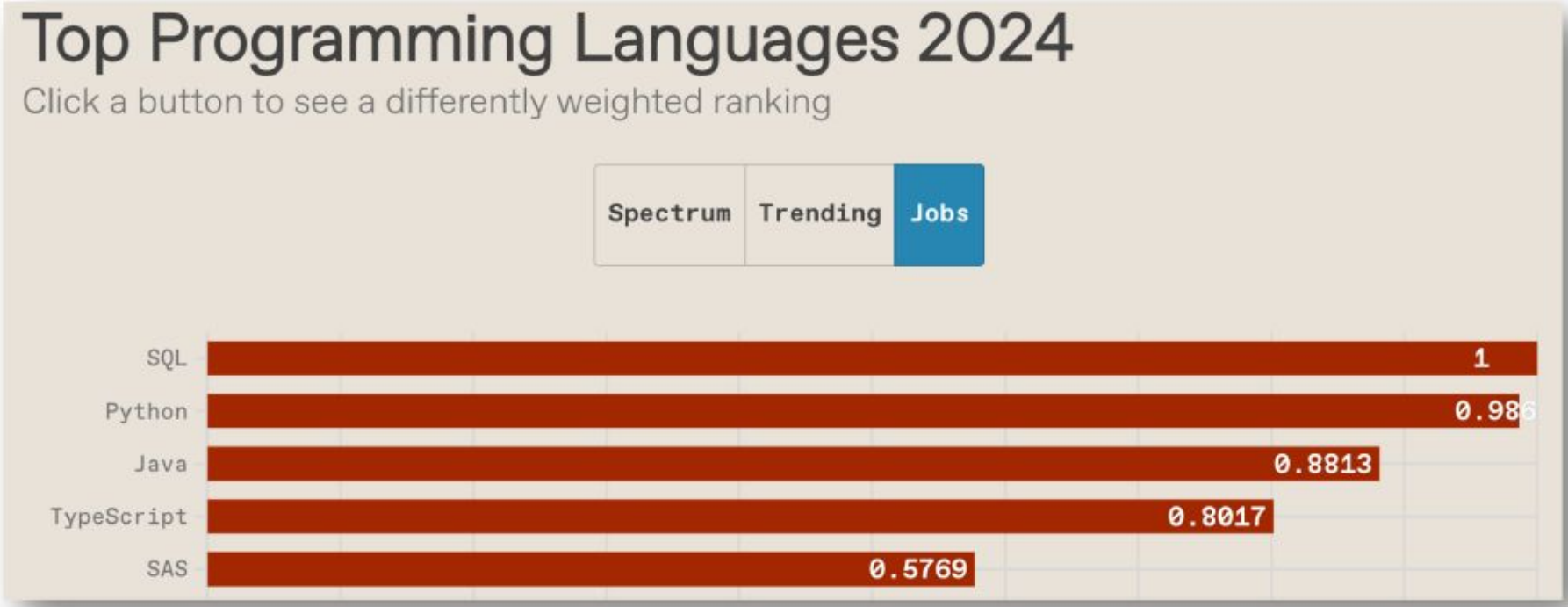
- How about outside of ~~Microsoft~~ Office?
- e.g., Drones / Security cameras / Firewalls



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

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

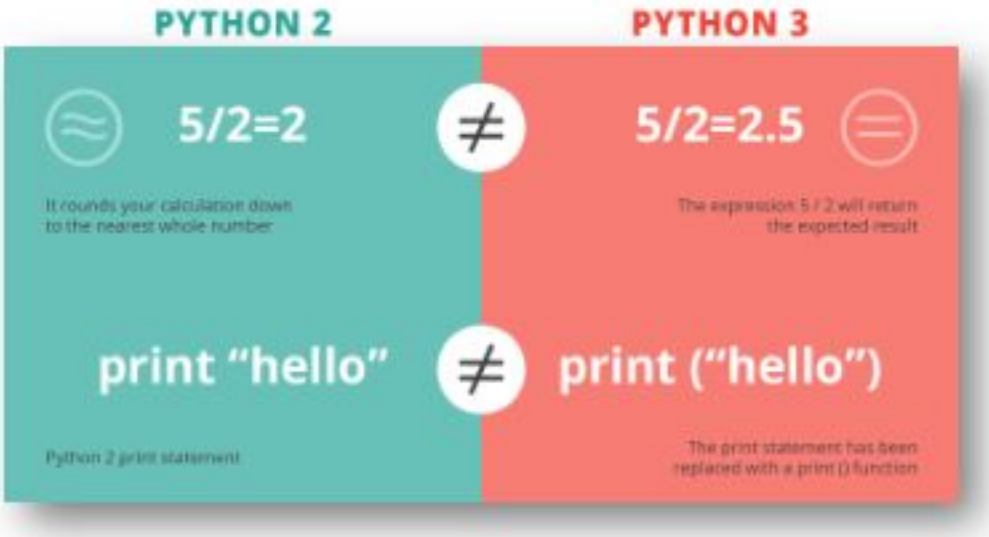
Level II

- **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

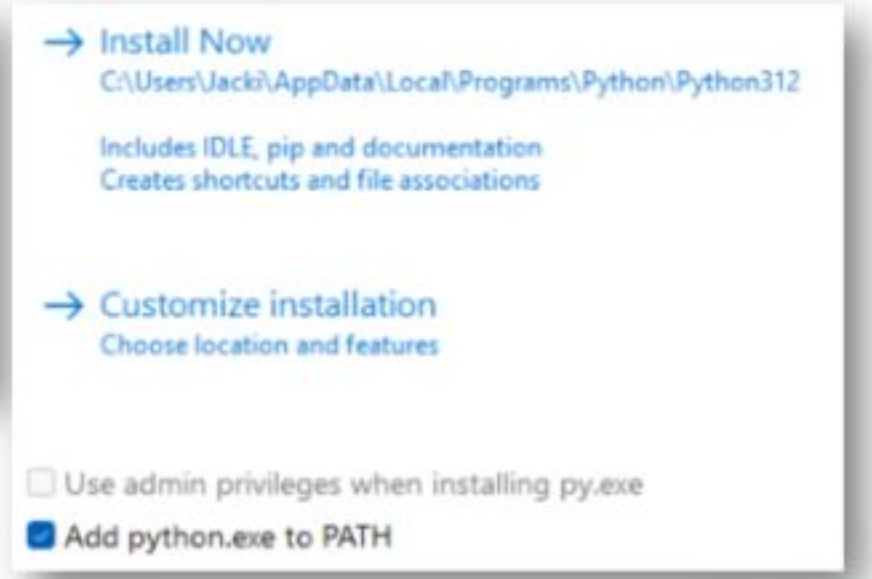
- The language and basic runtime environment
- Just like Apple iOS, it keeps updating (e.g., Python 3.12.2)
  - Python 2 was no longer supported from 2020[1]



