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 / 17 May 2025

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

Table Of Contents

Introduction to business process automation

Business Process automation with VBA

Business process automation with Python

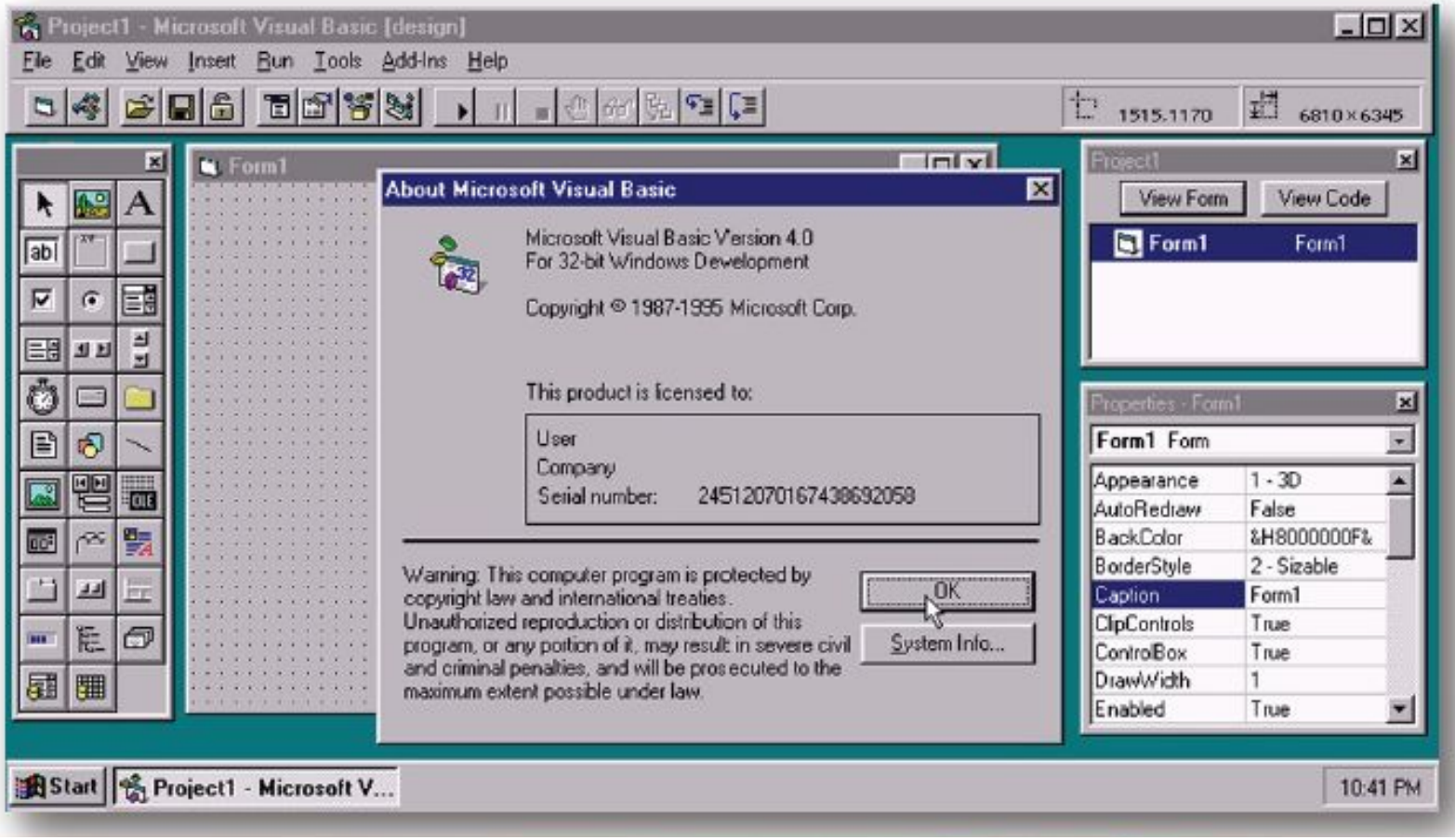
Introduction to project management for business process automation

Development and implementation of business process automation

Final Group Presentation

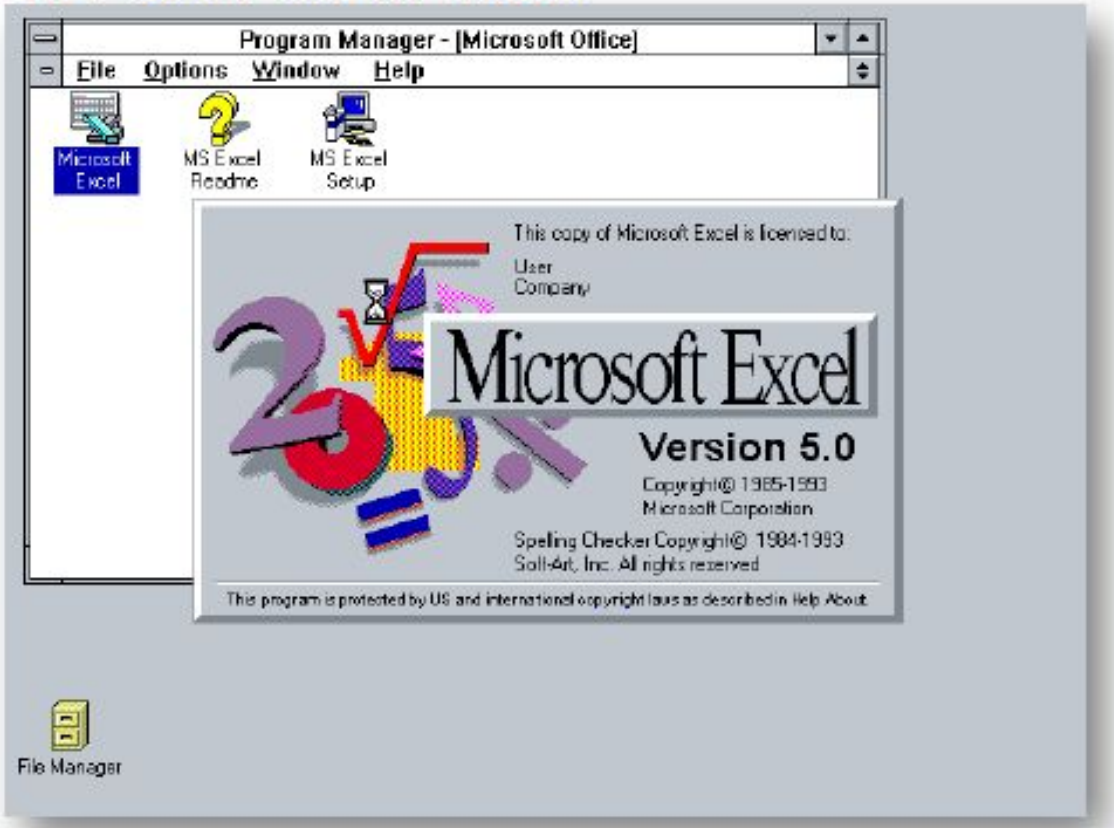
VBA - Visual Basic

- VB was “The” Microsoft way to write programs



VBA - Visual Basic for Applications

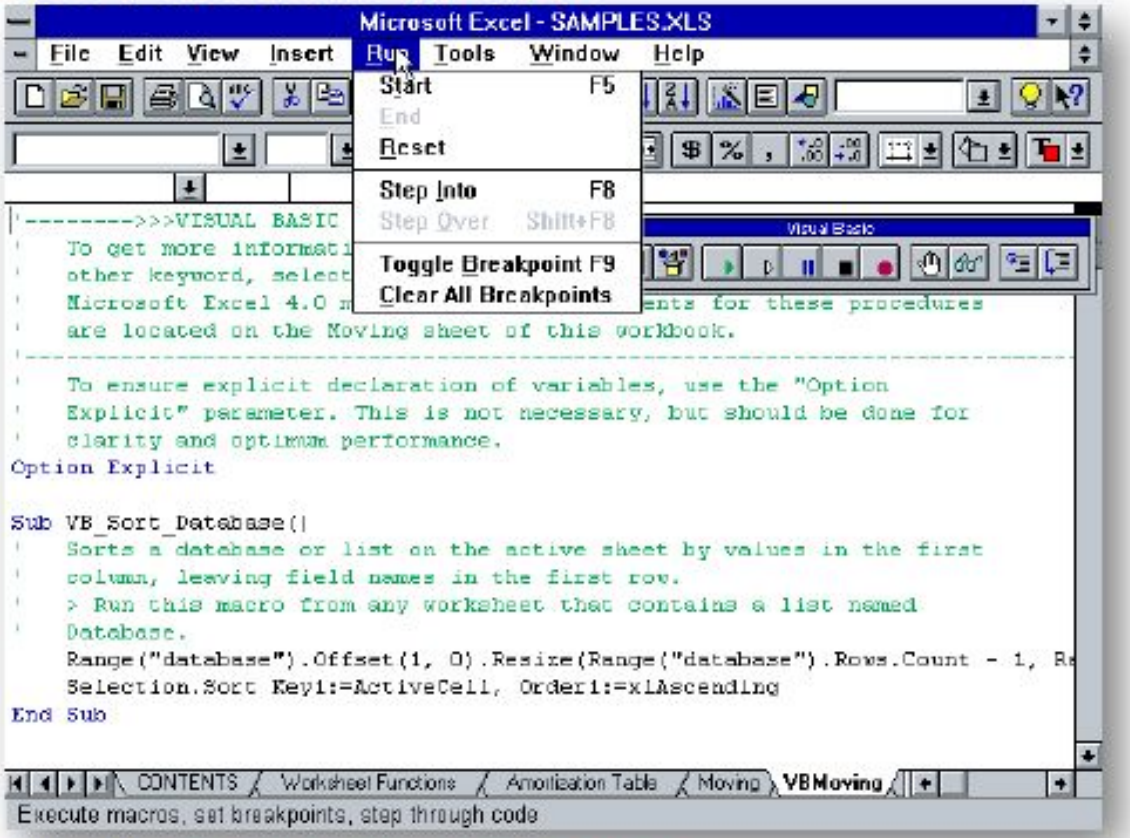
- Introduced to Excel 5.0 in 1993



1. *Microsoft Excel 5.x*. WinWorld. https://winworldpc.com/product/microsoft-excel/5x

VBA - Visual Basic for Applications

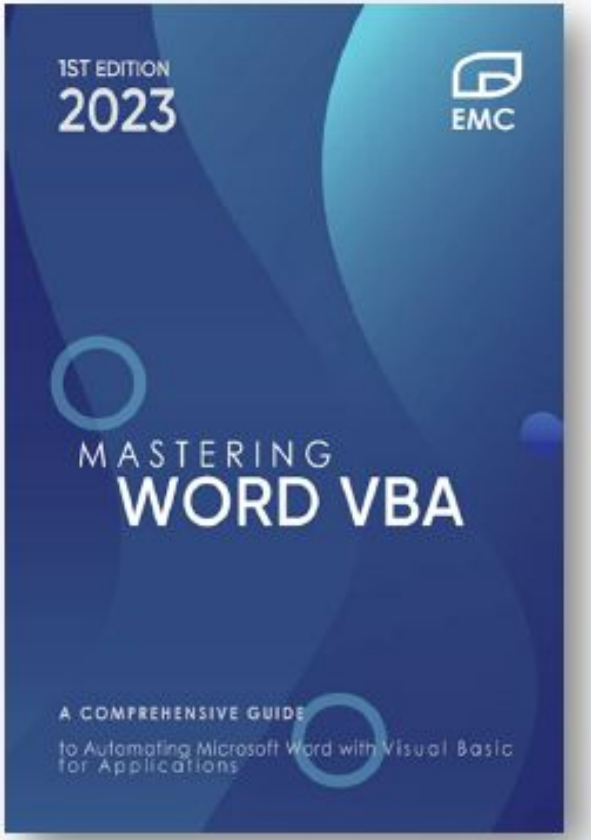
- Introduced to Excel 5.0 in 1993



1. *Microsoft Excel 5.x*. WinWorld. https://winworldpc.com/product/microsoft-excel/5x

VBA - Visual Basic for Applications

- Gradually extended to Microsoft Office
- VB classic ➔ ended in 2008
- VB.NET ➔ evolution ended in 2020[1]
- VBA ➔ still supported (e.g., M365)



1. Team, N. (2020, March 11). *Visual Basic support planned for .NET 5.0*. Visual Basic Blog. https://devblogs.microsoft.com/vbteam/visual-basic-support-planned-for-net-5-0/

Business Process Automation with VBA

VBA – Why?

- Automation
- User interaction
- Widely used in the industry

1. O. (2022, June 8). *Getting started with VBA in Office*. Getting Started With VBA in Office | Microsoft Learn. https://learn.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

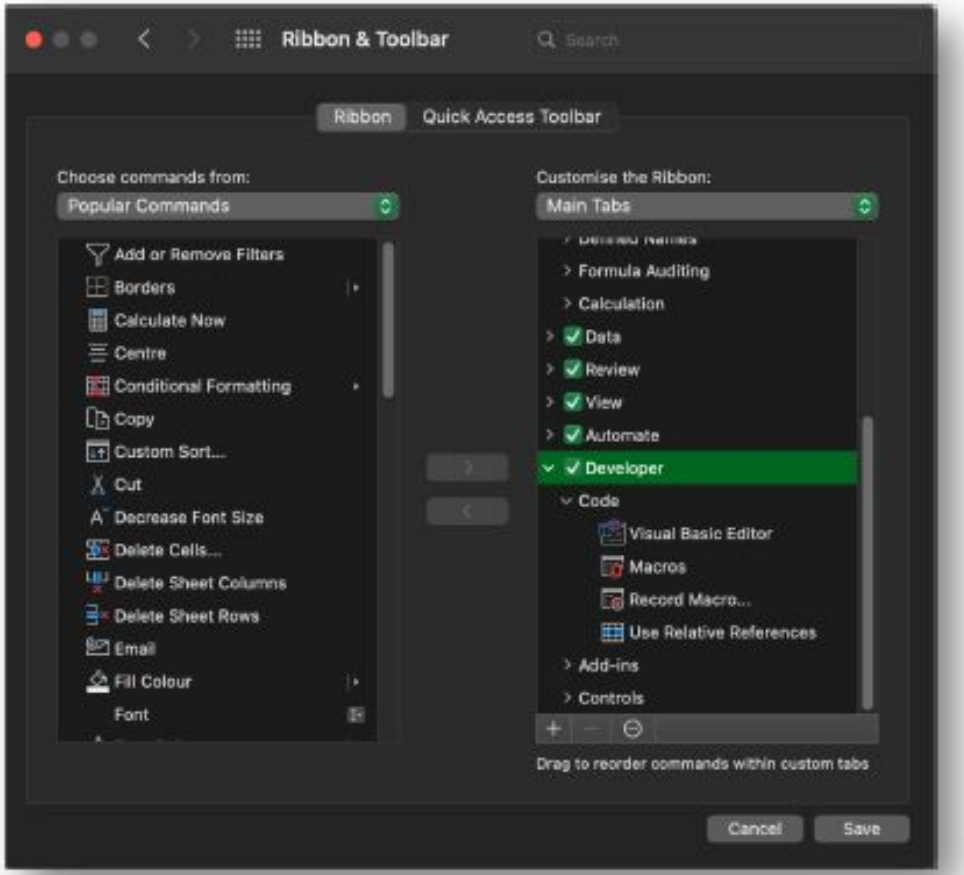
VBA - Automation

- Macros ➔ Microsoft's response to automation
- Naming:
  - **Micro**-instruction vs **Macro**-instruction
  - 1 key to perform series of commands / actions

1. *Why is Excel VBA called "Macros"?* Why Is Excel VBA Called "Macros"? https://www.excelforum.com/the-water-cooler/791161-why-is-excel-vba-called-macros.html

Macros – Where?

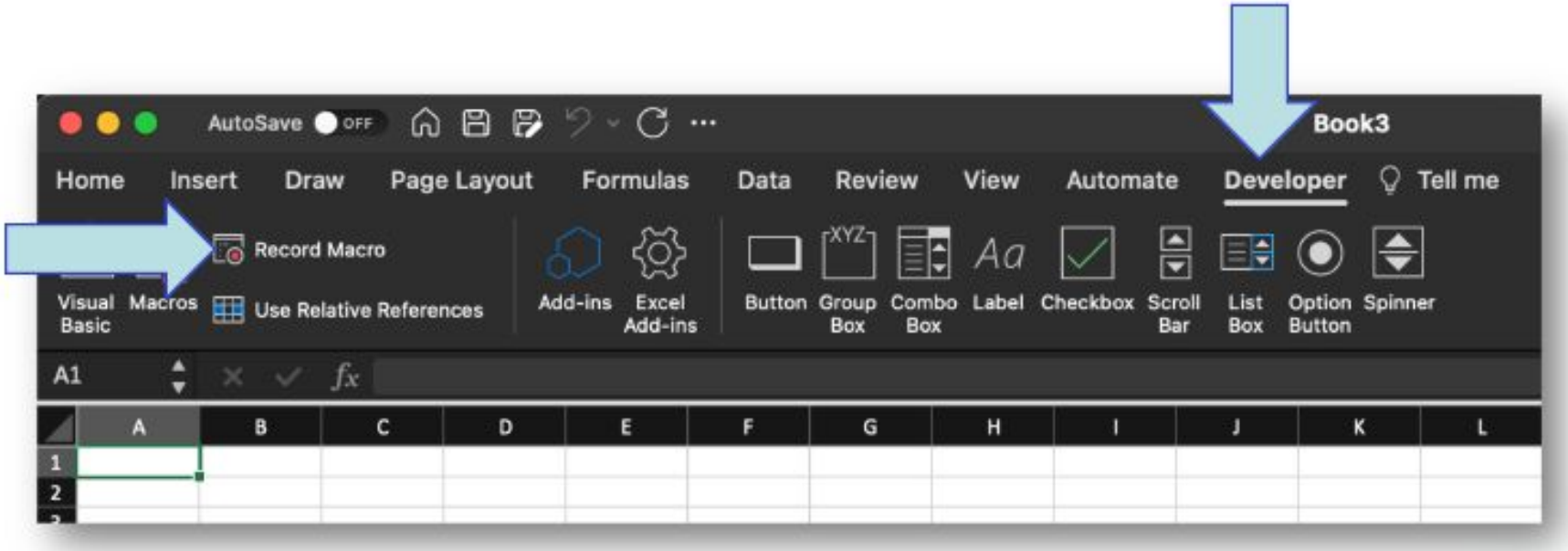
- "Developer" tab
  - Hidden by default
  - File
    - ➔ Options
    - ➔ Customize Ribbon
  - Turn on "Developer"



1. *Show the Developer tab - Microsoft Support*. Show the Developer Tab - Microsoft Support. https://support.microsoft.com/en-us/office/show-the-developer-tab-e1192344-5e56-4d45-931b-e5fd9bea2d45

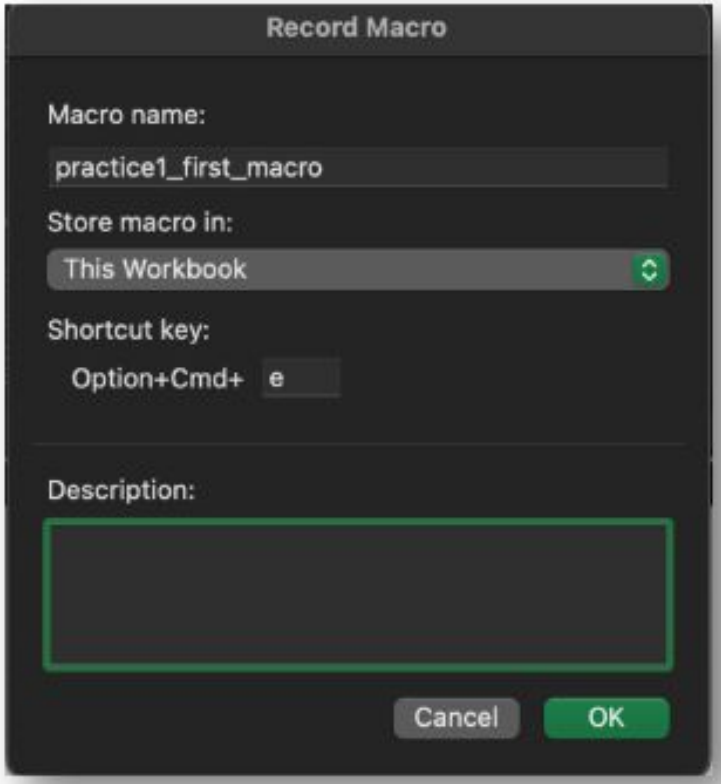
Practice 1 - Your first macro

- Click “Record Macro”



Practice 1 - Your first macro

- Type in a name (use "_" instead of spaces)
- Set a shortcut key
- Click OK

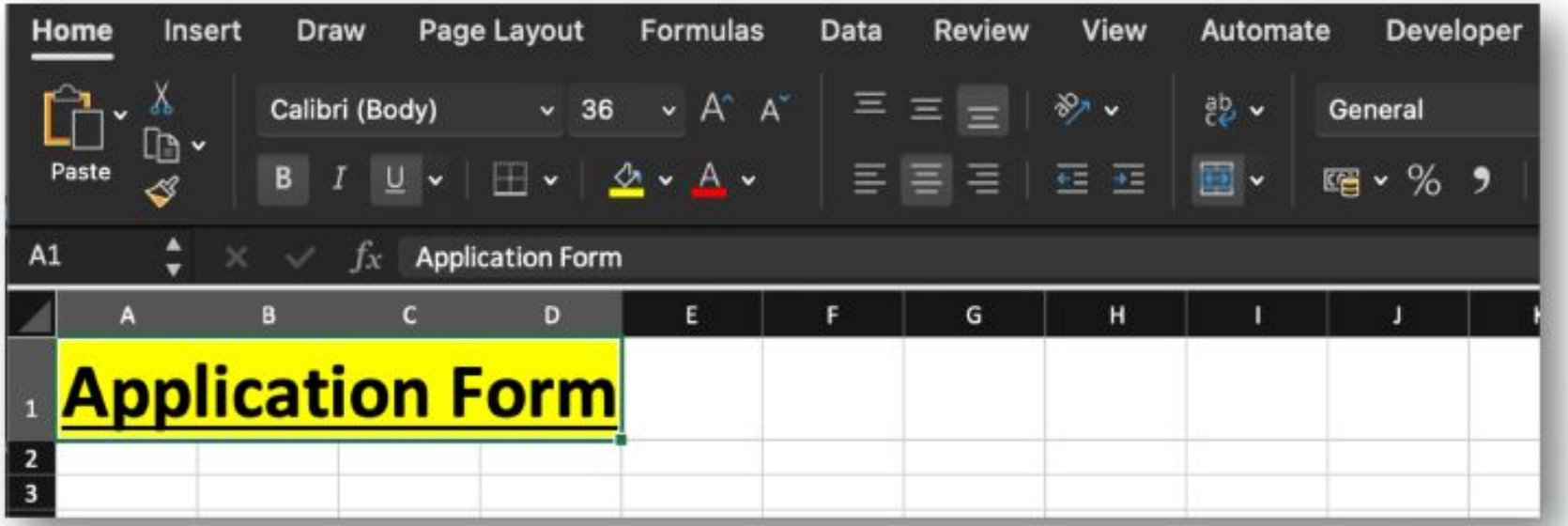


Practice 1 - Your first macro

- The recording has started
- Add a new sheet with the “+” at the bottom-left

Practice 1 - Your first macro

- Go back to “Home” tab
- Select Cells A1 to D1 ➔ Merge
- Enter “Application Form” at the top-left
- Style it: e.g., Font size + Bold + Underline

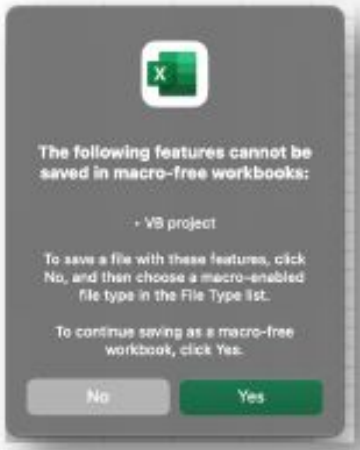


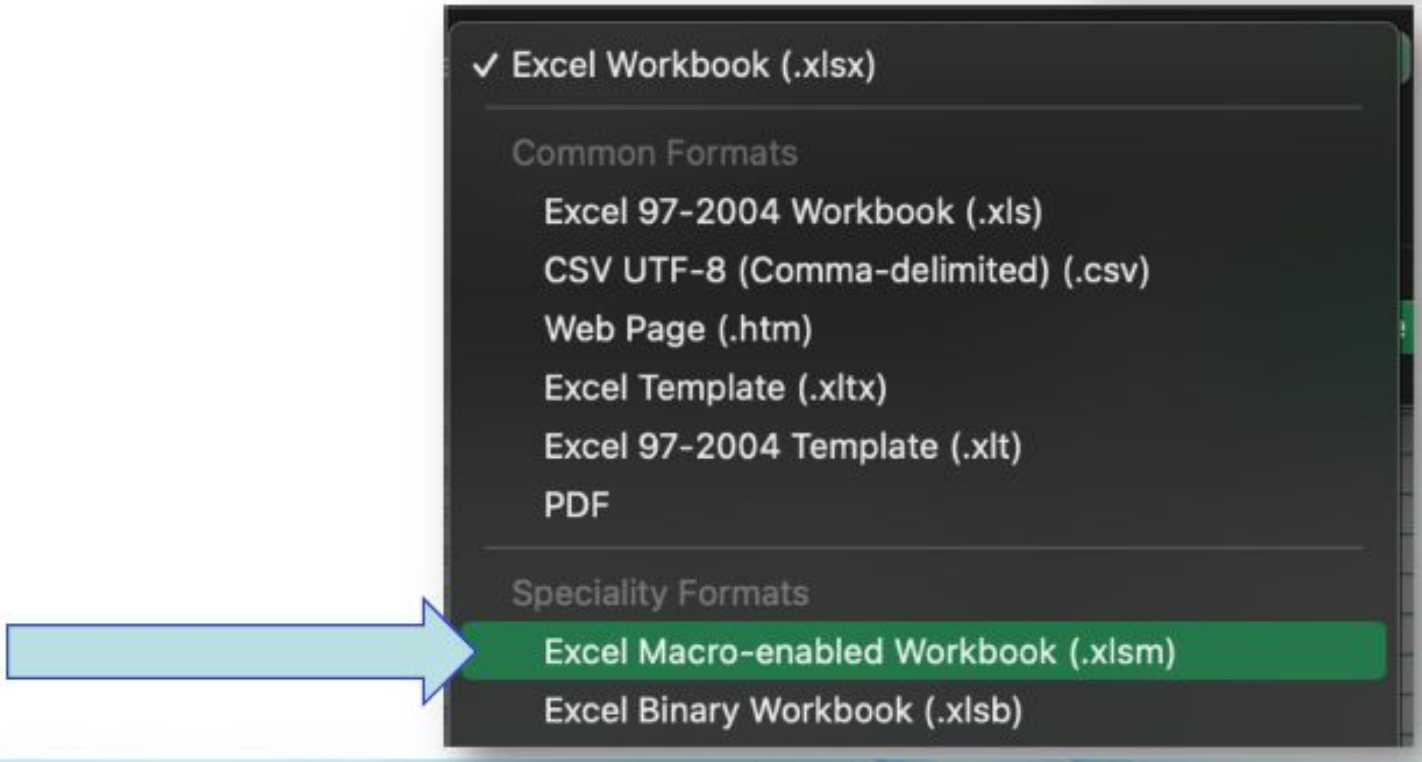
Practice 1 - Your first macro

- Go back to “Developer” tab
- Hit “Stop Recording”

Practice 1 - Your first macro

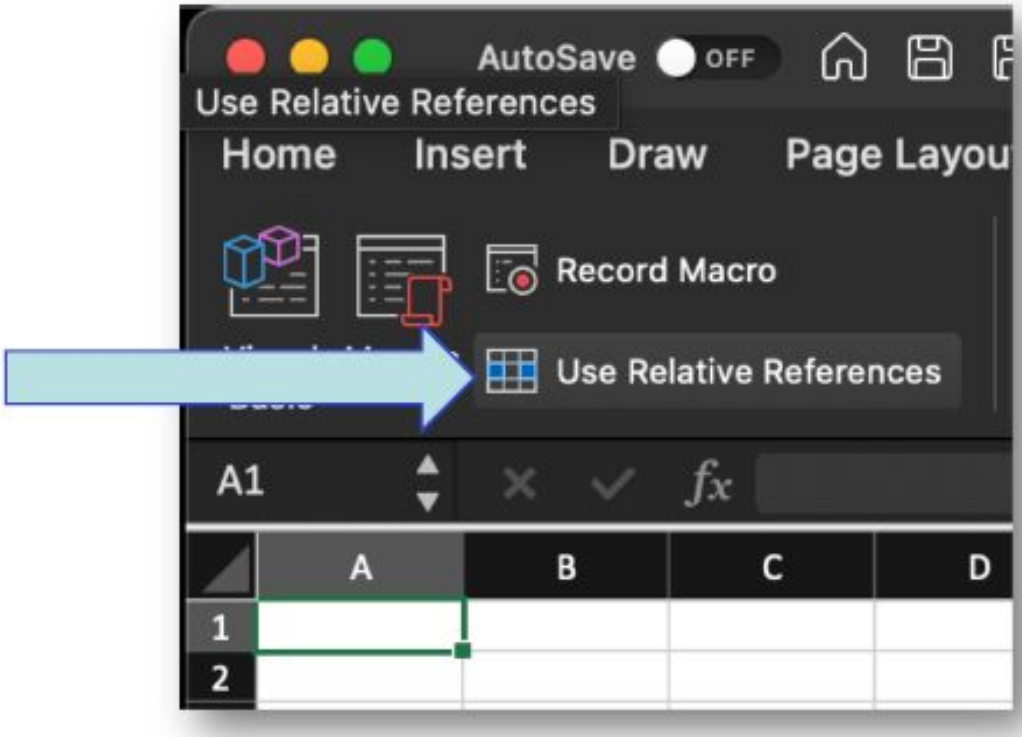
- Save as .xlsm to preserve your macros!





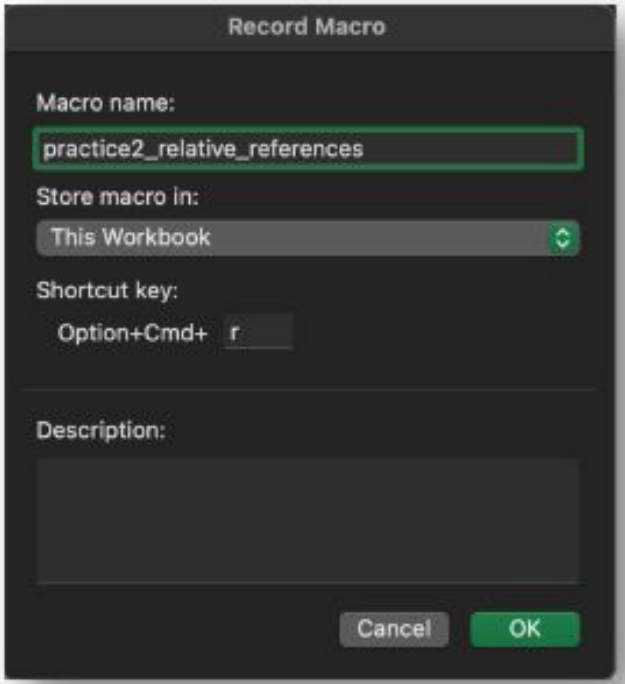
Relative Reference

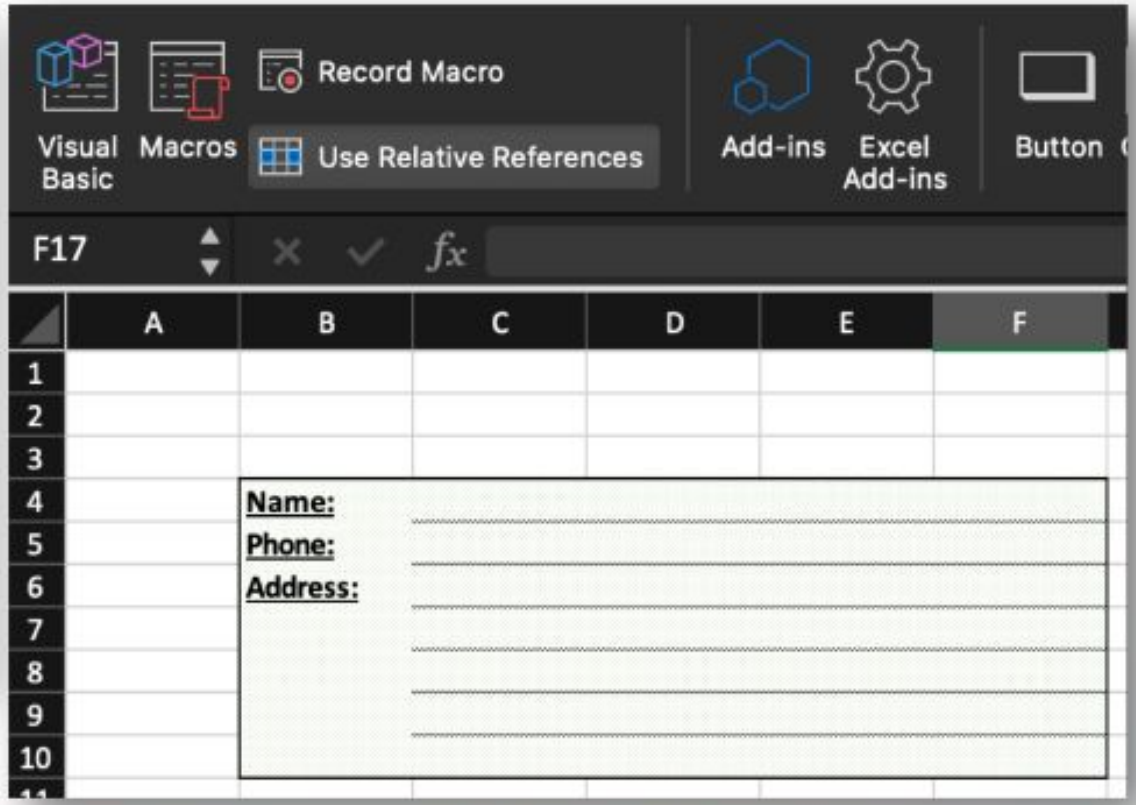
- Record and execute based on “current selection”
- Click BEFORE “Record Macro”



Practice 2 – Relative References

- Create a new macro to generate below



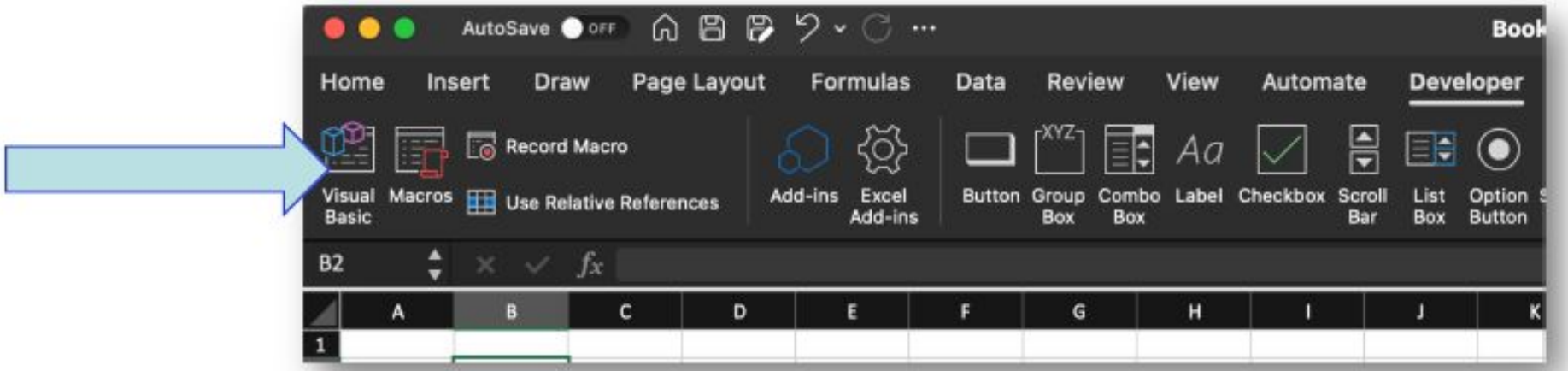


VBA - Why not just use macros?