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

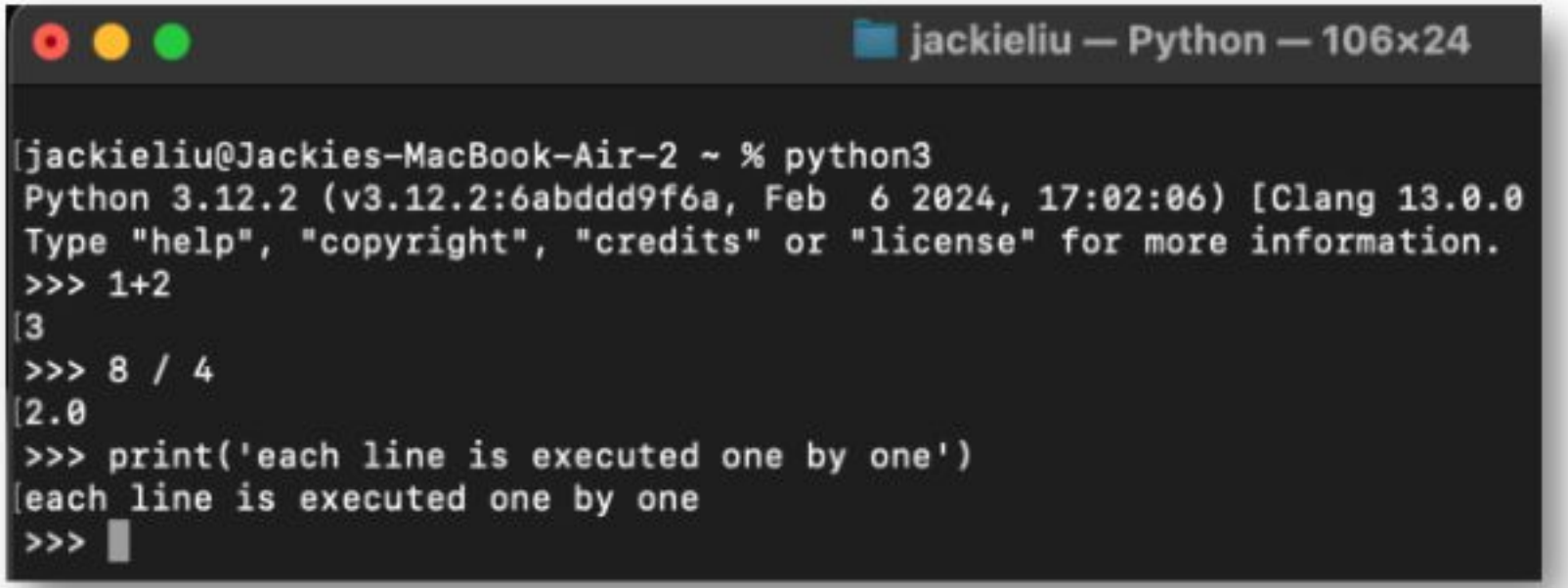
Installation – Python Runtime

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



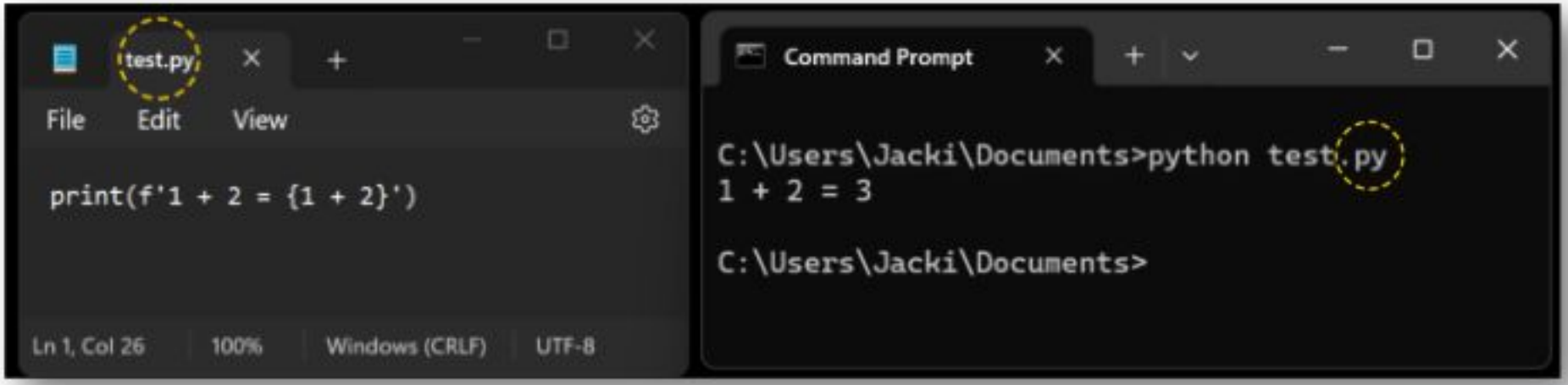
How to run Python – In the old days

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



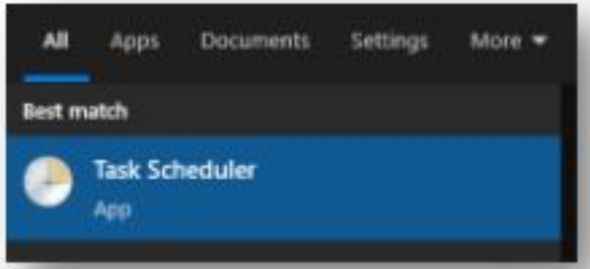
How to run Python – Running as a file

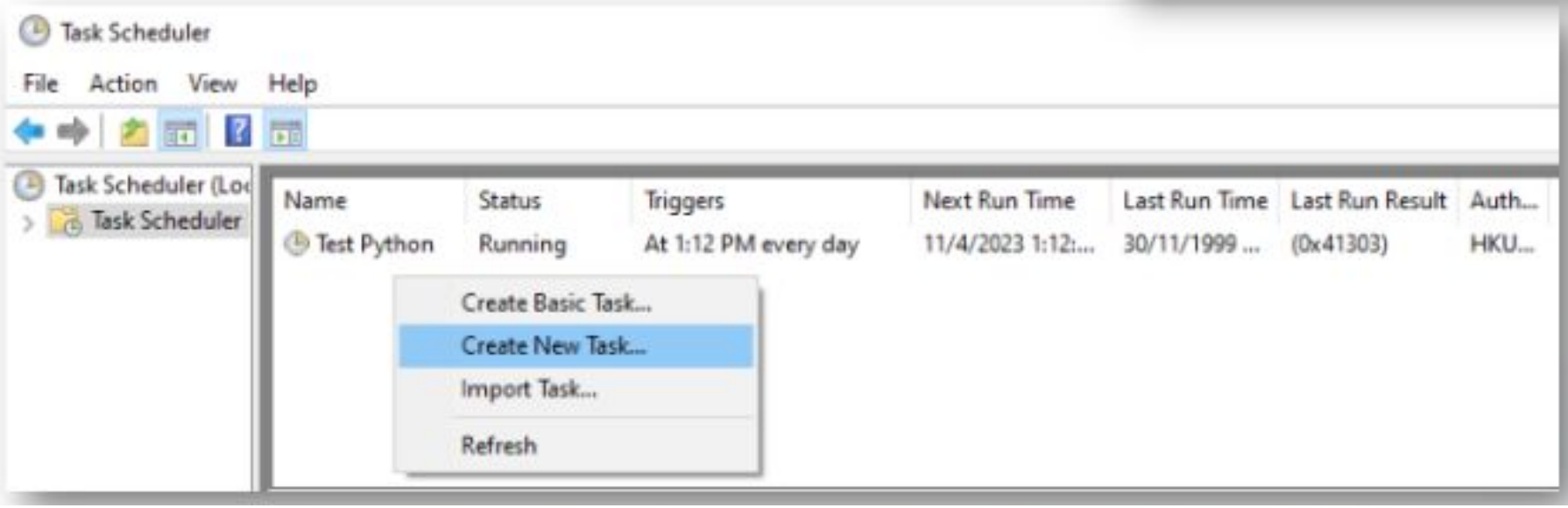
- Python interpreter + Text editor
  - Write codes in a text editor (e.g., Notepad, Vim, Sublime Text)
  - Save codes as a ".py" file
  - Run all the lines in one go



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



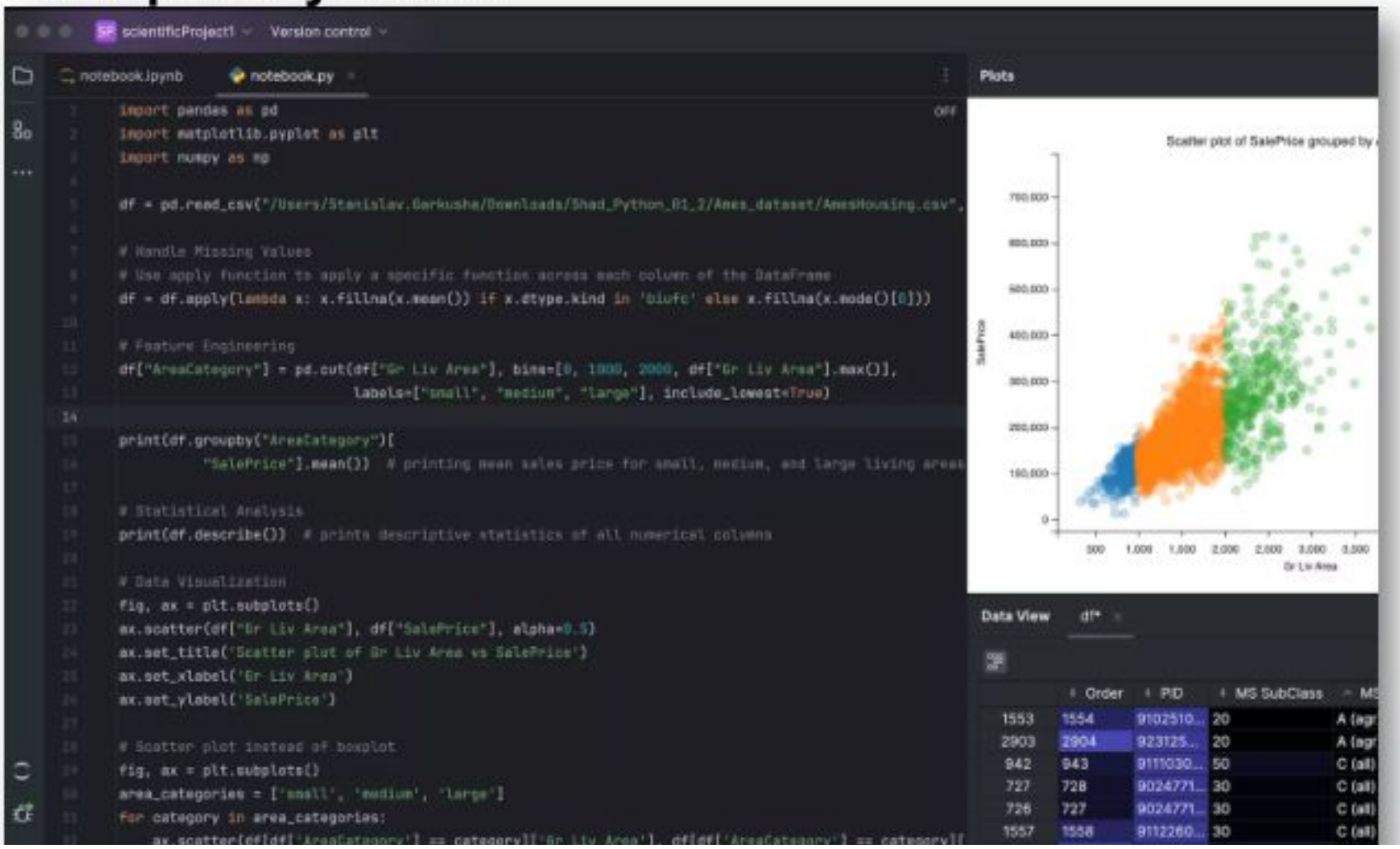


How to run Python – IDE

- 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

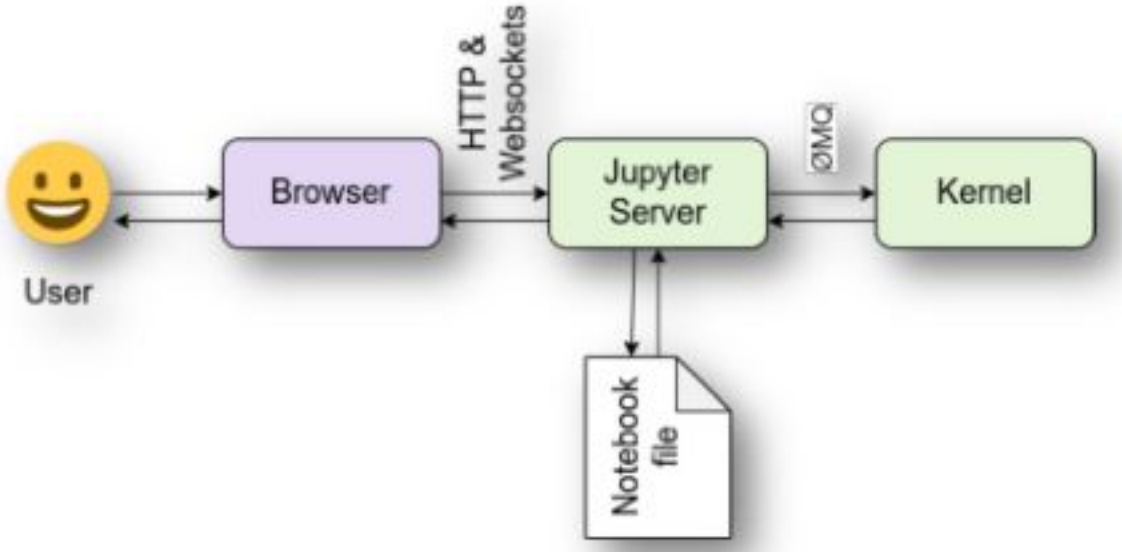
Business Process Automation with Python

Example - PyCharm



How to run Python – Jupyter Notebook

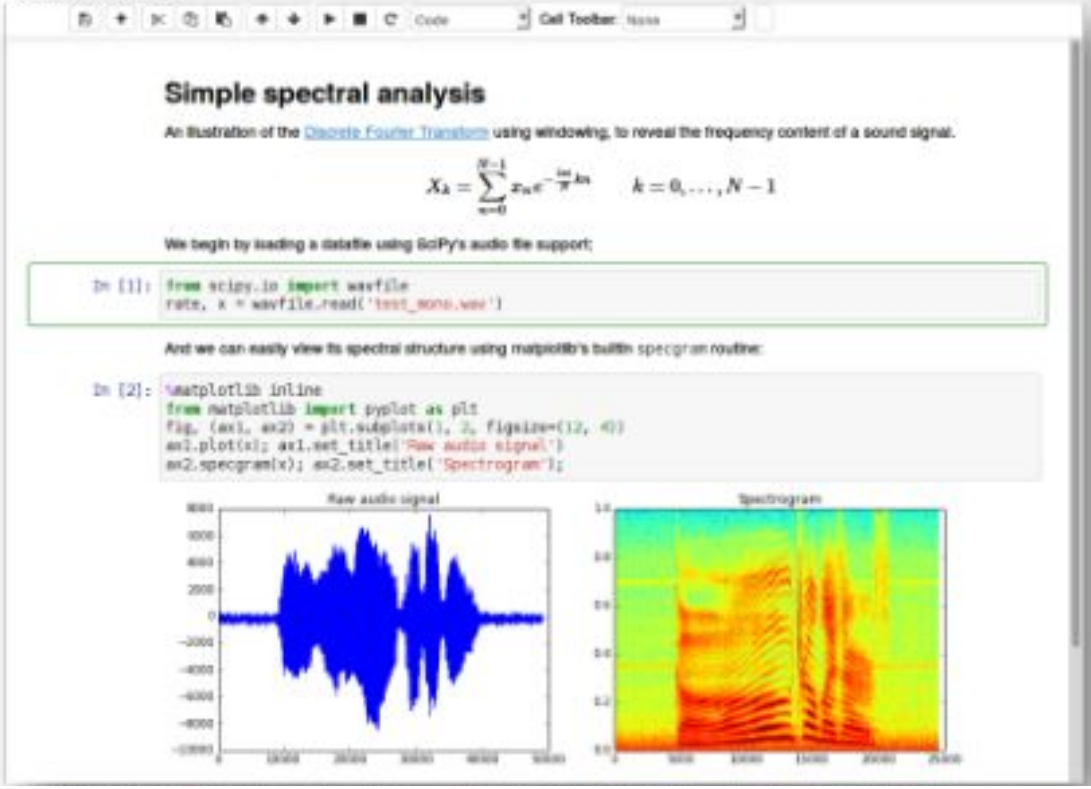
- Web-based interactive platform
  - Accessible: Python runtime on a web server



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

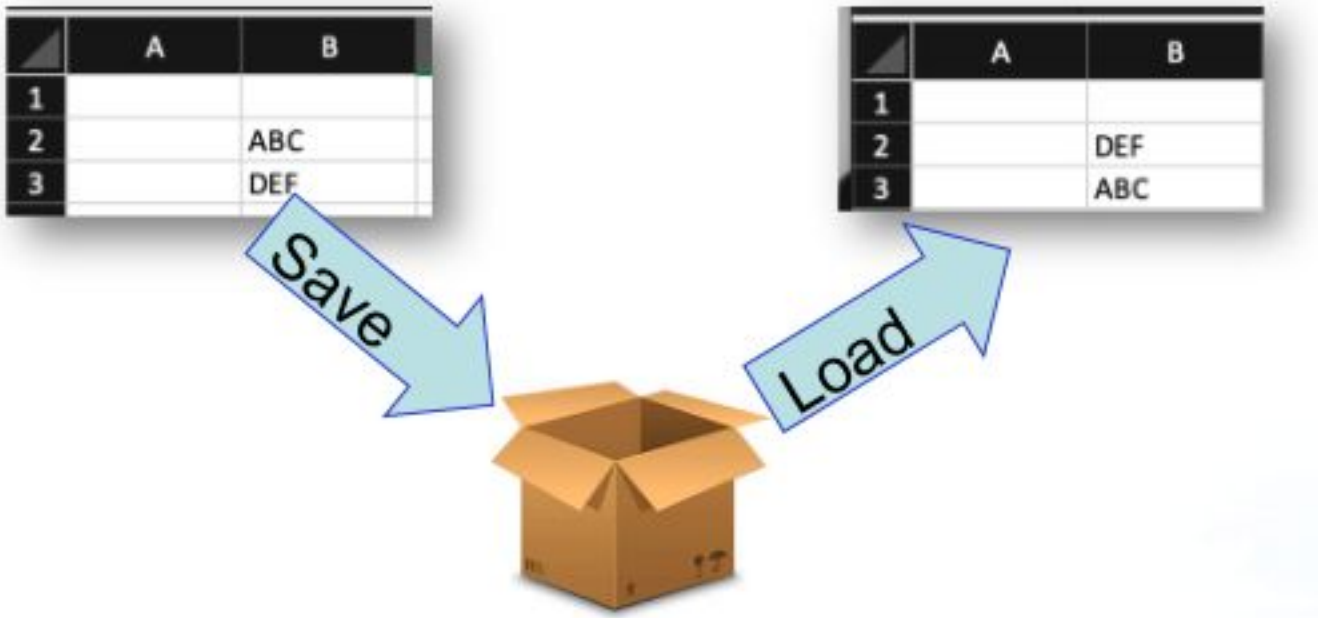
1. *Google Colab FAQ*. https://research.google.com/colaboratory/faq.html
2. *Google Colaboratory*. https://colab.research.google.com

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

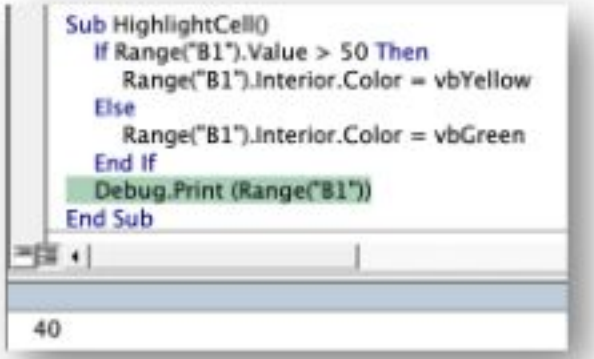


Variables

- 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()