- Macros ➔ great for repetition
- Need ➔ More flexibility and interactivity
- Solution ➔ Dig into the codes

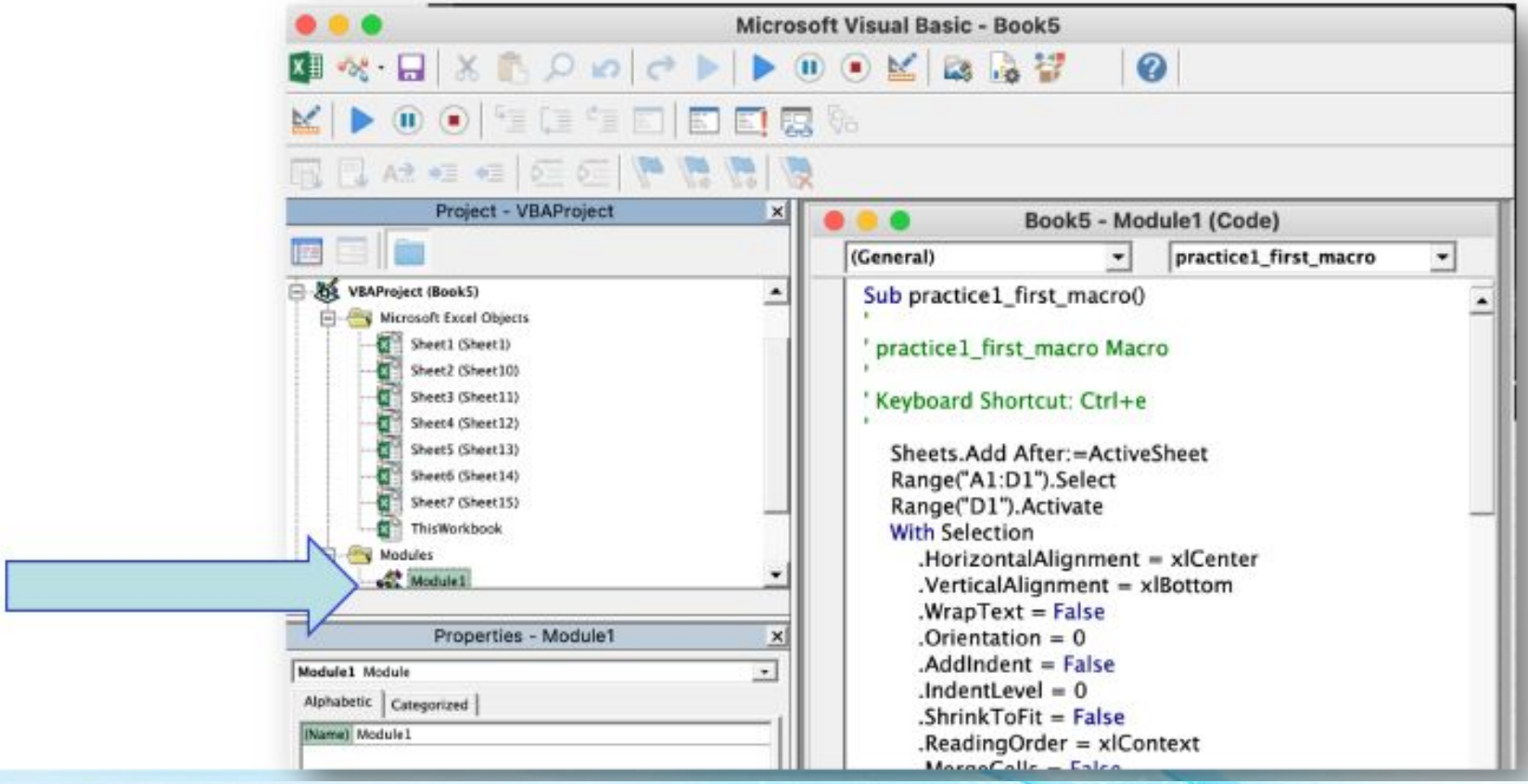
Entering Visual Basic Mode

- Developer tab ➔ Visual Basic



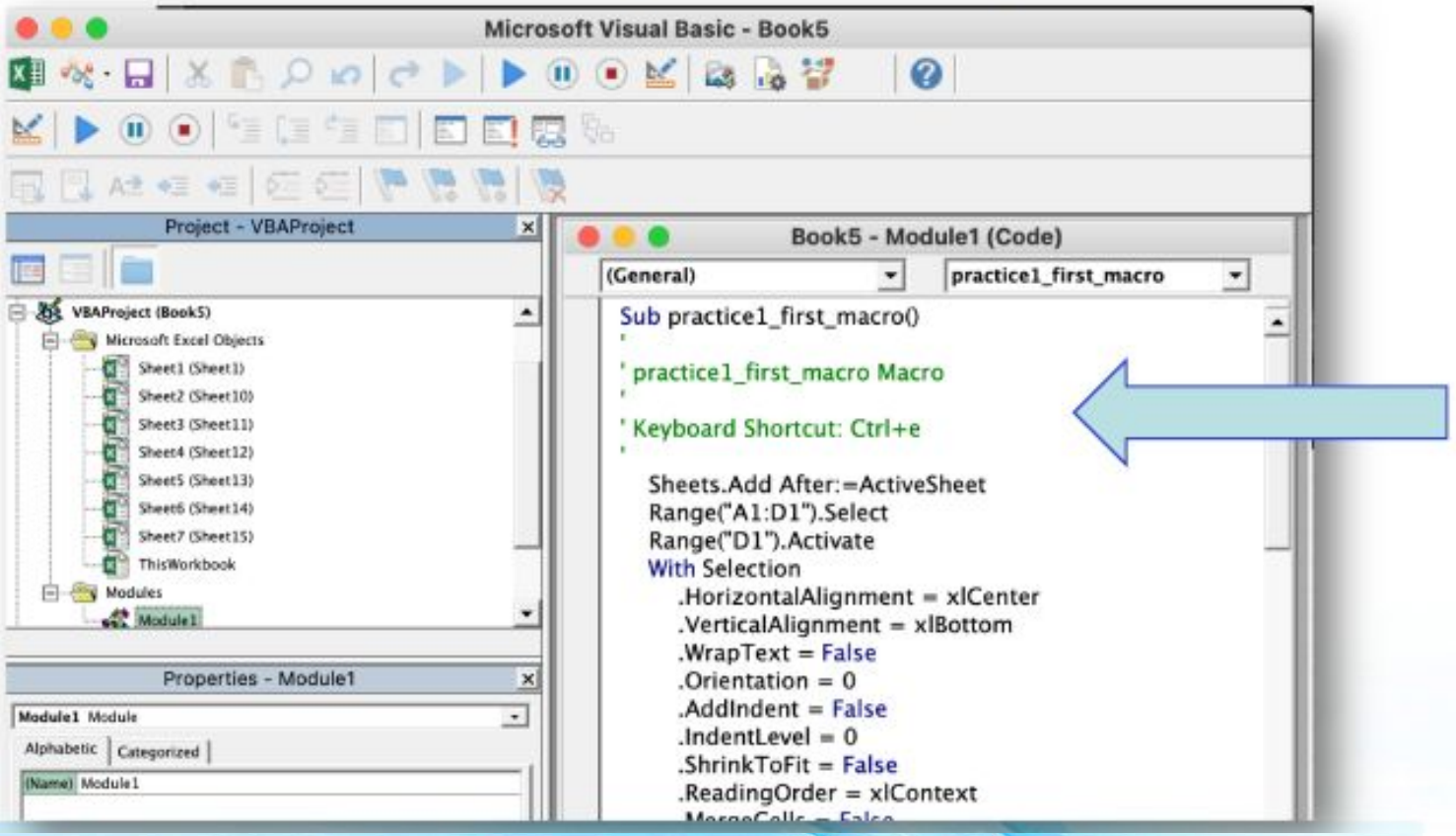
Entering Visual Basic Mode

- Integrated Development Environment (IDE)
- Modules = where codes are usually stored



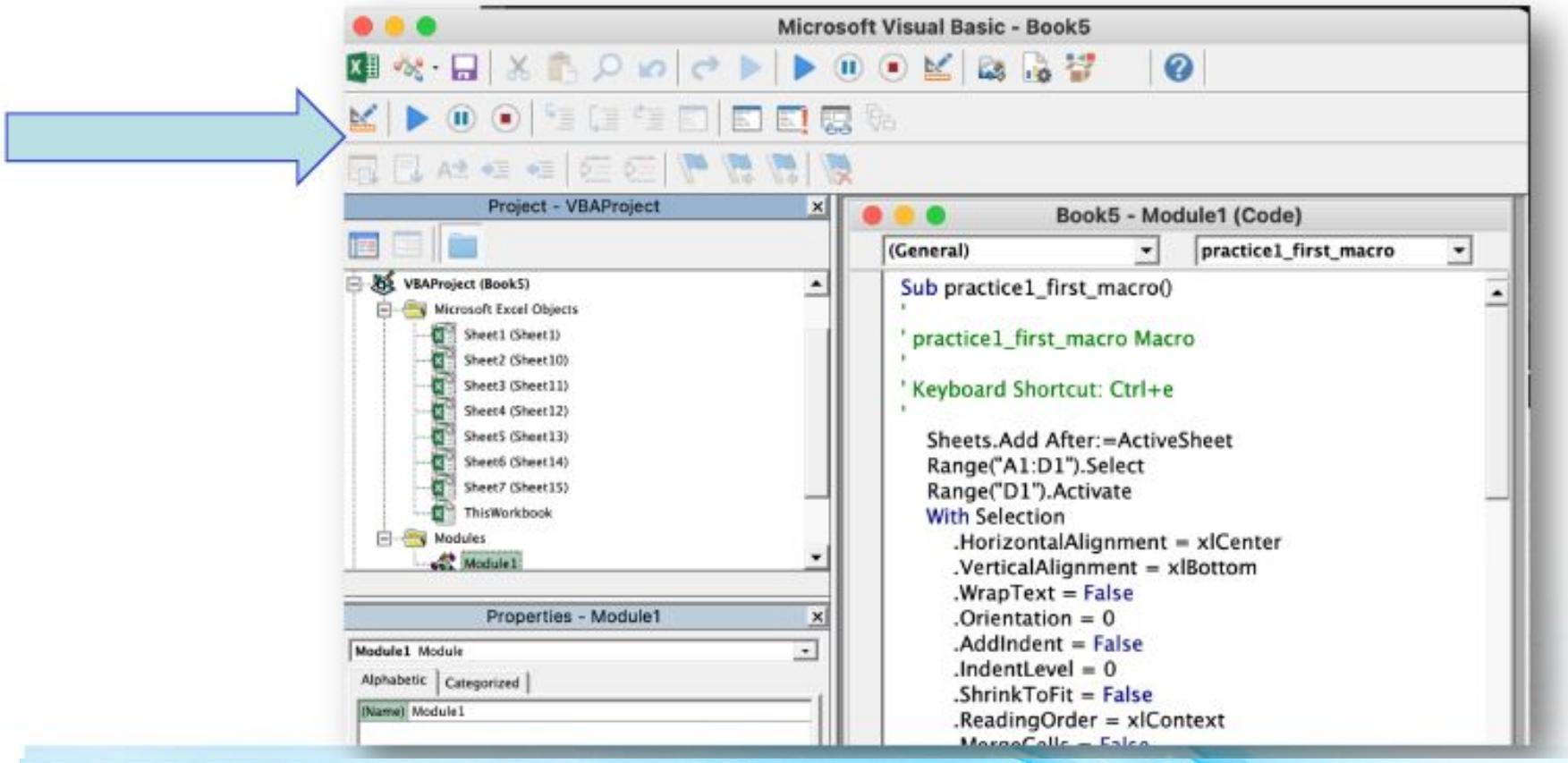
Entering Visual Basic Mode

- Main editor on the right
- Modules ➔ 1 or more “Sub” (sub-routines) or “Function”



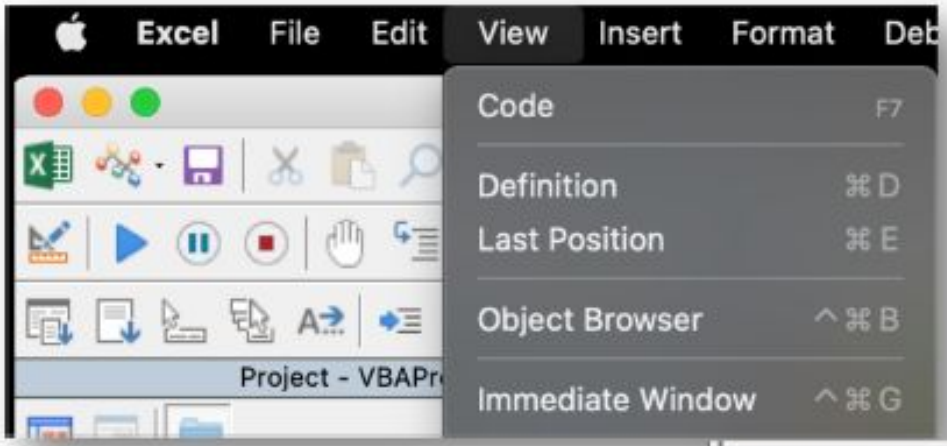
Entering Visual Basic Mode

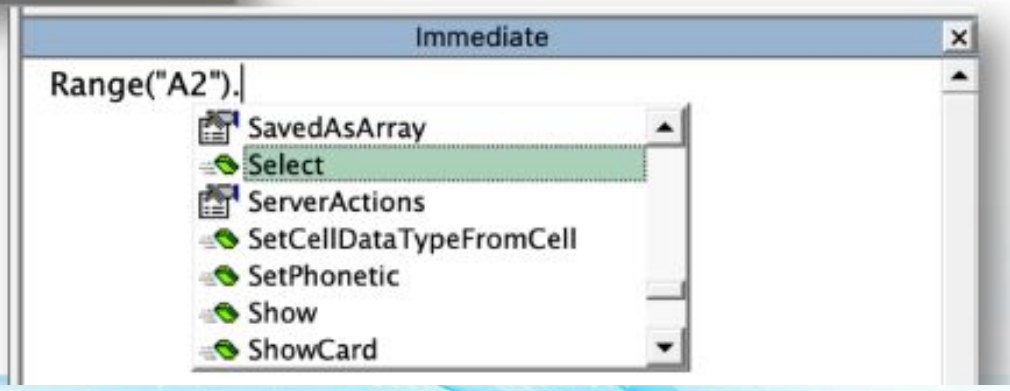
- Debugging toolbar at the top



Immediate Window

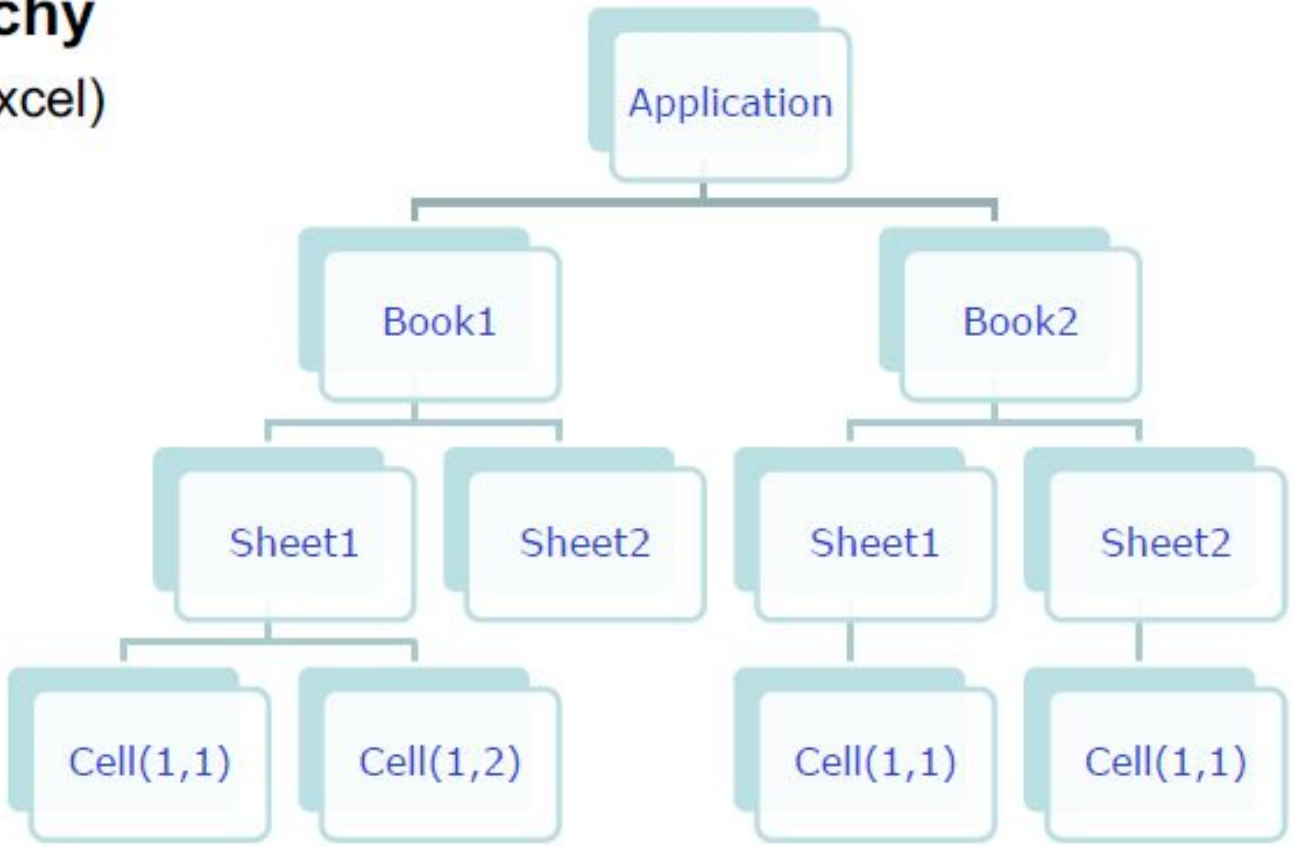
- View ➔ Immediate Window
- Useful for trying out codes (Thanks to Auto-complete)