- 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
```

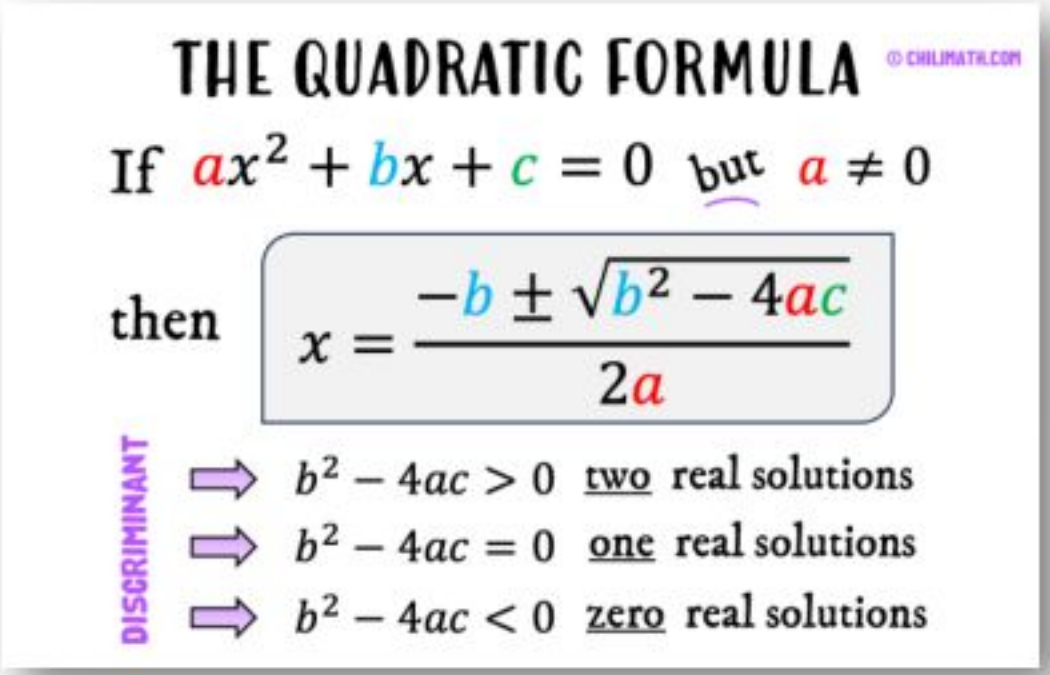
- Float

```
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

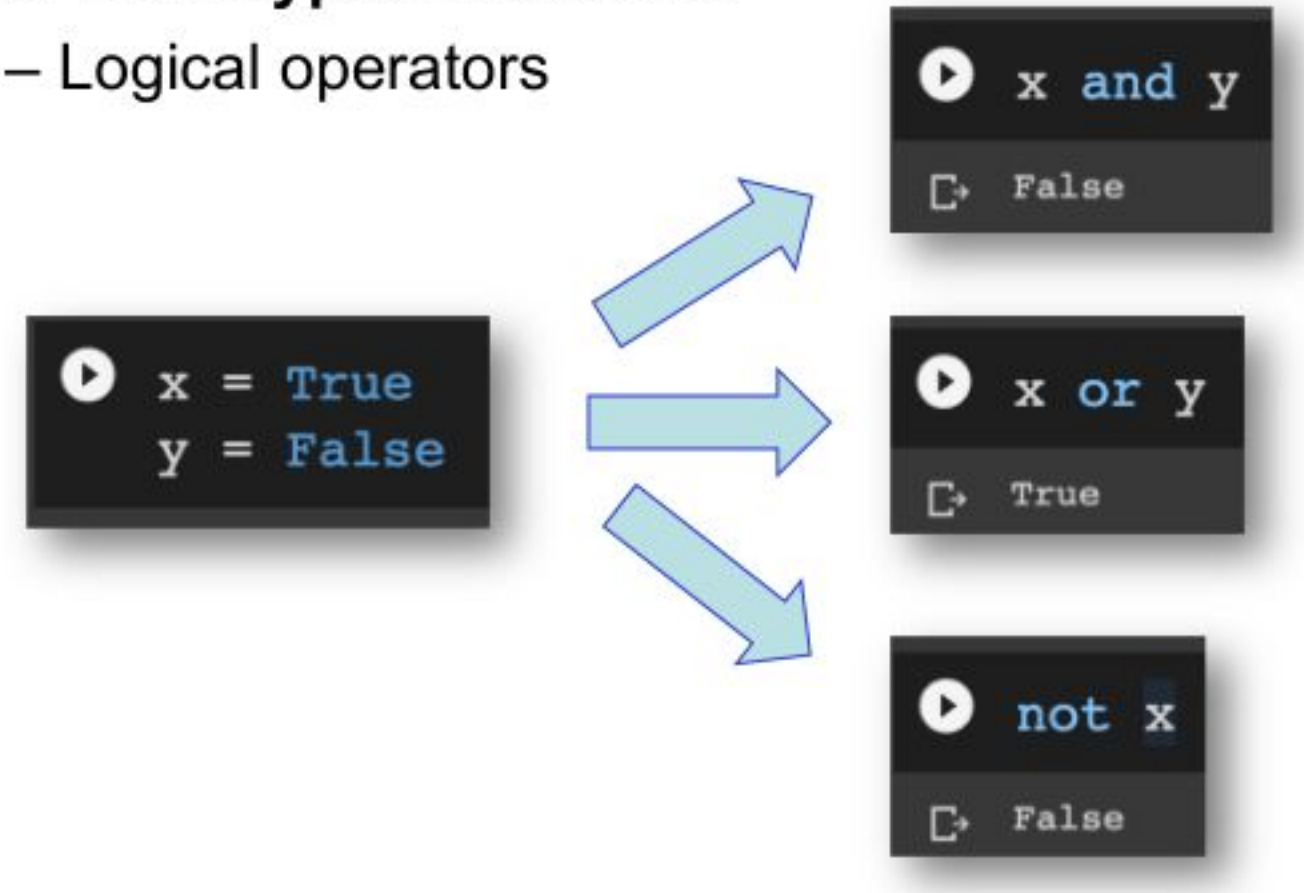
- Common Functions
  - round() / pow() / abs()

```
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

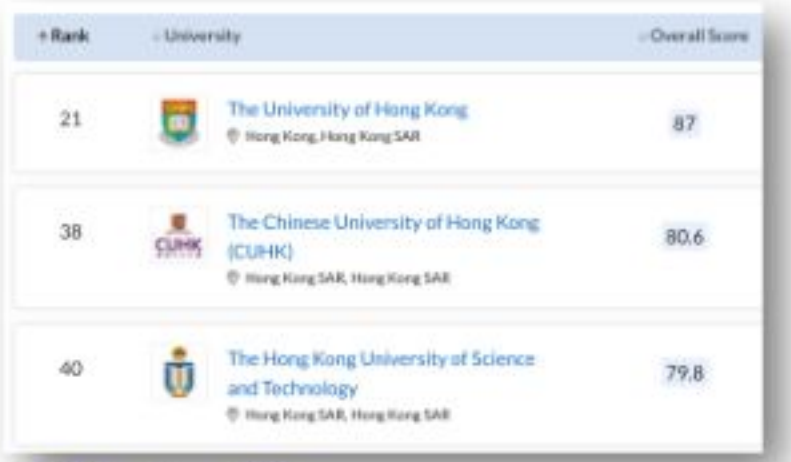
- Compute market capitalization of 2 companies
- Show the difference between the two
- Format the result up to 2 decimal places

- **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

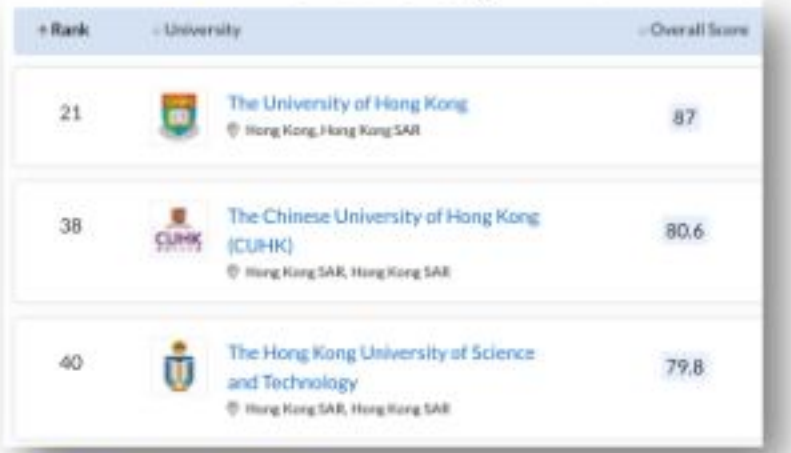


Rank	University	Overall Score
21	The University of Hong Kong Hong Kong, Hong Kong SAR	87
38	The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR	80.6
40	The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR	79.8

* 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 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
```

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!']))
```

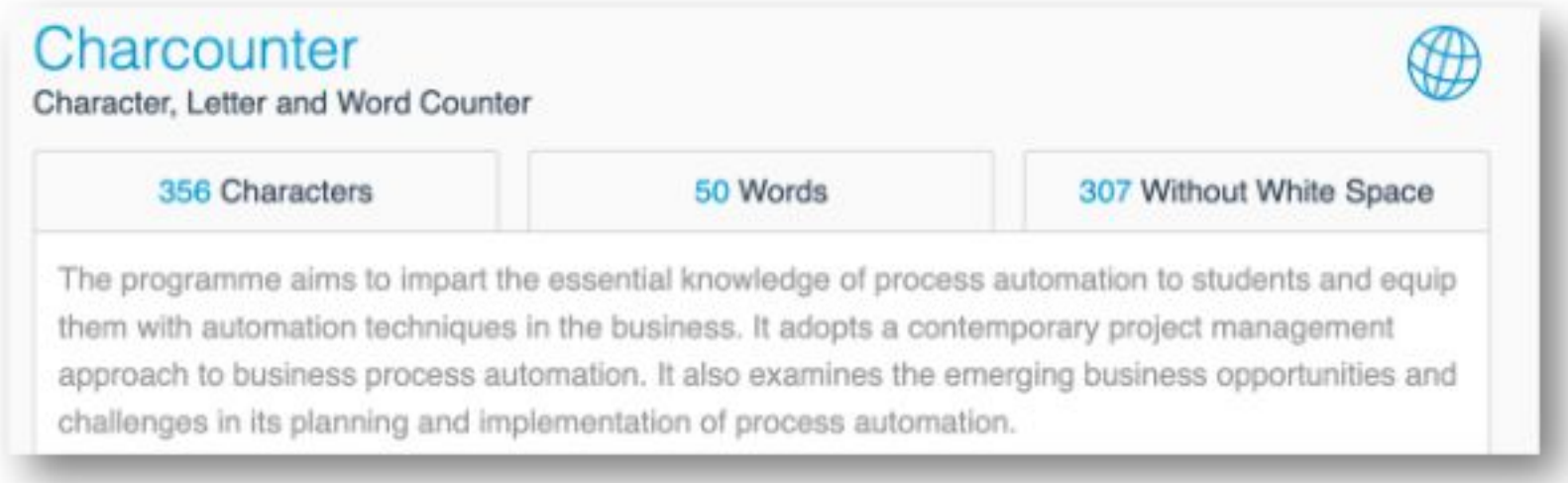
```
hello
Welcome,Jackie!
```