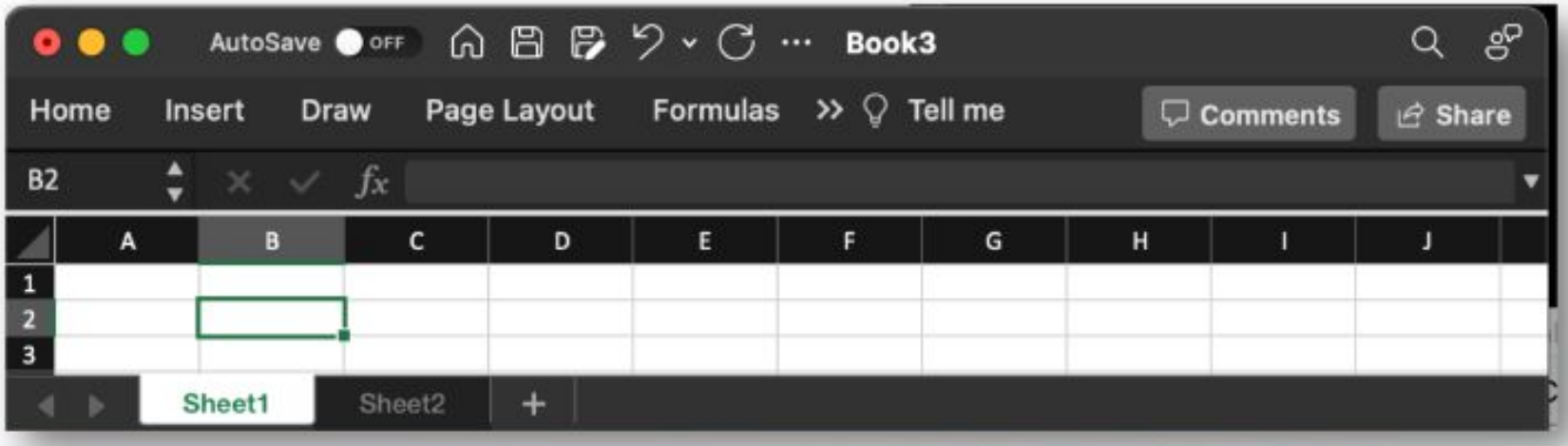
Object hierarchy

- Application (Excel)
- Workbooks
- Worksheets
- Cells



Object reference – Call something by name

- English: This man / that girl / “Mr. Chan”
- VBA
  - Application ➔ Excel.Application / Outlook.Application / etc
  - Workbooks("Book1")
  - Worksheets("Sheet1") / Worksheets(1)
  - Cells(2,3)
  - Range("B2")



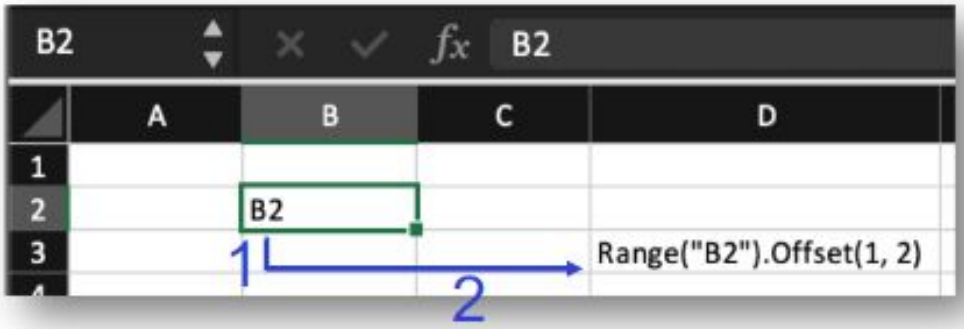
Object reference - Implied naming

- English: Here / Today
- VBA
  - ActiveWorkbook
  - ActiveSheet
  - Selection
    - If you write:
      - Range("A1:D1").Select
    - From that moment onwards:
      - Selection ➔ Same as Range("A1:D1")

* ThisWorkbook
  - This = Where the macro is stored
  - Active = Changeable by user

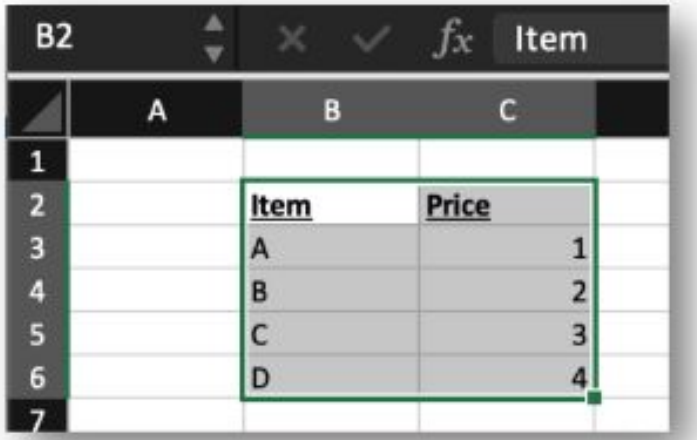
Object reference - Offset

- English: Is it that building? No, the one on the right
- VBA
  - Range("B2").Offset(1, 2)
  - Worksheets(ActiveSheet.Index + 1).Select

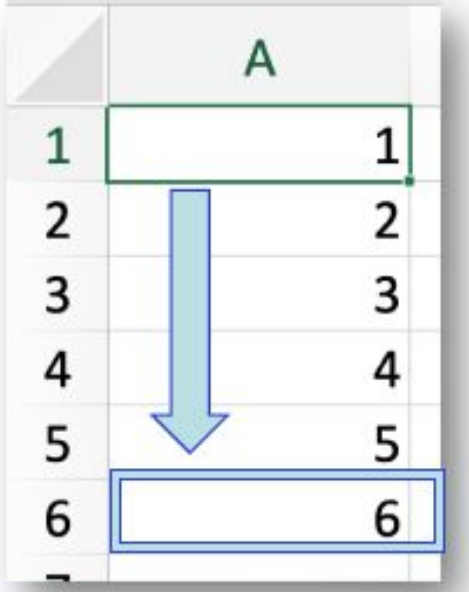


Object reference – Offset

- Region
  - Selection.CurrentRegion.Select



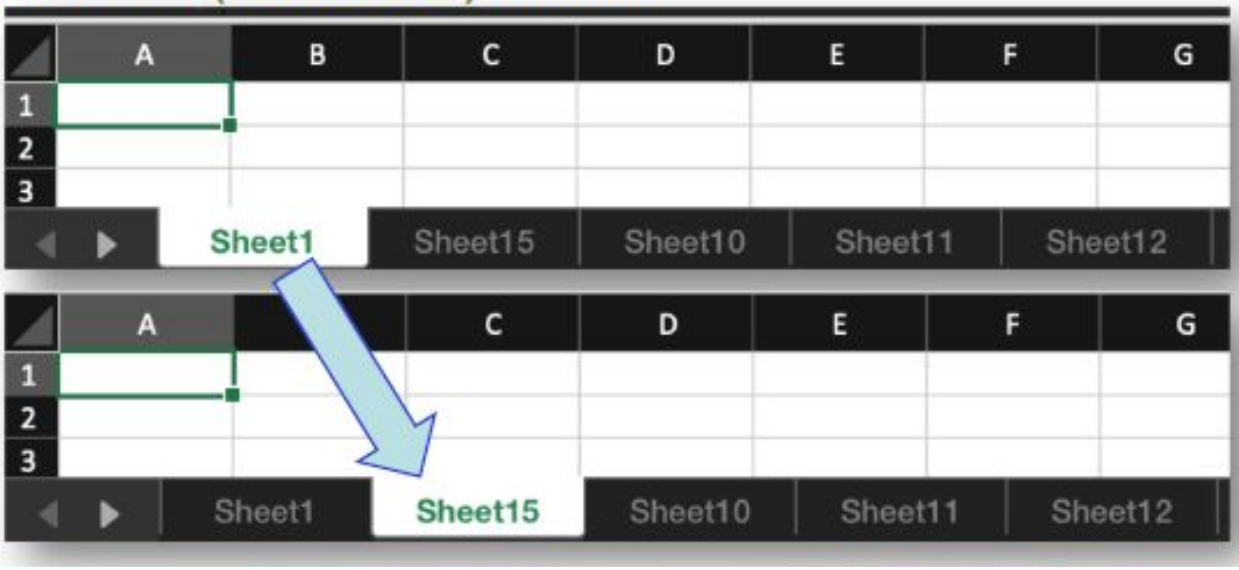
- Boundary
  - Range("A1").End(xlDown).Select




```
?Range("A1").End(xlDown).Row
 6
```

Object properties – Actions

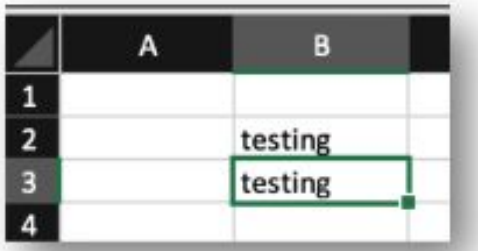
- English
  - Jackie runs / That person sings
- VBA (Actions are also known as Methods)
  - ActiveSheet.PrintOut
  - Workbooks("Book5").PrintPreview
  - Worksheets("Sheet15").Select



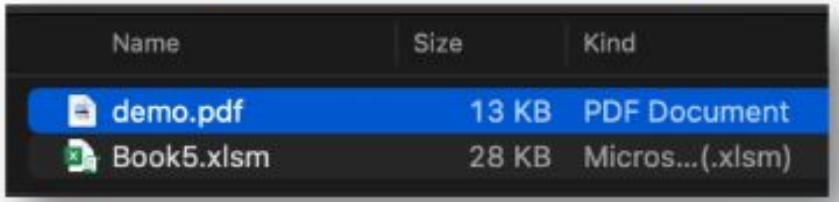
Object properties – More actions

- VBA – Some methods require parameters

- Copy & Paste
  - Range("B2").Copy Range("B3")

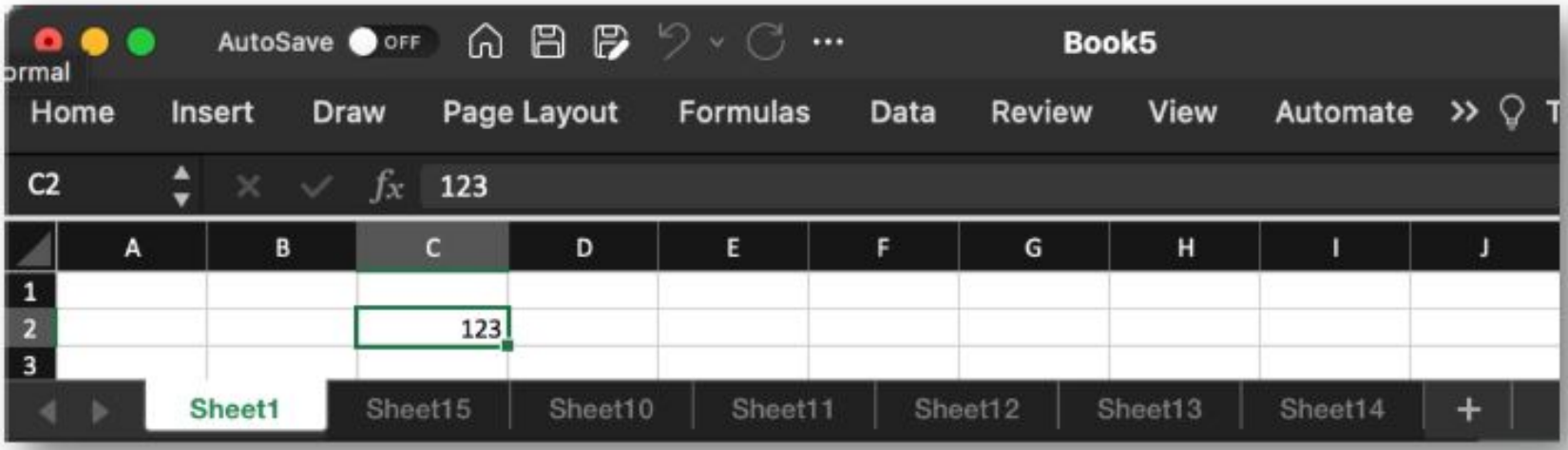


- Export to PDF
  - ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, Filename:="demo.pdf"



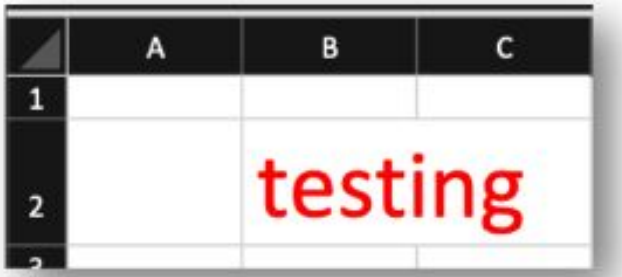
Object properties – Value

- English
  - HSBC ➔ Account 123-654321-012 ➔ Value ➔ $1000
- VBA
  - Workbooks("Book5").Worksheets("Sheet1").Cells(2,3).Value = 123



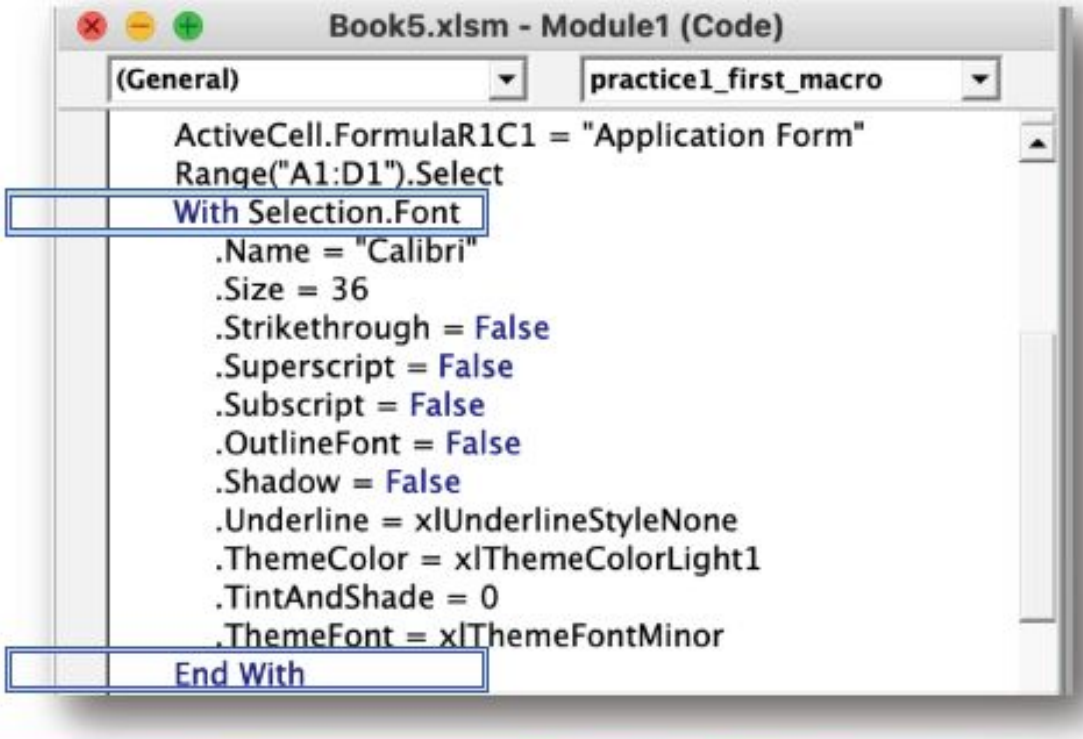
Object properties – Other properties

- VBA
  - Worksheets("Sheet1").Range("B2").Font.Size = 36
  - Worksheets("Sheet1").Range("B2").Font.Color = vbRed



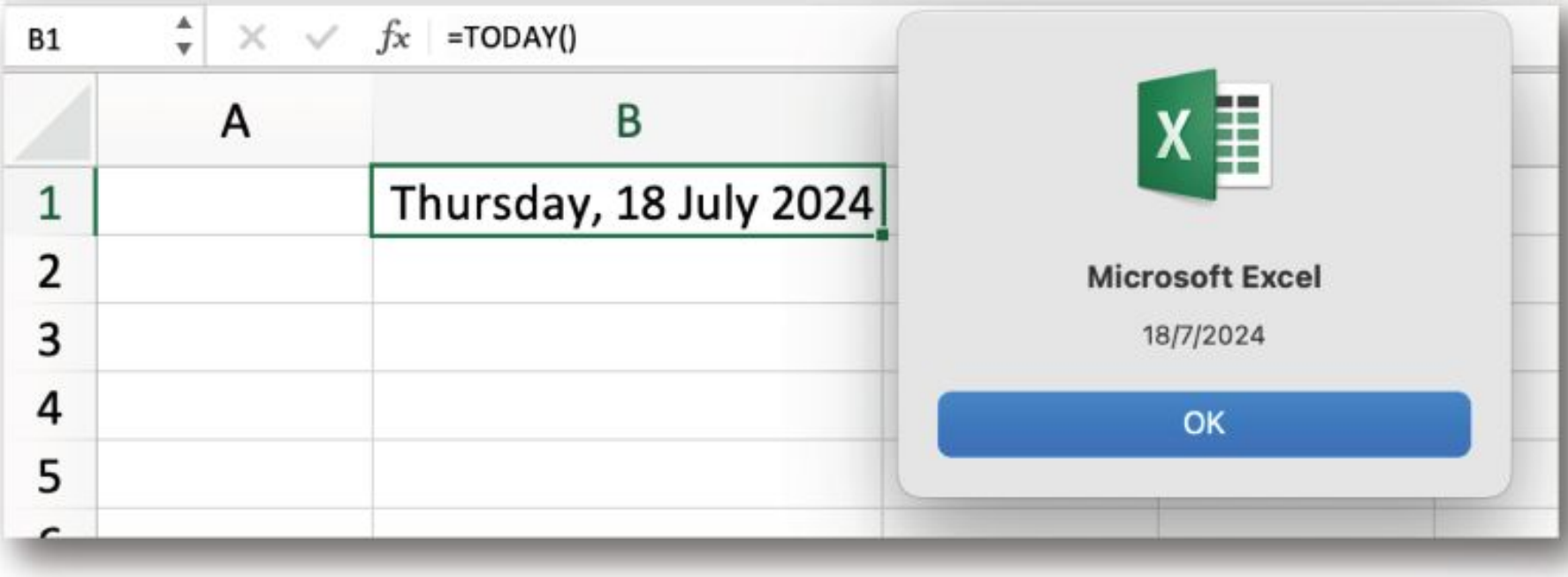
Object properties – Editing many properties