```
# 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'>
```

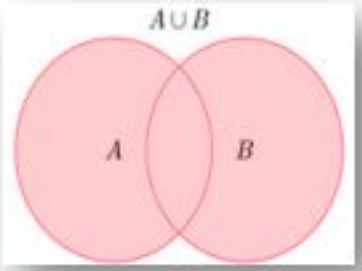
  - Can be converted between sequential data types

```
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

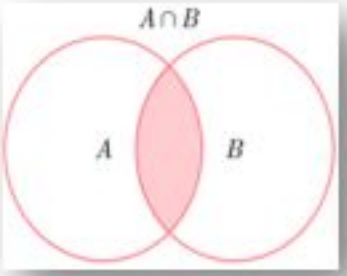



```
# Set operations
school_set_2022 = {'A', 'B', 'C'}
school_set_2023 = {     'B', 'C', 'D'}
```

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

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

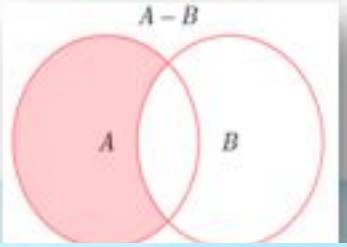
  - Intersection




```
# 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

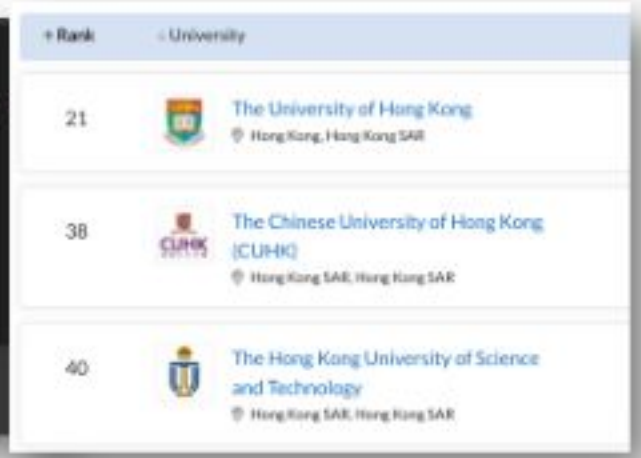
  - Looking up a keyword
    ➔ like finding words in **dictionaries**

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'}
```

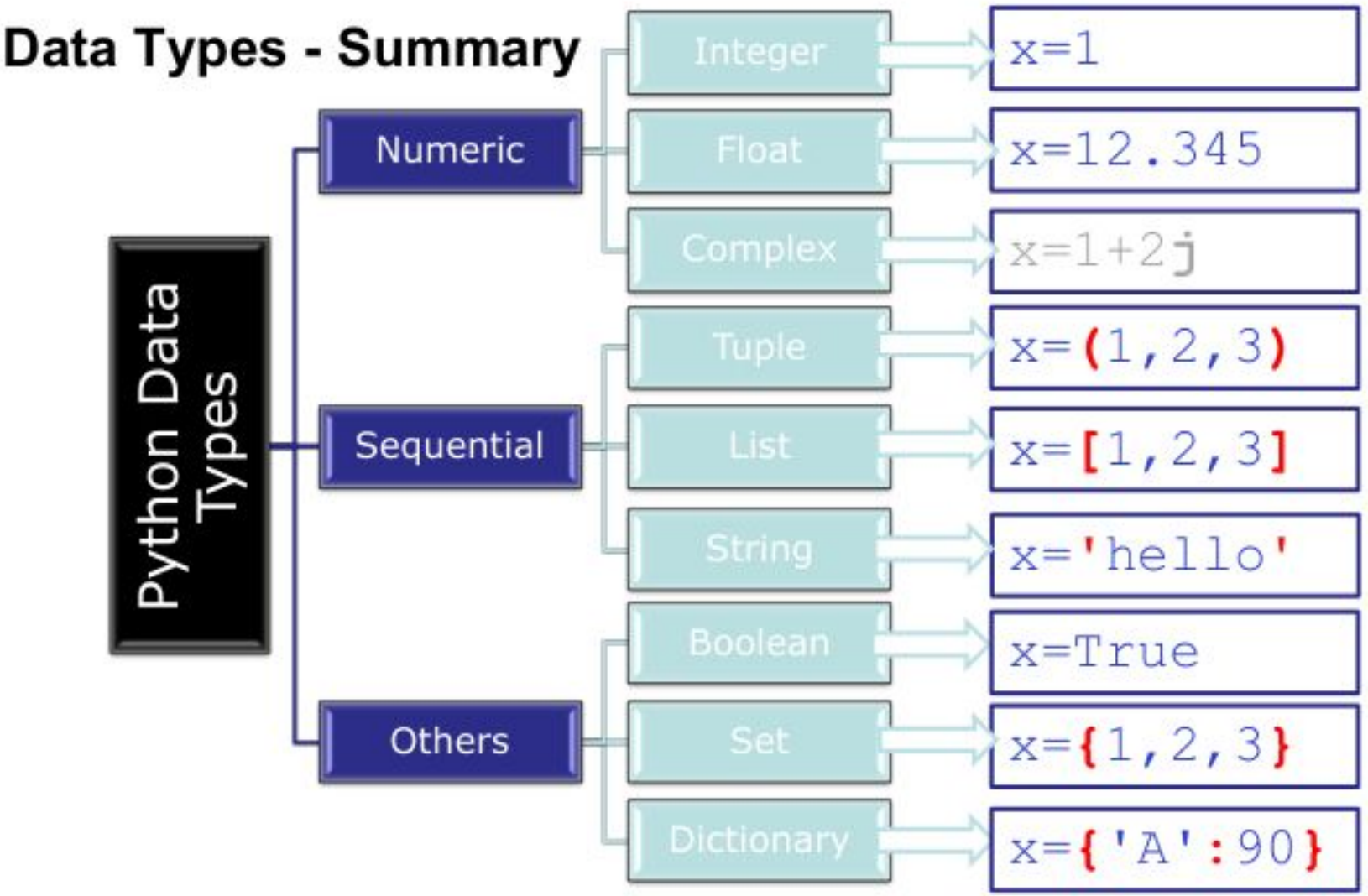
Data Type - Dictionary

- **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
Python Data Types
Numeric
Sequential
Others
Integer
Float
Complex
Tuple
List
String
Boolean
Set
Dictionary
x=1
x=12.345
x=1+2j
x=(1,2,3)
x=[1,2,3]
x='hello'
x=True
x={1,2,3}
x={'A':90}

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

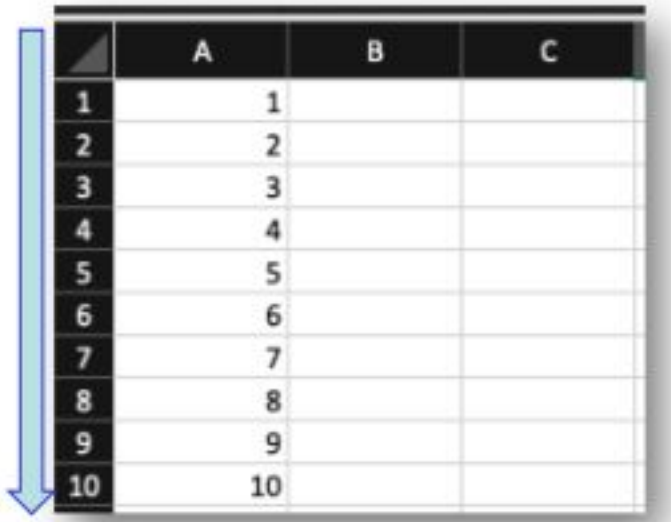
```
Dim speed As Integer
speed = 70

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

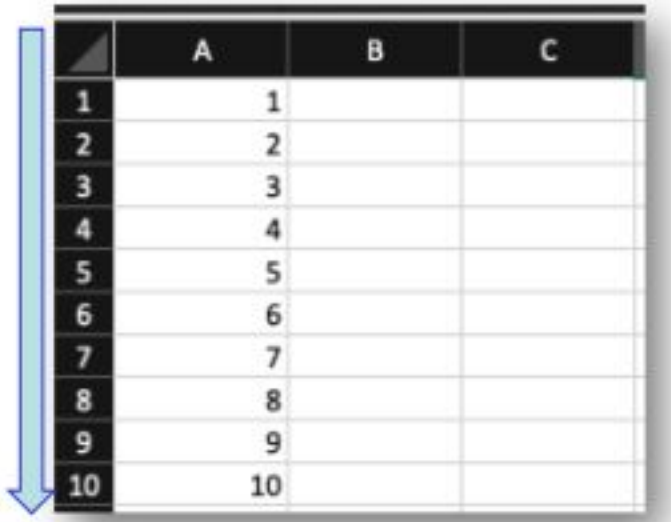
```
For i = 1 To 10
        Debug.Print (i)
Next i
```

Python

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

Loops – Using "for"

- Different ways to specify ranges



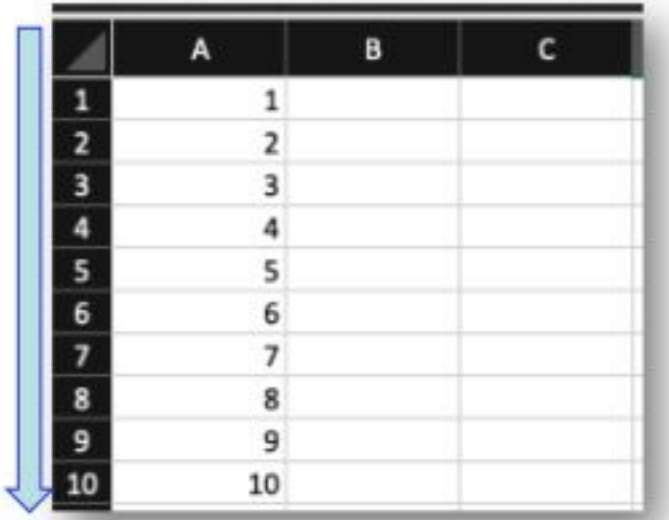
	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
[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



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

```
[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)
```

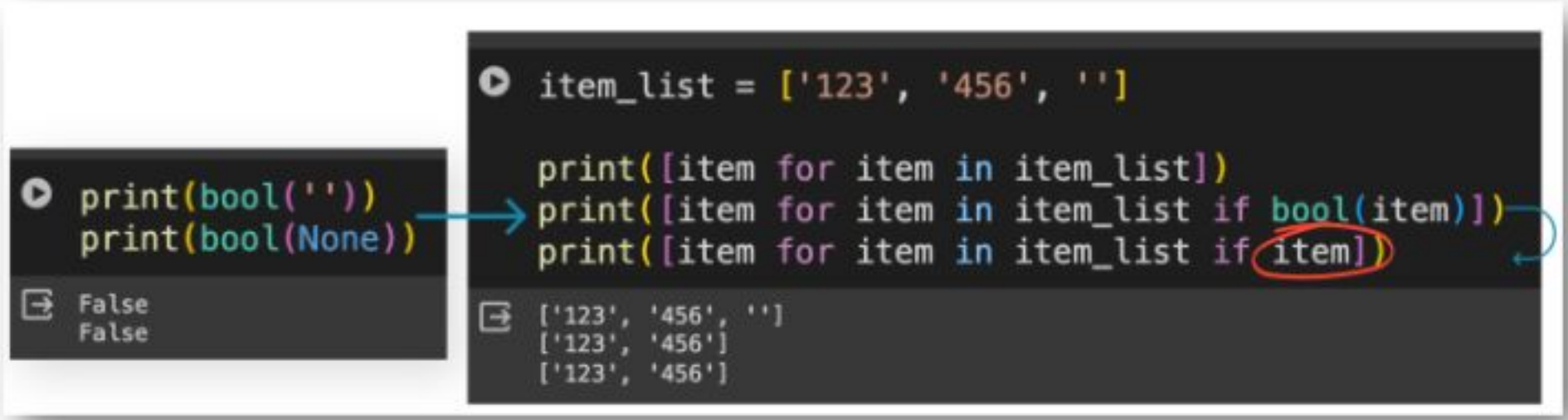
```
[0, 2, 4, 6, 8, 10]
```

```
# 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]
```

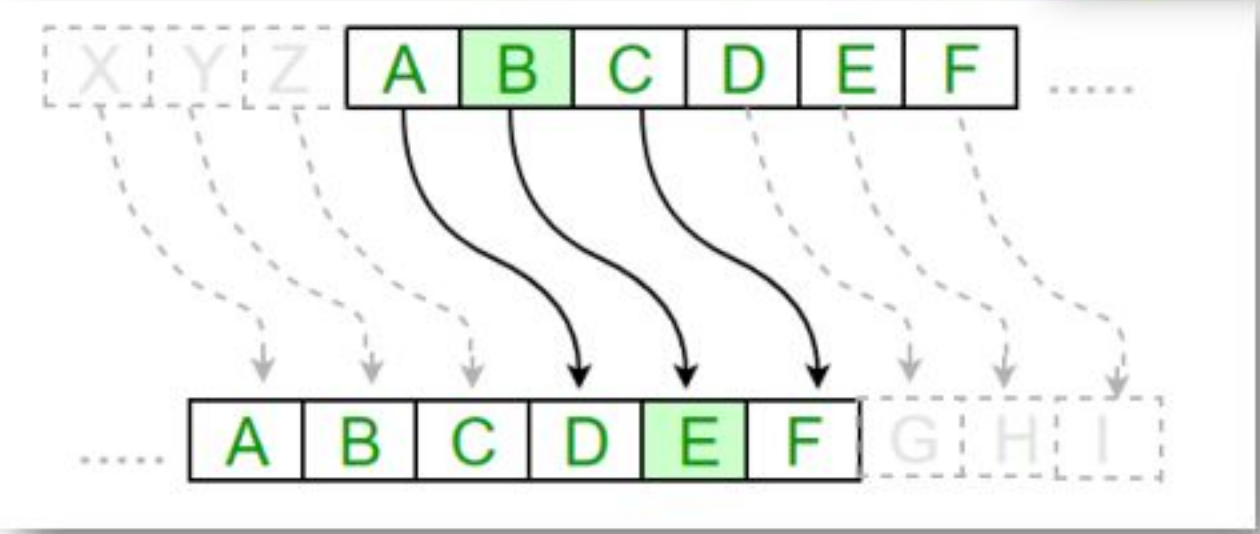
Loops – Using "for"

- List comprehension + Filtering



Practice 3 – Caesar Cipher

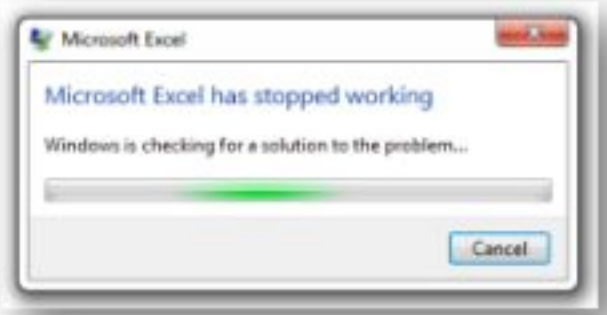
- **Encryption: Shift each letter by 3**
  - Reference: https://cryptii.com/pipes/caesar-cipher

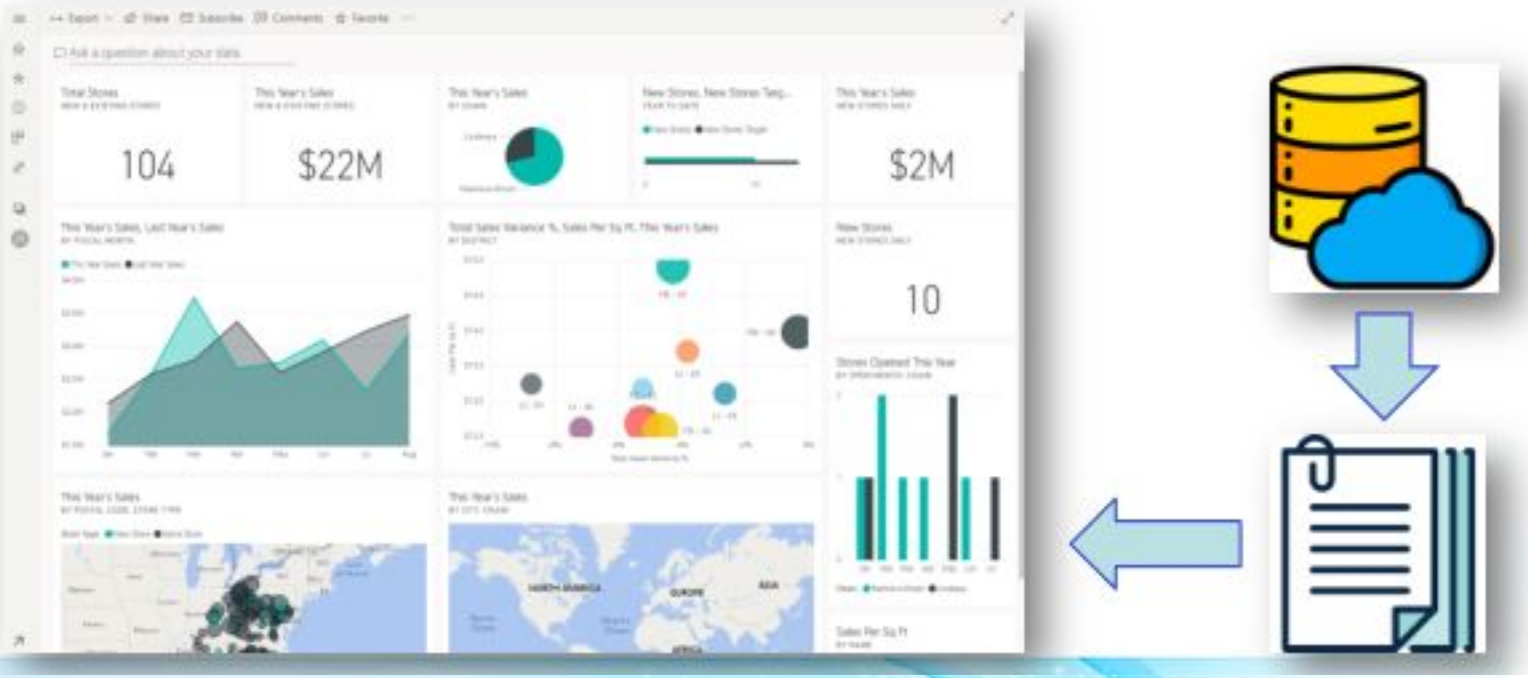


- **Starting notebook:**
  https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

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')

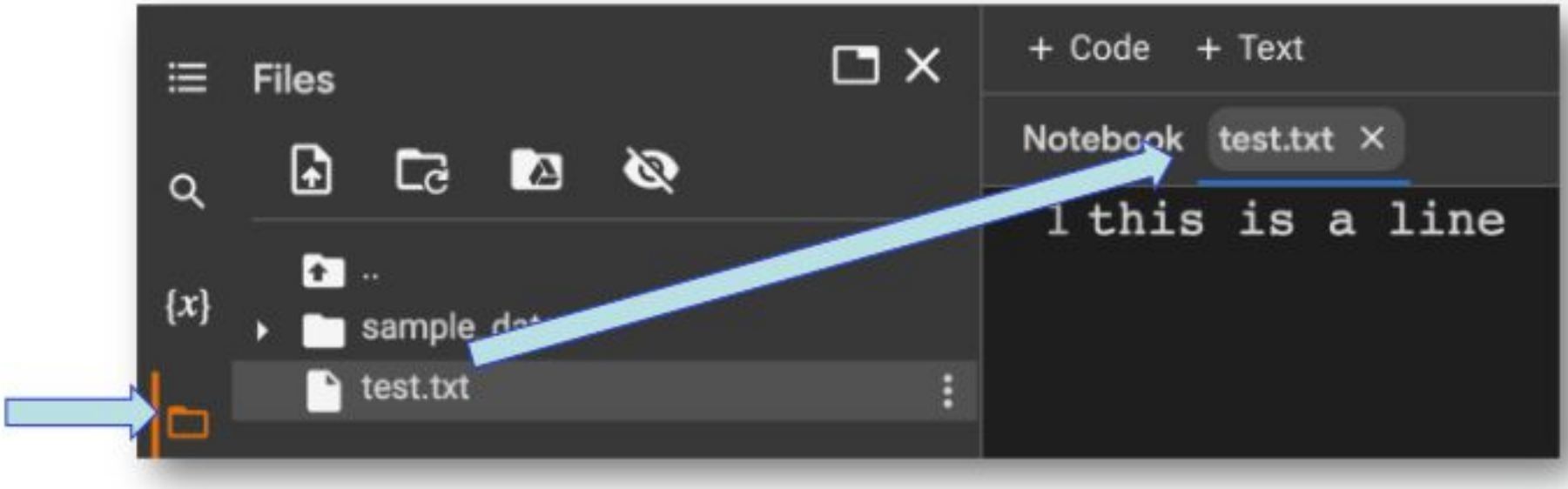
File Input / Output

- **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

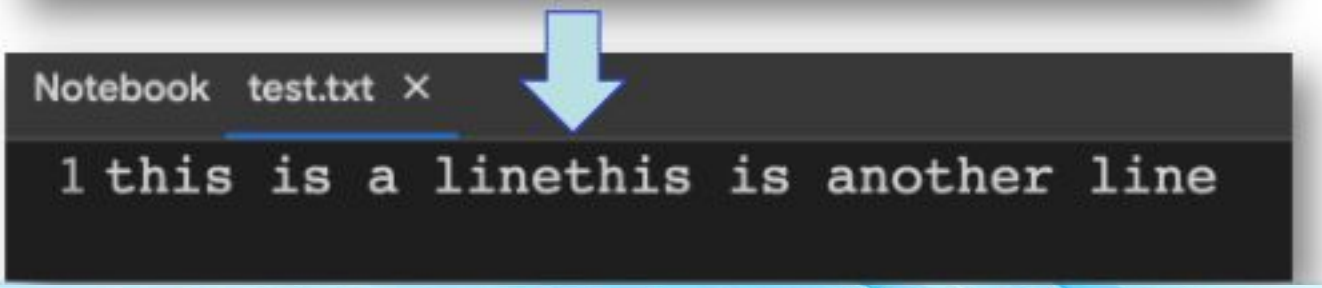
- **Writing a text file**
  - Viewing the result



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']
```