- "With" statement
  - Reduce redundancy
  - Example from Practice 1



Object properties – Value check

- Use “MsgBox”
  - MsgBox(Worksheets("Sheet1").Range("B1").Value)



Object properties – Value check

- Use “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
```

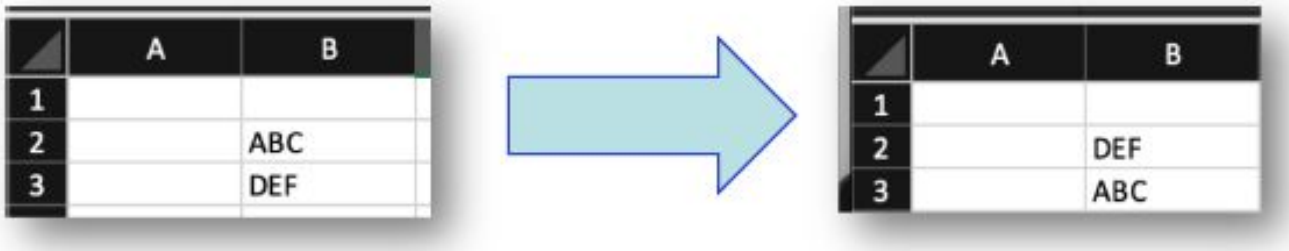
- Use “?” in the immediate window
  - ?ActiveWorkbook.Sheets.Count
  - ?Range("B2").Formula

```
Immediate
?ActiveWorkbook.Sheets.Count
 7
```

```
?Range("B2").Formula
=TODAY()
```

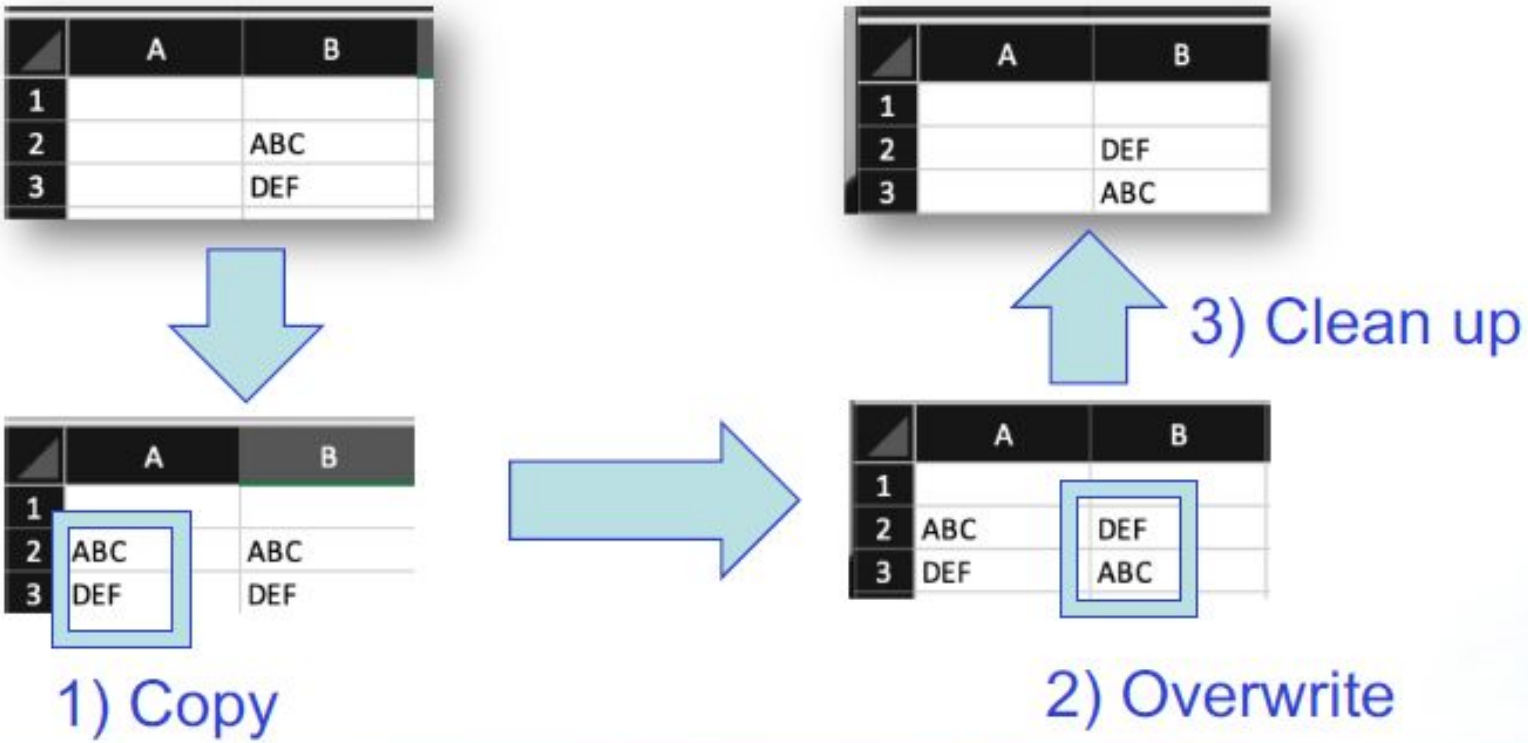
Checkpoint – Value swapping

- How to swap (switch) values between 2 cells?



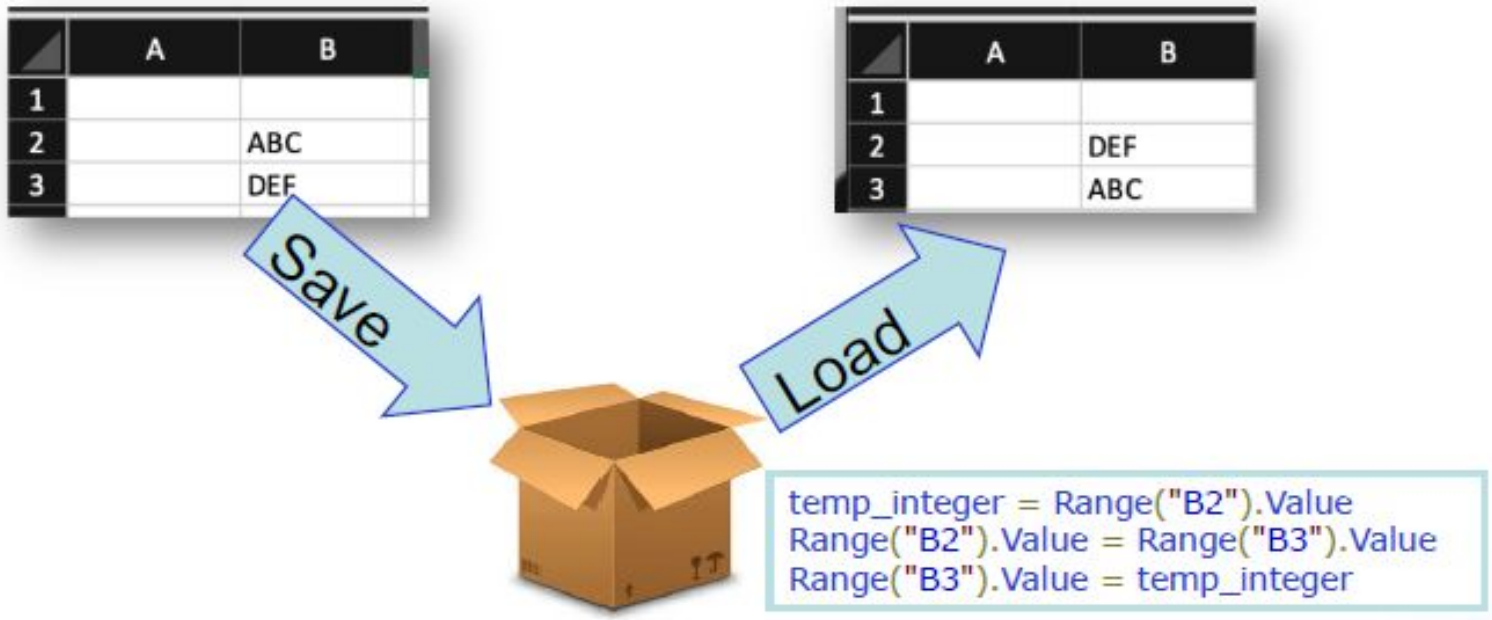
Checkpoint – Value swapping

- How to swap (switch) values between 2 cells?
  - 1 solution…



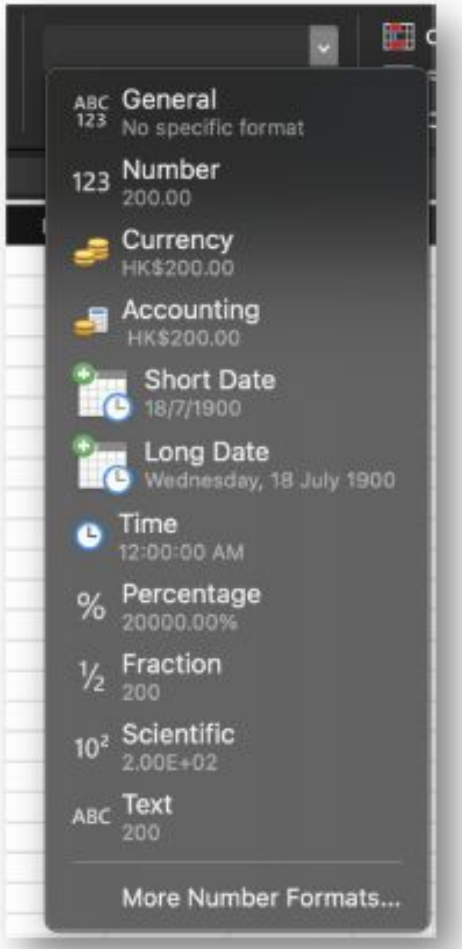
Variables – Motivation

- Storage of values without changing the sheets
- Meaningful references during calculation

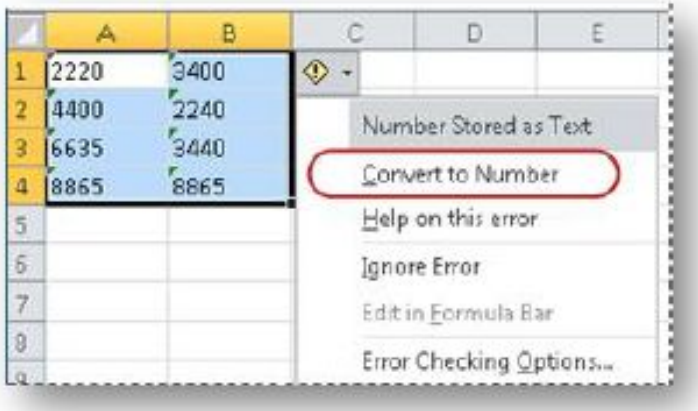


Variables – Data Types

- Excel is good at detecting types
  - Numbers / Date / Text



- Though sometimes can still be wrong



1. *Fix text-formatted numbers by applying a number format - Microsoft Support*. Fix Text-formatted Numbers by Applying a Number Format - Microsoft Support. https://support.microsoft.com/en-us/office/fix-text-formatted-numbers-by-applying-a-number-format-6599c03a-954d-4d83-b78a-23af2c8845d0

Variables – Data Types

- Best to define variables with types

Commonly Used VBA Data Types

Data Type	Bytes Used	Range of Values
Integer	2	-32,768 to 32,767
Long	4	-2.147 Billion to 2.147 Billion
Single	4	45 digits (left or right of decimal point)
Double	8	308 digits (left or right of decimal point)
Currency	8	-922 T to 922 T (4 digits after decimal point)
Date	8	1/1/100 to 12/31/9999
Boolean	2	True or False
String	1 per char	Varies according to number of characters
Variant	Varies	Any data type

Variables – Data Types

- Integer
  - Dim year As Integer
  - year = 2024
- Double
  - Dim pi As Double
  - pi = 3.14125

- Smart conversion during calculation

```
?TypeName( "A" & 3 )
String

?"A"&3
A 3
```

```
?TypeName(2)
Integer

?TypeName(3.3)
Double

?TypeName(2 * 3.4)
Double

?TypeName(5-3)
Integer
```

Variables – Data Types

- Date
  - Dim today As Date
  - today = DateSerial(2024, 7, 1)
  - today = Date()
- String
  - Dim message As String
  - message = "Welcome!"
- Boolean
  - Range("A1").Font.Bold = True
  - Range("A1").Font.Underline = False

Common functions - Numbers

- IsNumeric() ➔ Check if a text is number

```
?IsNumeric("123")
True

?IsNumeric("ABCD")
False
```

- CInt() / CDbl() ➔ Text to number

```
?CInt("123")
 123

?CInt("123.45")
 123

?CDbl("123.45")
 123.45

?CDbl("123")
 123
```

Common functions - Numbers

- Round() ➔ Rounding to a certain decimal place

```
?Round(5.1234)
 5

?Round(5.1234, 2)
 5.12
```

- Log() / Exp() ➔ For financial / engineering calculation

```
?exp(2)
 7.38905609893065

?log(10)
 2.30258509299405

?log(exp(1))
 1
```

Common functions - Date

- DateSerial ➔ Create a date literal

```
?DateSerial(2024, 12, 25)
25/12/2024
```

- Date() ➔ Today's date

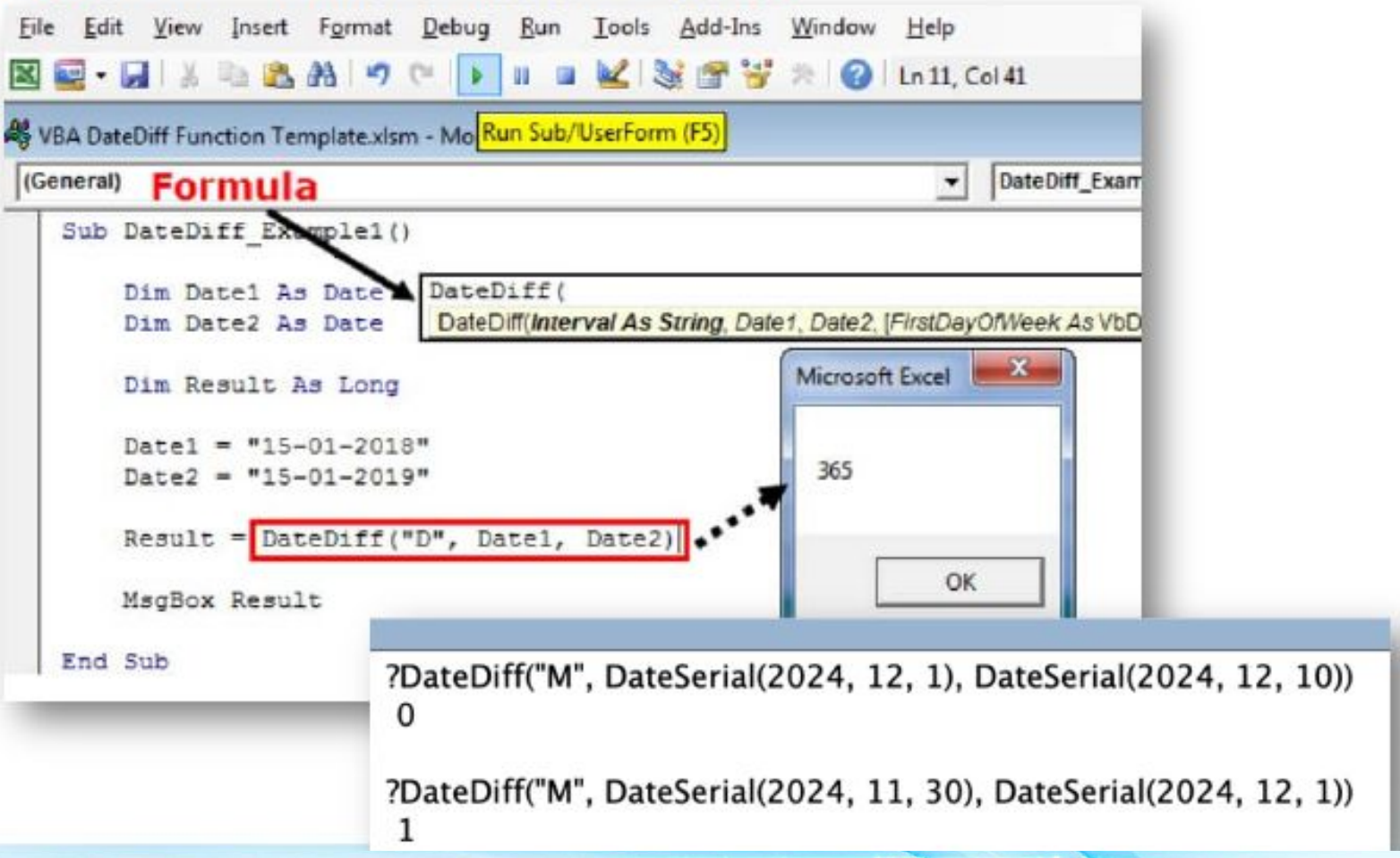
```
?Date()
18/7/2024
```

- Year() / Month() / Day() ➔ Extract

```
?Year(DateSerial(2024, 12, 25))
 2024

?Month(DateSerial(2024, 12, 25))
 12

?Day(DateSerial(2024, 12, 25))
 25
```

Common functions - Date

- DateDiff – Difference of 2 dates



Common functions - String

- CDate() ➔ Text to Date

```
?CDate("2023-03-25")
25/3/2023

?CDate("April 1, 2023")
1/4/2023

?CDate("25/2")
25/2/2023
```

- Format() ➔ Date to Text

```
?Format(DateSerial(2023, 3, 25), "M")
3

?Format(DateSerial(2023, 3, 25), "MM")
03

?Format(DateSerial(2023, 3, 25), "MMM")
Mar

?Format(DateSerial(2023, 3, 25), "MMMM")
March

?Format(DateSerial(2023, 3, 25), "DD-MMM-YYYY")
25-Mar-2023
```

Common functions - String

- Left() / Right() ➔ Substring

```
?Left("0005.HK", 4)
0005

?Right("0005.HK", 2)
HK
```

- Split() ➔ Tokenizing into a string array

```
?Split("A,B,C", ",")(0)
A

?Split("A,B,C", ",")(2)
C
```

```
?UBound(Split("A,B,C", ","))
 2
```

Common functions - String

- Trim() ➔ Remove empty spaces
  - Note: "" gives 1 quote

```
?""""" & Trim("    Hello!    ") & """""
"Hello!"
```

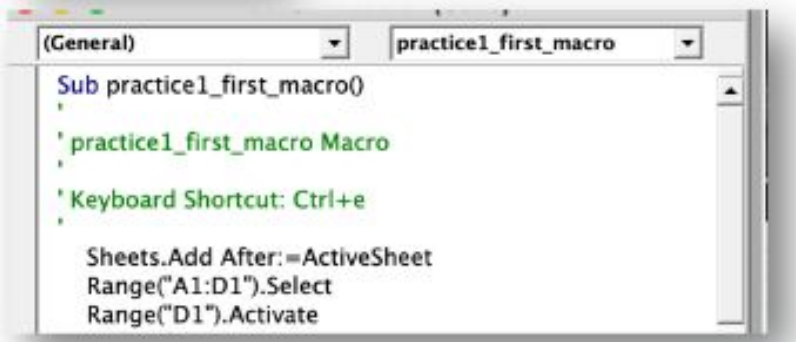
- InStr(x, y) ➔ Find y in x
  - Case sensitive

```
?InStr("Search here", "Search")
 1

?InStr("Search here", "here")
 8

?InStr("Search here", "nothing")
 0

?InStr("Search here", "search")
 0
```

```
?Lcase("HeLLo")
hello

?InStr(Lcase("Search here"), "search")
 1
```

Sub-routines

- Macros recorded are wrapped with "Sub … End Sub"
- Sub-routines are just reusable codes
- Within the scope of a sub-routine
  - Click the "Play" button to execute



Control flow – If, Then, Else

- Example: speed check


```
Sub HighlightCell()

    If Range("B1").Value > 50 Then
        Range("B1").Interior.Color = vbYellow
    Else
        Range("B1").Interior.Color = vbGreen
    End If

End Sub
```

Control flow – If, Then, Else

- Mark ➔ Grade

Mark	*Letter Grade*
90+	A+
85-89	A
80-84	A-
77-79	B+
73-76	B
70-72	B-
67-69	C+
63-66	C
60-62	C-
57-59	D+
50-56	D
0-49	F

Control flow – Select, Case

- Range check

```
?GetGrade(90)
A

?GetGrade(80)
B
```

```
Function GetGrade(score As Integer) As String
        Select Case score
        Case Is >= 90
                GetGrade = "A"
        Case Is >= 80
                GetGrade = "B"
        Case Is >= 70
                GetGrade = "C"
        Case Is >= 60
                GetGrade = "D"
        Case Else
                GetGrade = "F"
        End Select
End Function
```

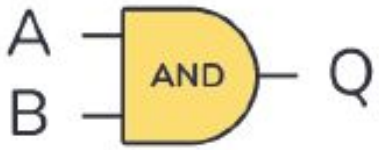
Control flow – Select, Case

- Grade ➔ Marks

```
Dim letter_grade As String
Dim grade_point As Double

Select Case letter_grade
Case "A", "A+"
        grade_point = 4.0
Case "A-"
        grade_point = 3.7
Case "B+"
        grade_point = 3.3
...
Case Else
        grade_point = 0.0
End Select
```

Control flow – Conditions

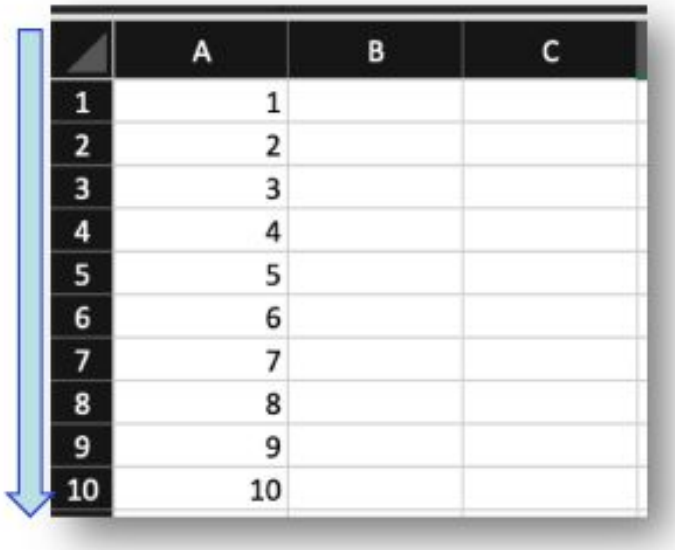
- AND
  - (True AND True) ➔ True
  - e.g., Month validity: 1 <= month AND month <= 12
- OR
  - (True OR False) ➔ True
  - e.g., Game Area: gold > 1000 OR level > 10
- NOT
  - NOT x=0 is same as x <> 0



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Loops – Using "For"

- Iterating through the rows
- What is changing?
  - A1 ➔ A2 ➔ A3 ➔ A4…

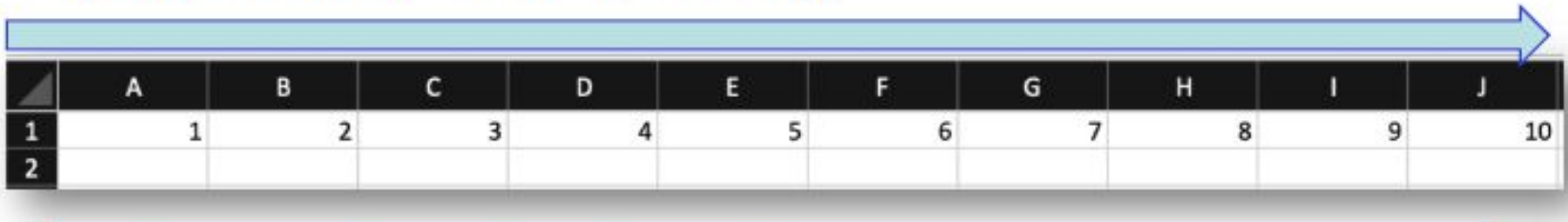


	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
Sub LoopThroughRows()
        For i = 1 To 10
                Range("A" & i).Value = i
        Next i
End Sub
```

Loops – Using "For"

- Iterating through the columns
- Again, track the change: A1 ➔ B1 ➔ C1 ➔ …
- Option 1: Make use of offset()



	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
2										

```
Sub LoopThroughColumns()
        For i = 1 To 10
                Range("A1").Offset(0, i - 1).Value = i
        Next i
End Sub
```

Loops – What is happening?

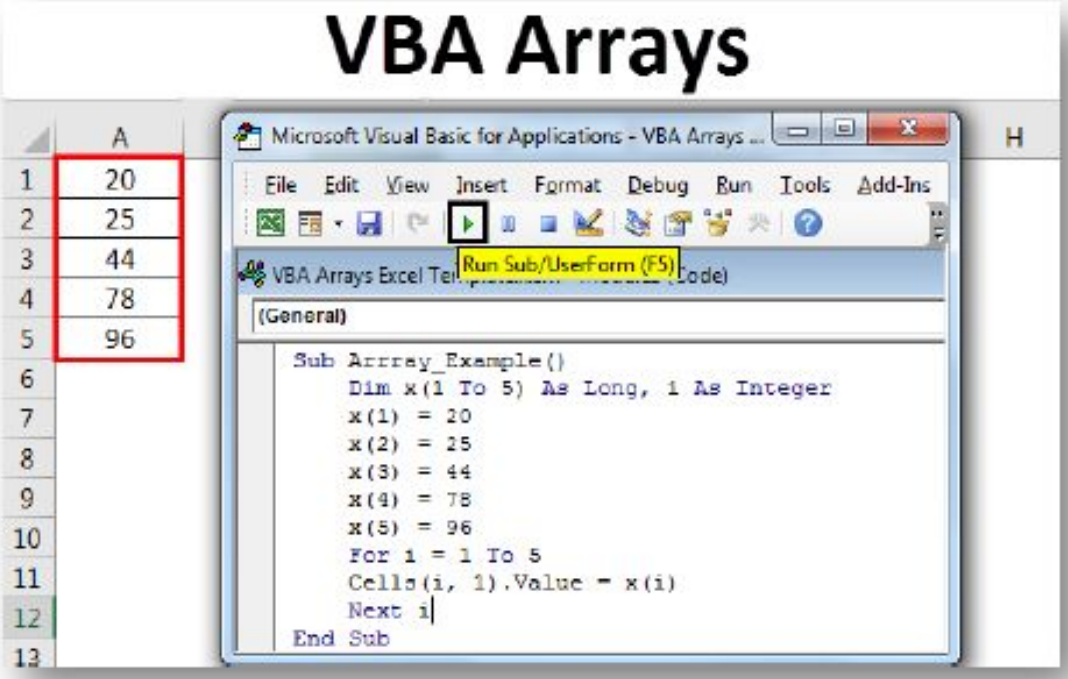
- Temporary variable + "Array"

```
Sub LoopThroughRows()
        For i = 1 To 10
                Range("A" & i).Value = i
        Next i
End Sub
```

```
Sub LoopThroughColumnsWithCells()
        Dim i As Integer: i = 1
        For Each cell In Range("A1:J1").Cells
                cell.Value = i
                i = i + 1
        Next cell
End Sub
```

Loops – What is happening?

- Array = many variables



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

Business Process Automation with Python

Motivation – Scope

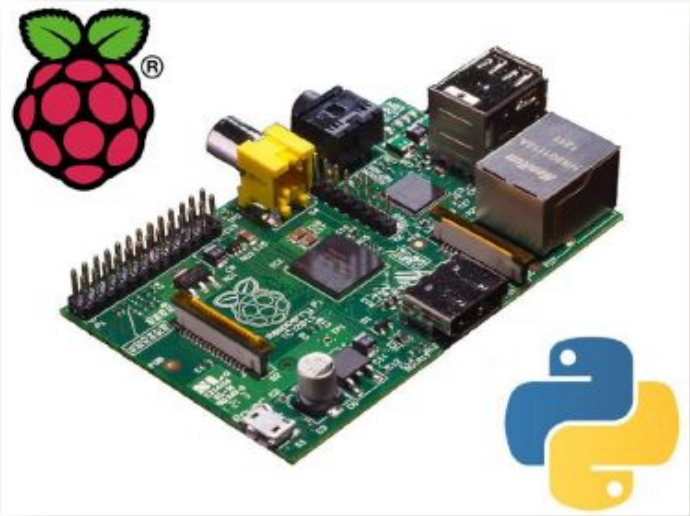
- How about outside of Microsoft Office?
- Program trading[1] / Web / AI / Business Intelligence

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

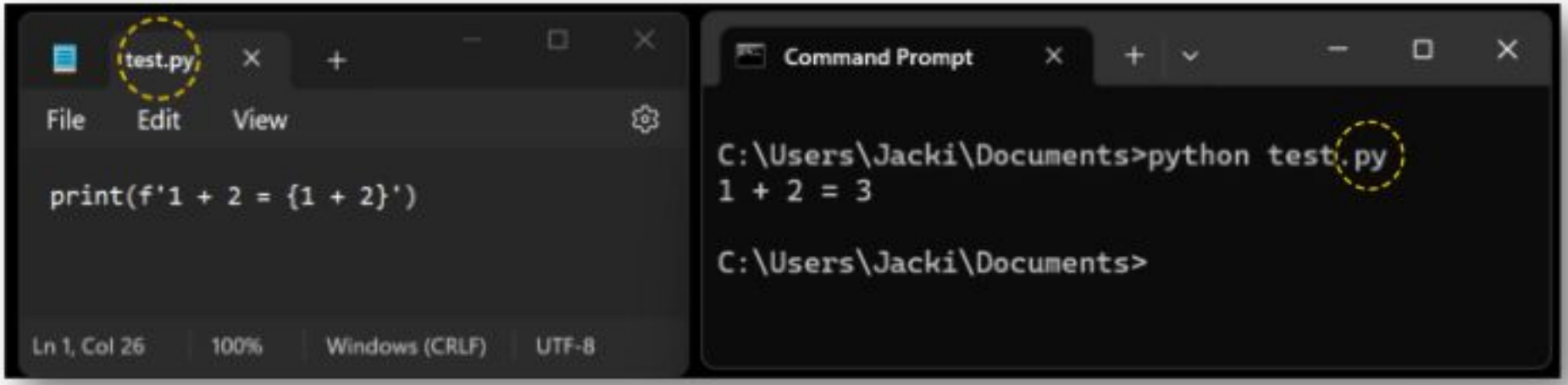
- How about outside of ~~Microsoft~~ Office?
- e.g., Drones / Security cameras / Firewalls



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

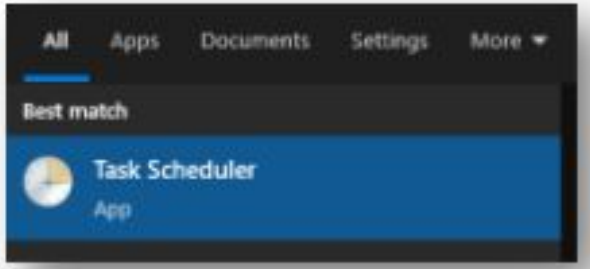
How to run Python – Running as a file

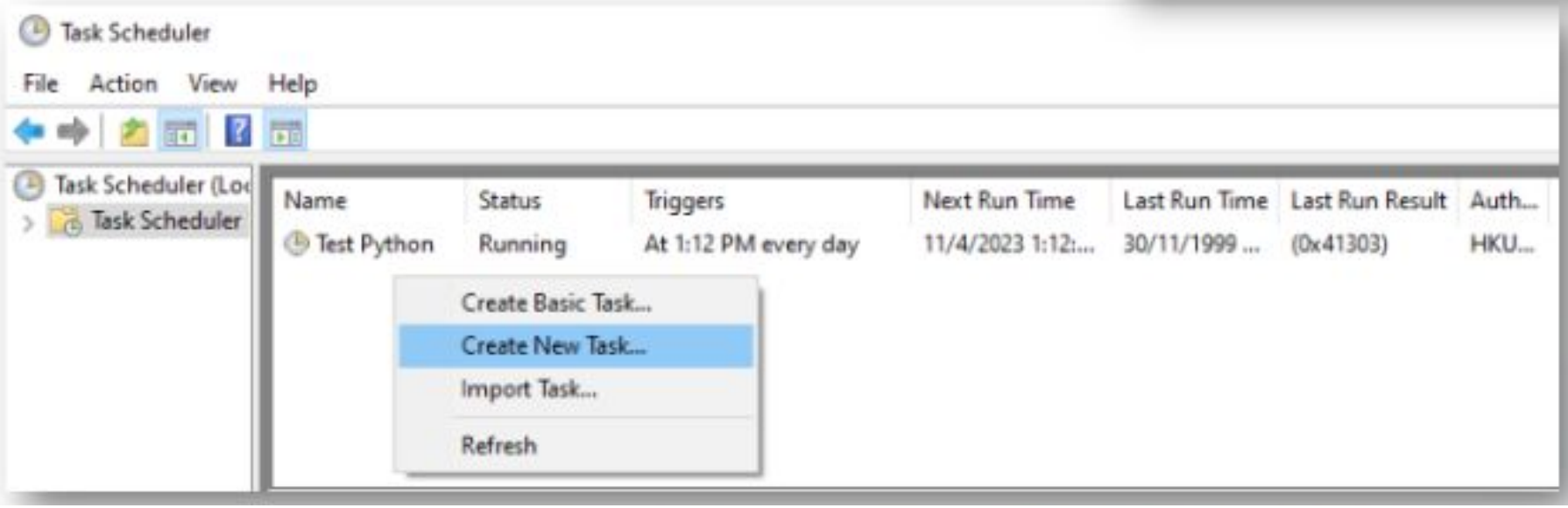
- Python interpreter + Text editor
  - Write codes in a text editor (e.g., Notepad, Vim, Sublime Text)
  - Save codes as a ".py" file
  - Run all the lines in one go