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

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

資料一線通 DATA.GOV.HK

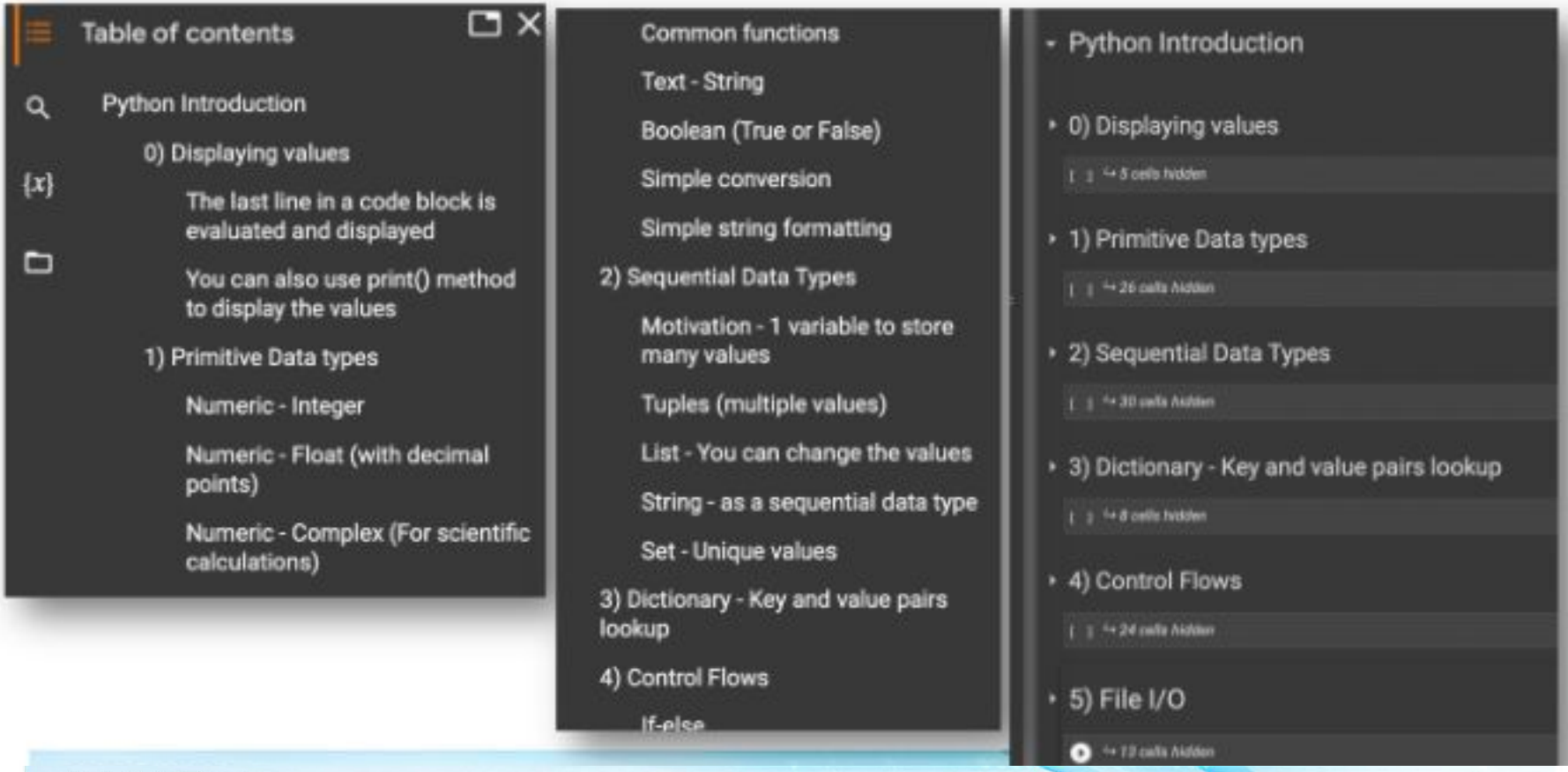
	A	B	C	D	E
1	Account	Expenditure Group	Subhead	Item	Actual($)
2	RECURRENT	PERSONNAL EMOLUMENTS	SALARIES	SALARIES	151,760,978
3	RECURRENT	PERSONNAL EMOLUMENTS	ALLOWANCES	OVERTIME	507,492
4	RECURRENT	PERSONNAL EMOLUMENTS	ALLOWANCES	ACTING ALLOWANCES	5,636,585
5	RECURRENT	PERSONNEL RELATED EXPENSES	MANDATORY 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	9,534,616
8	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	CLOTHES AND UNIFORMS	21,940
9	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	CLEANING MATERIALS	107,532
10	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	OFFICE STATIONERY AND MATERIALS	672,661
11	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	GENERAL PUBLICATIONS, PERIODICALS AND JOURNALS	123,429
12	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	OFFICE FURNITURE AND EQUIPMENT(INCLUDING RENTAL)	445,709
13	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	PAPERS	53,221
14	RECURRENT	DEPARTMENAL EXPENSES	STORES AND EQUIPMENT	INFORMATION AND COMMUNICATIONS TECHNOLOGY	3,299,560
15	RECURRENT	DEPARTMENAL 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