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



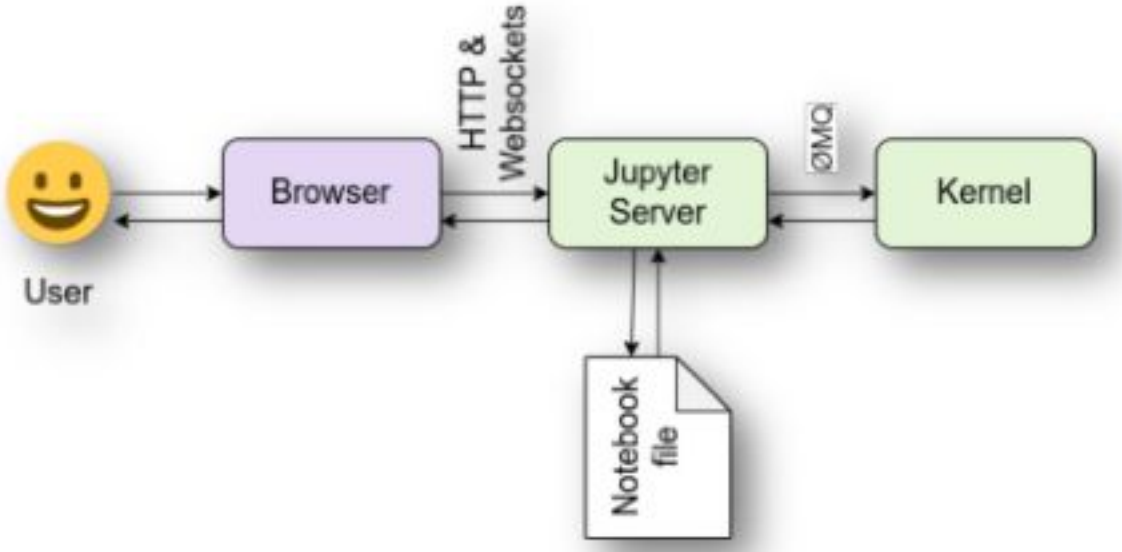


How to run Python – IDE

- 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

How to run Python – Jupyter Notebook

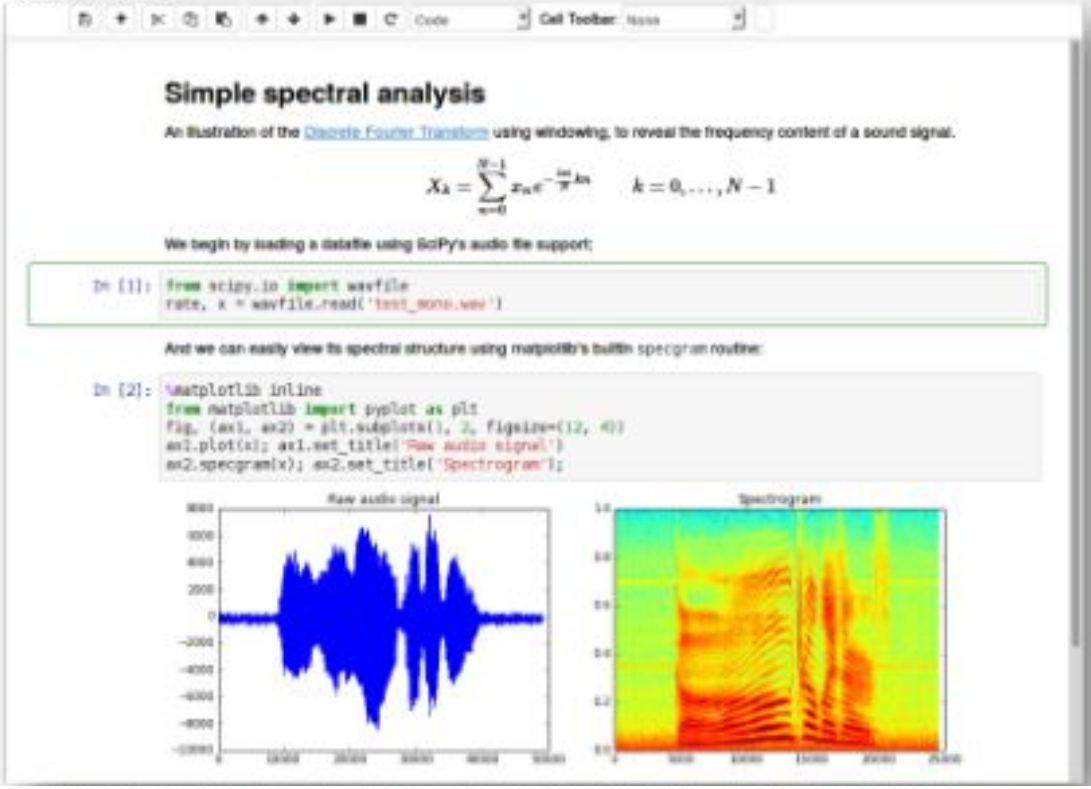
- Web-based interactive platform
  - Accessible: Python runtime on a web server



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



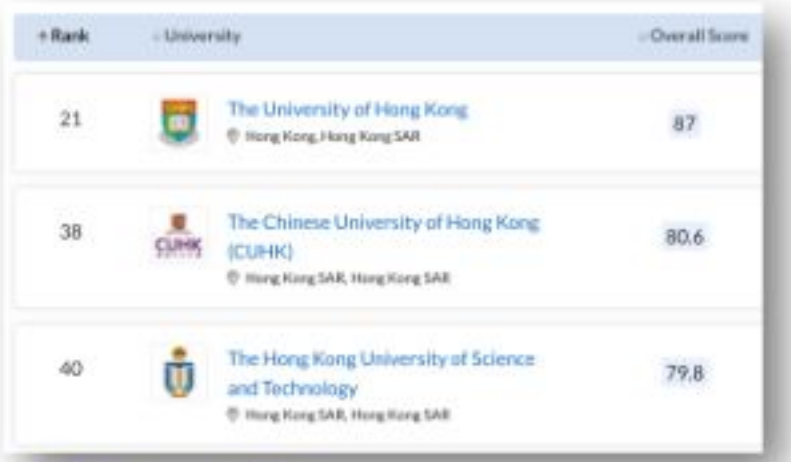
How to run Python – Jupyter Notebook

- 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

Sequential Data Types

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

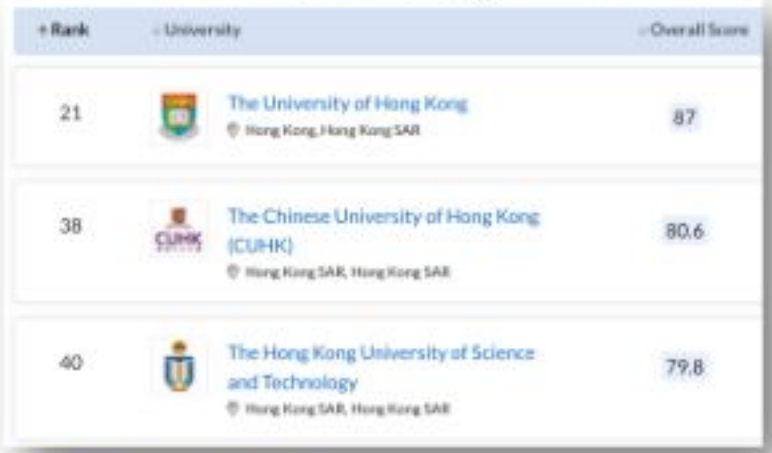


Rank	University	Overall Score
21	The University of Hong Kong Hong Kong, Hong Kong SAR	87
38	The Chinese University of Hong Kong (CUHK) Hong Kong SAR, Hong Kong SAR	80.6
40	The Hong Kong University of Science and Technology Hong Kong SAR, Hong Kong SAR	79.8

* 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!']))
```

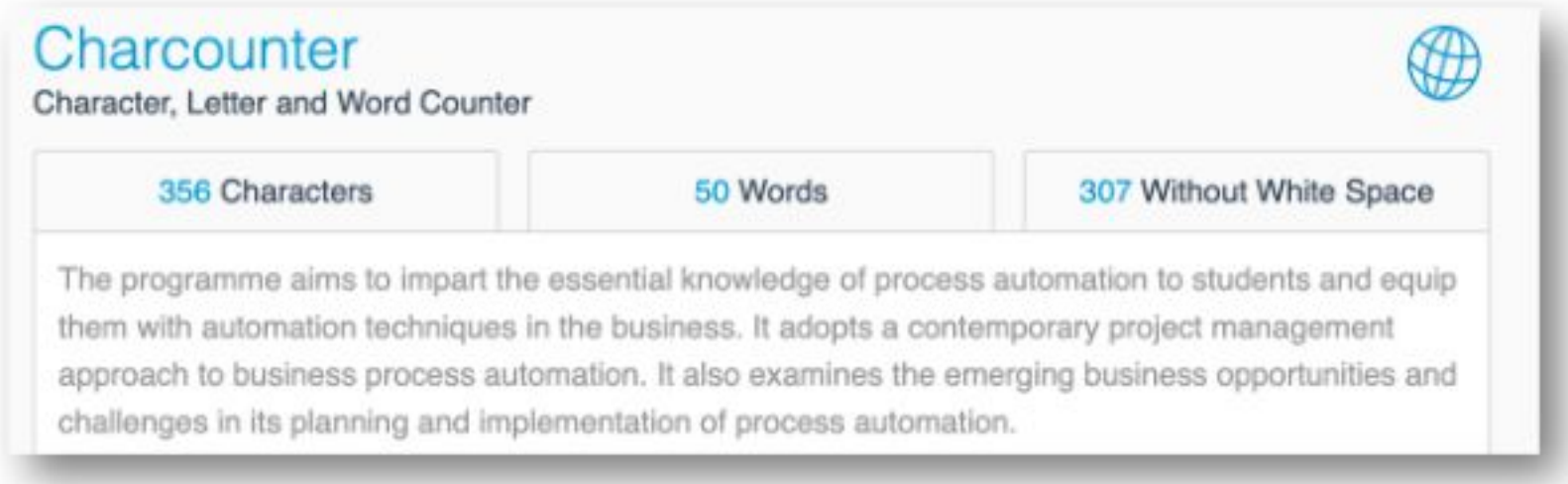
```
hello
Welcome,Jackie!
```

```
# 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'>
```

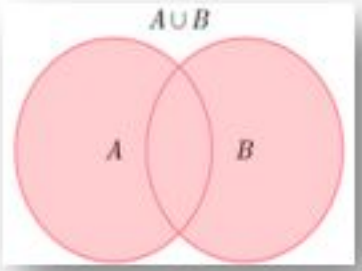
  - Can be converted between sequential data types

```
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

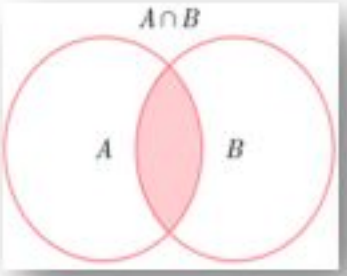



```
# Set operations
school_set_2022 = {'A', 'B', 'C'}
school_set_2023 = {     'B', 'C', 'D'}
```

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

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

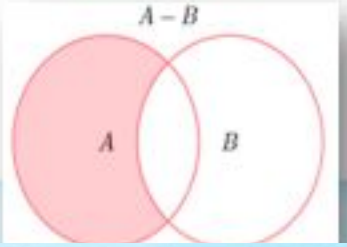
  - Intersection




```
# 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

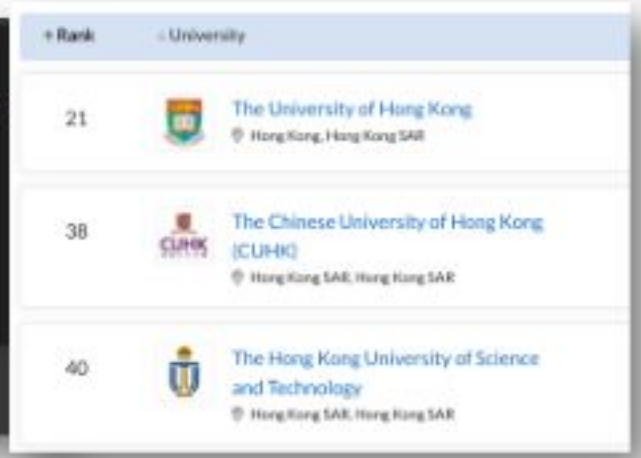
  - Looking up a keyword
    ➔ like finding words in **dictionaries**

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'}
```

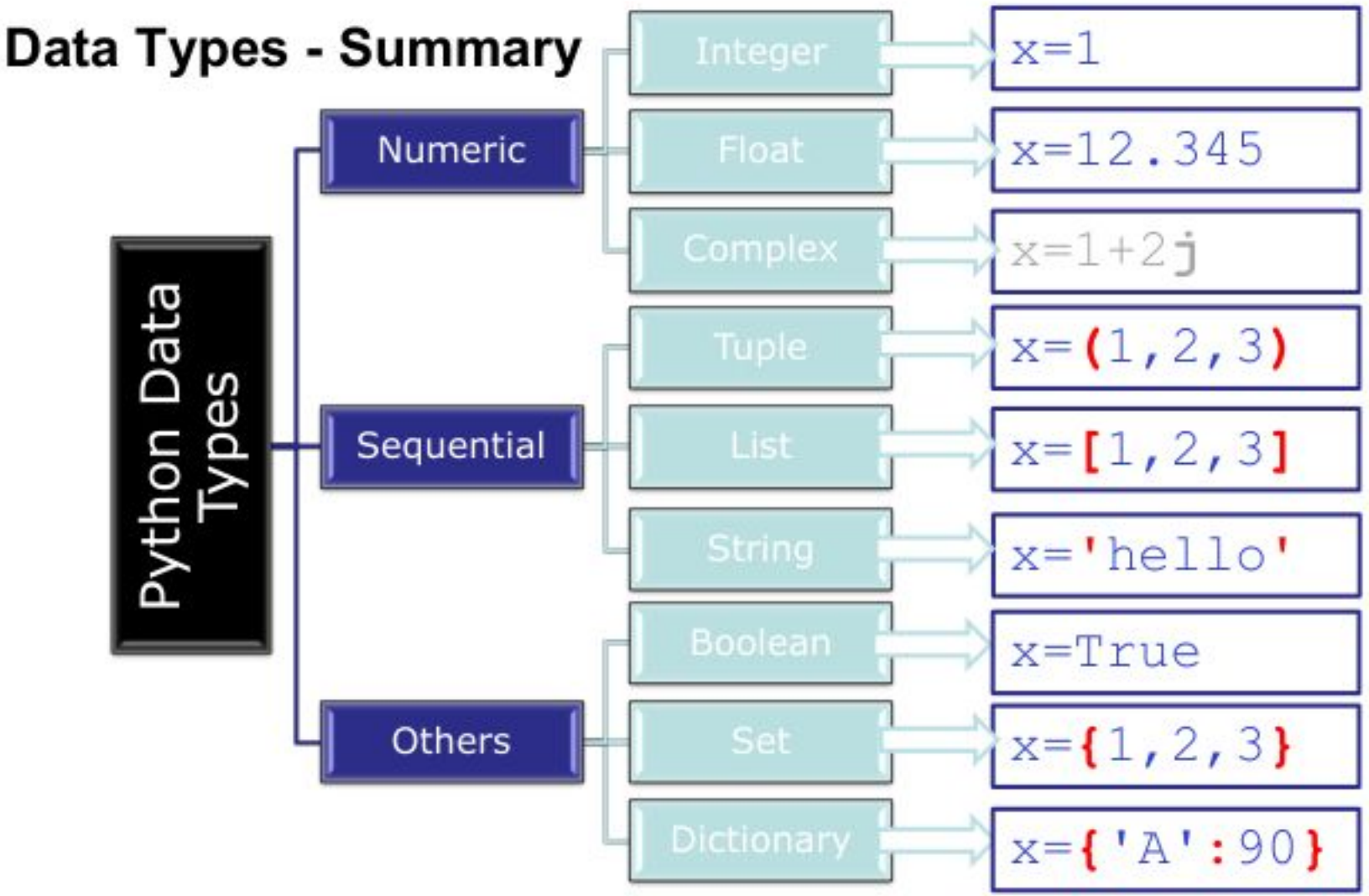
Data Type - Dictionary

- **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

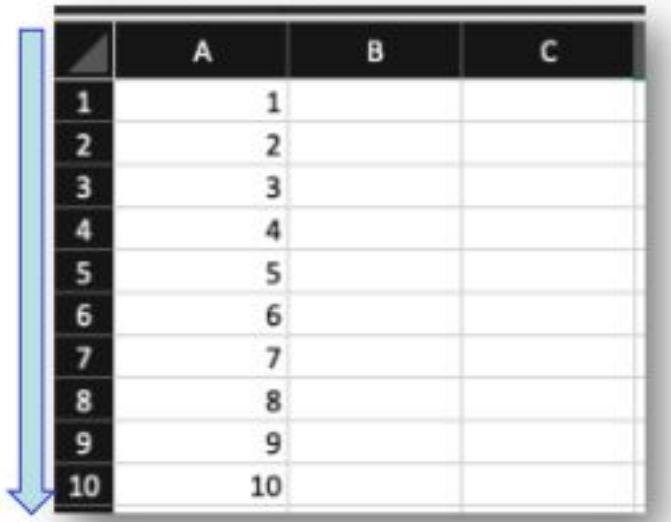
```
Dim speed As Integer
speed = 70

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

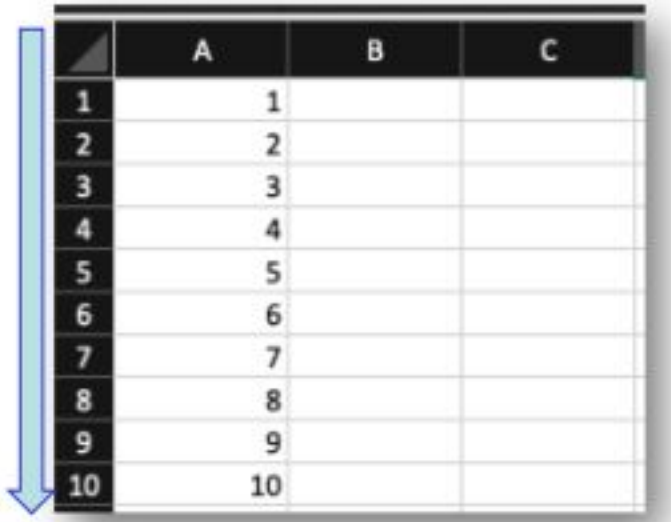
```
For i = 1 To 10
        Debug.Print (i)
Next i
```

Python

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

Loops – Using "for"

- Different ways to specify ranges



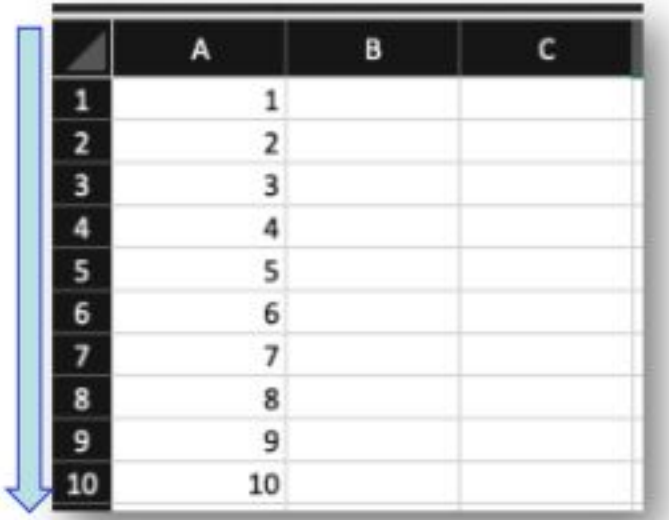
	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		