Notebook Summary

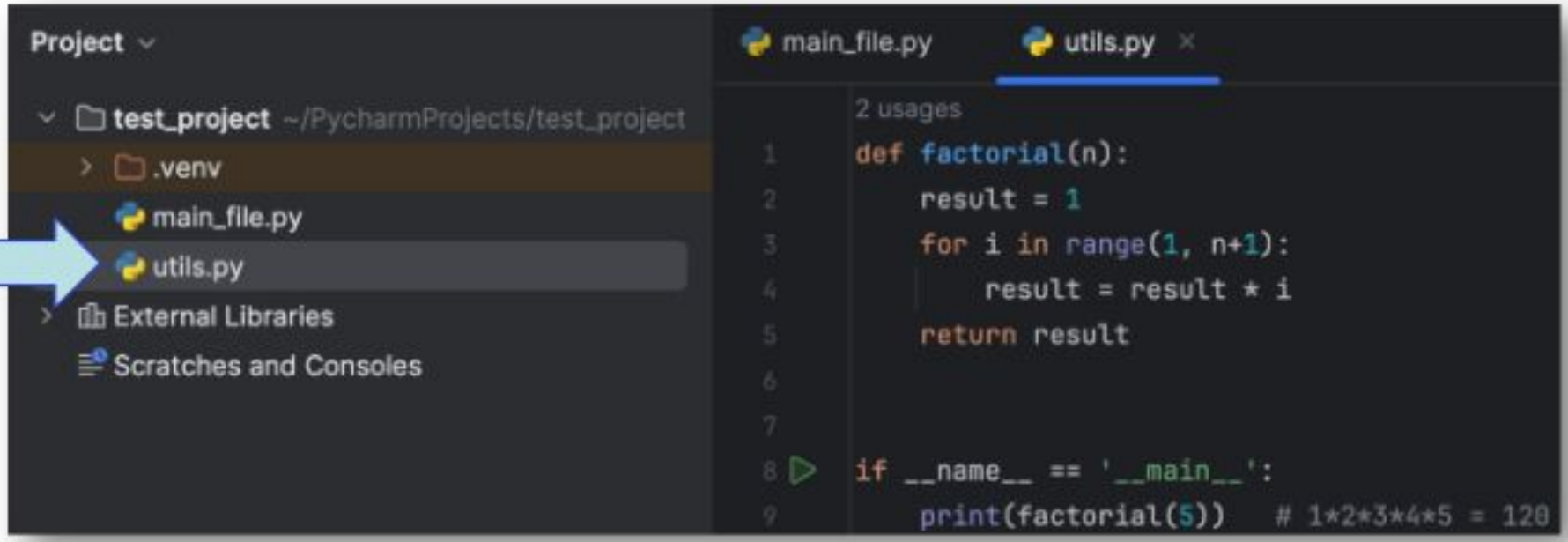
https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

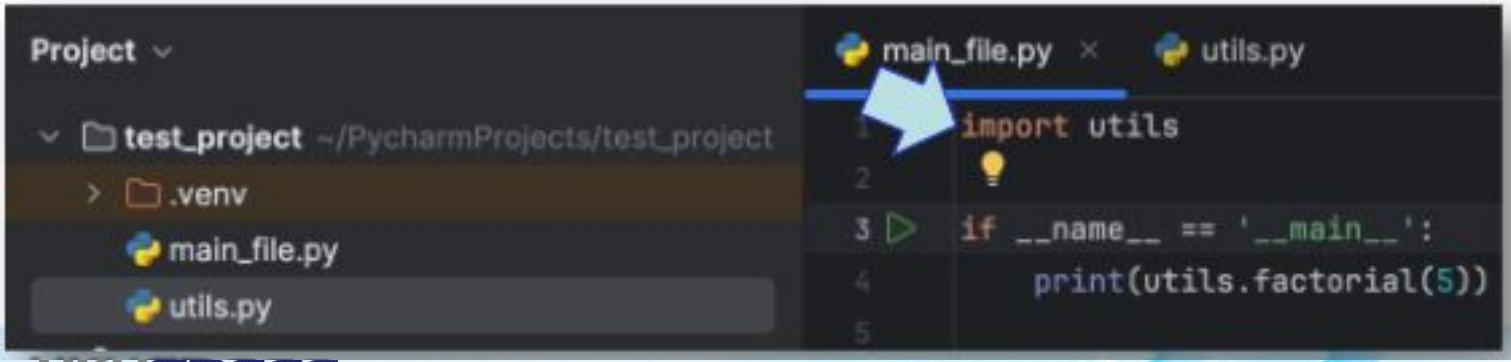


Practice 4 - Review

- 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”?

Importing Modules





Python Architecture (Python runtime + Packages)

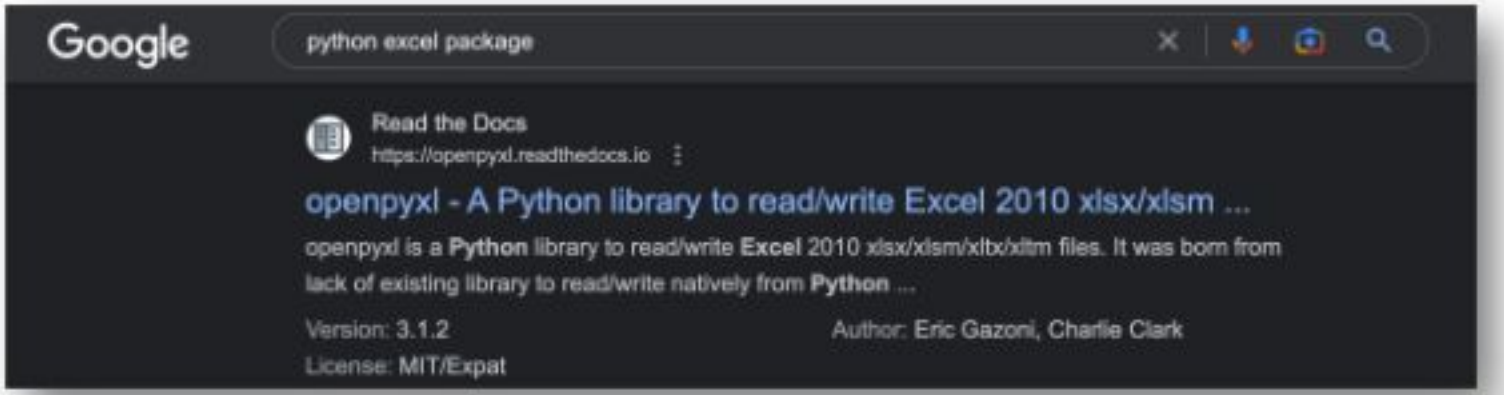
- Packages
  - Built-in / published codes which can be imported to use
  - e.g., **datetime** module ➔ **date** class ➔ **strftime** function

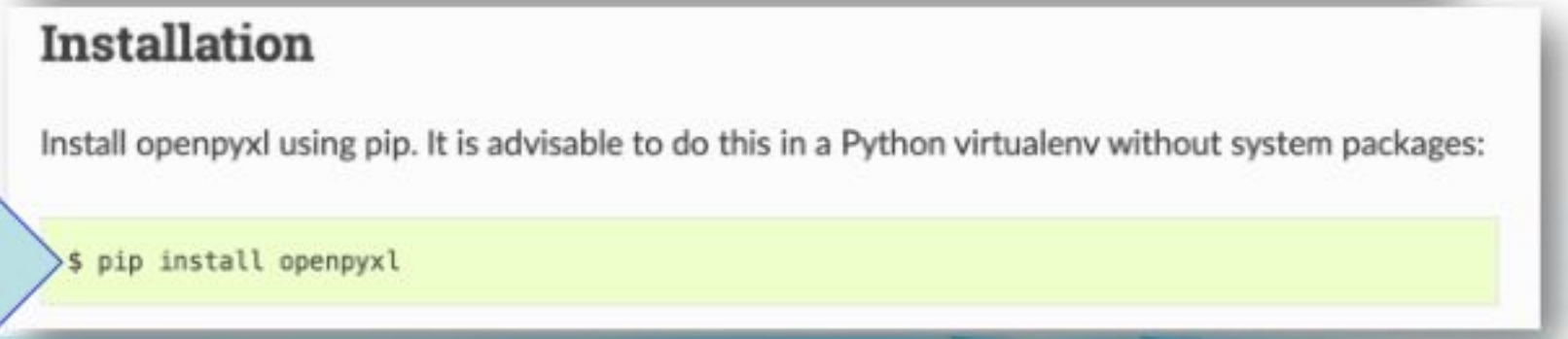
```
import datetime

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)
```

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





Packages - Installation

- In Jupyter Notebooks
  - Add "!" ➔ Install system-wide
  - Add "%" ➔ Install within notebook environment

```
!pip install pandas
```

```
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
```

```
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