```
[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)
```

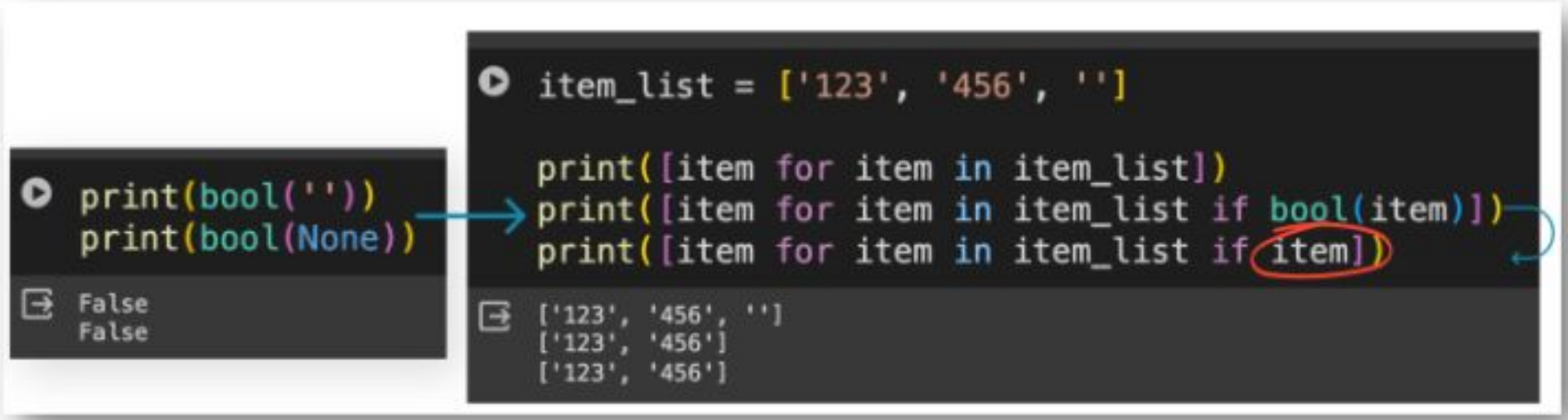
```
[0, 2, 4, 6, 8, 10]
```

```
# 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]
```

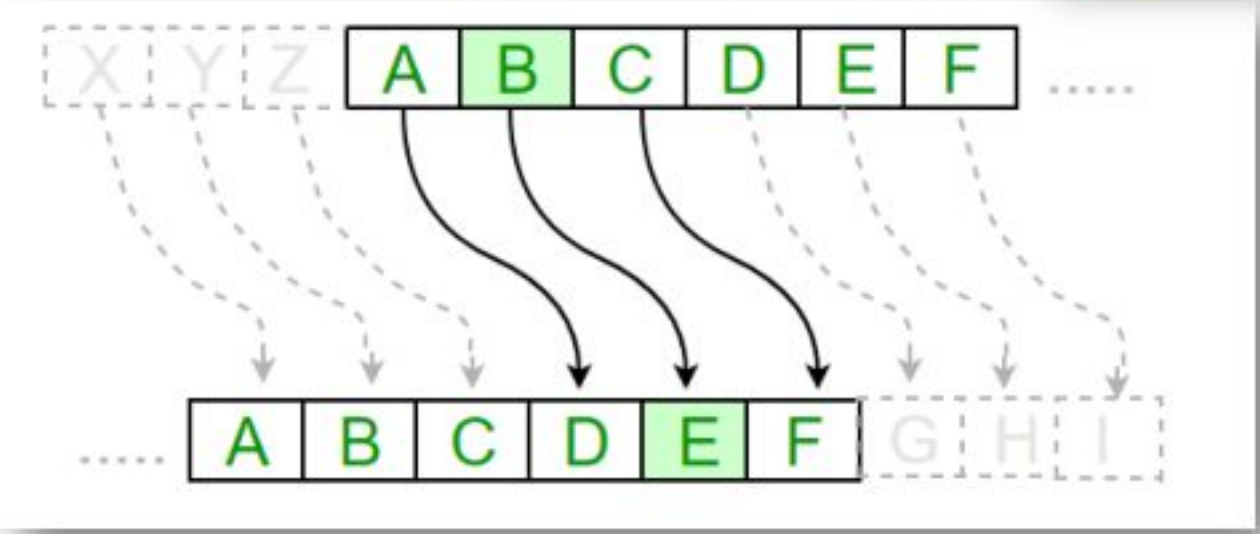
Loops – Using "for"

- List comprehension + Filtering



Practice 3 – Caesar Cipher

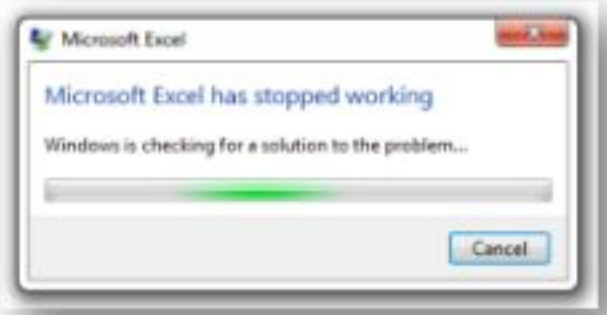
- **Encryption: Shift each letter by 3**
  - Reference: https://cryptii.com/pipes/caesar-cipher

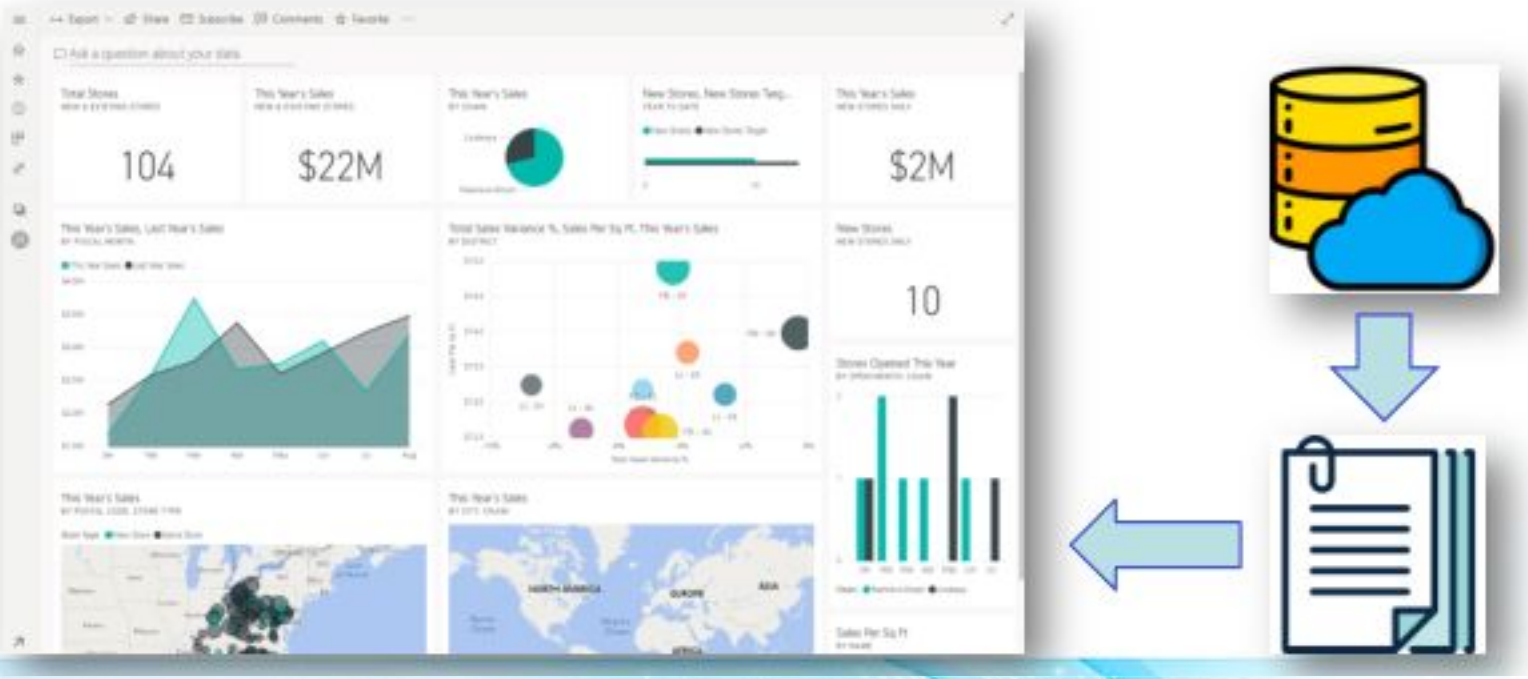


- **Starting notebook:**
  https://github.com/innoviai/ipa_courses/blob/main/Lecture%202/practice2_word_count_202409.ipynb

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')

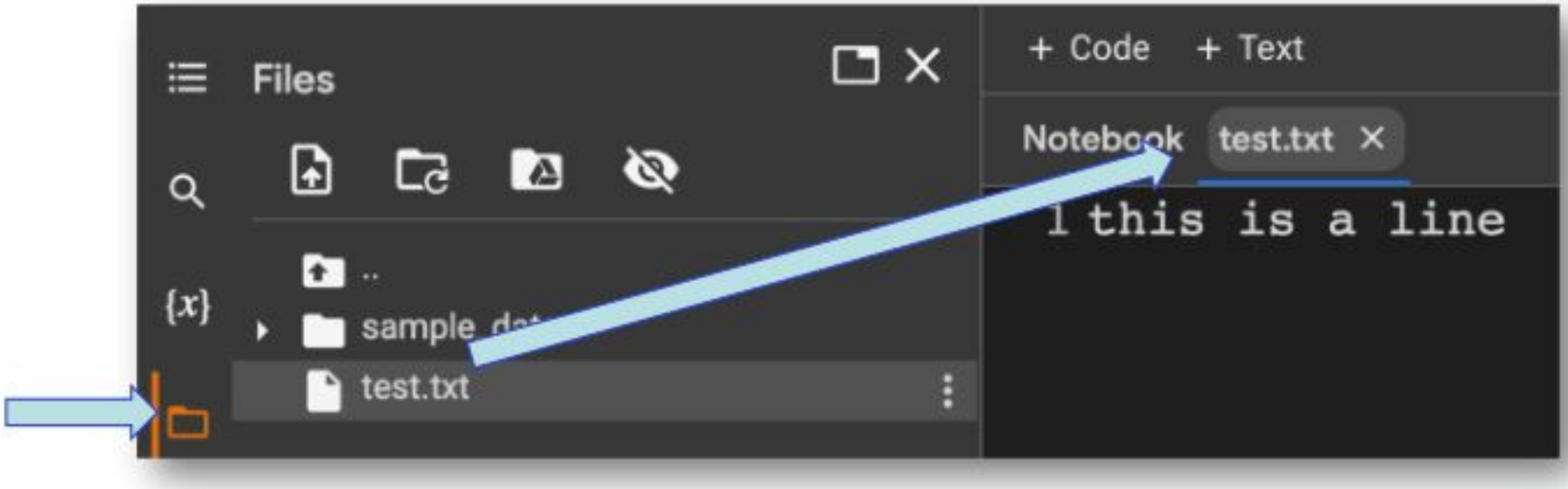
File Input / Output

- **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

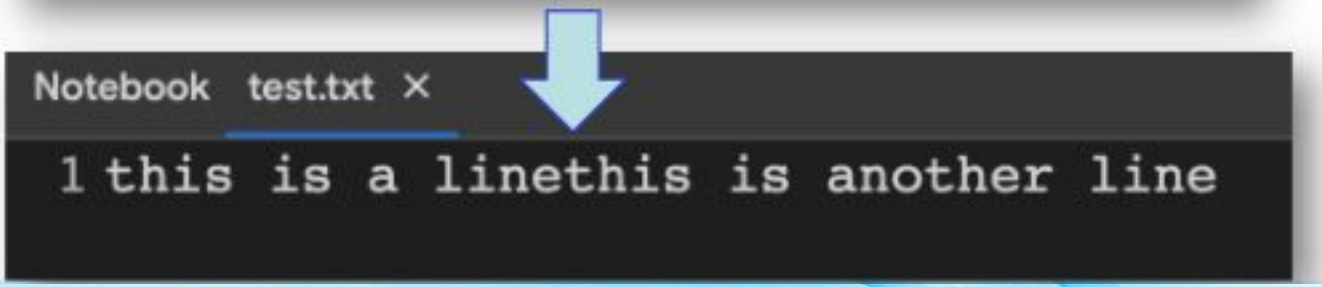
- **Writing a text file**
  - Viewing the result



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

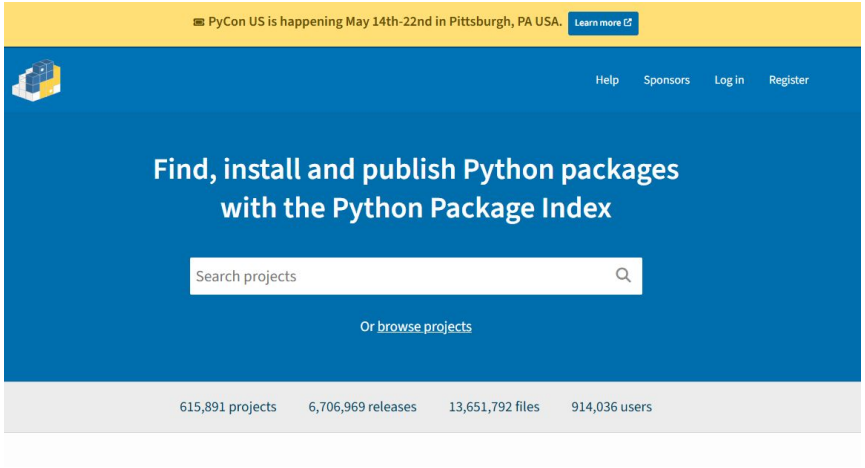
- **Reading a text file**
  - Handling separators
    - splitlines() ➔ useful against line separators '\n'
    - split() ➔ flexible, can handle commas in CSV files

```
print( file_text.split('\n') )
print( file_text.splitlines() )
```

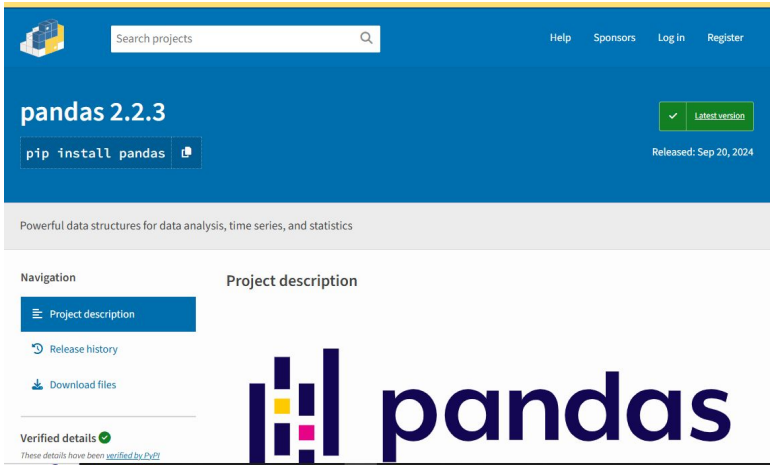
```
['this is a line', 'this is another line']
['this is a line', 'this is another line']
```

Library Installation in Python

Most of the python library can be founded in https://pypi.org/

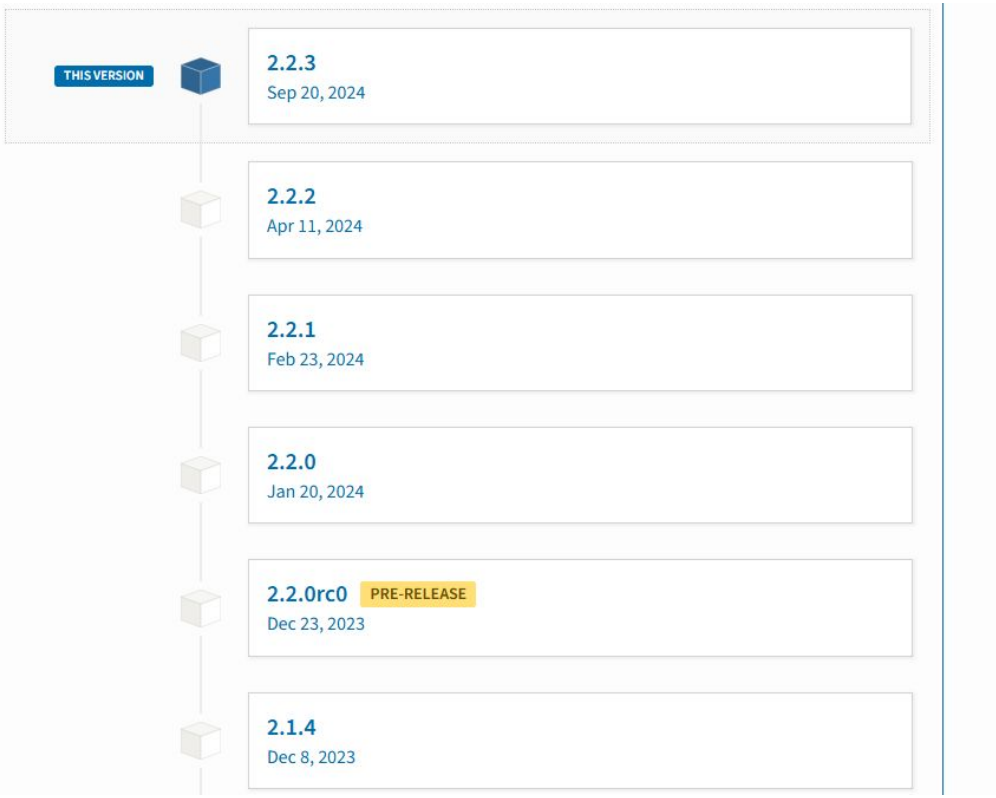


You can enter pandas to search or type https://pypi.org/project/pandas/



Library Installation in Python

Same library name have different version like 2.2.3, 2.1.0, …
The default page show the newest version.



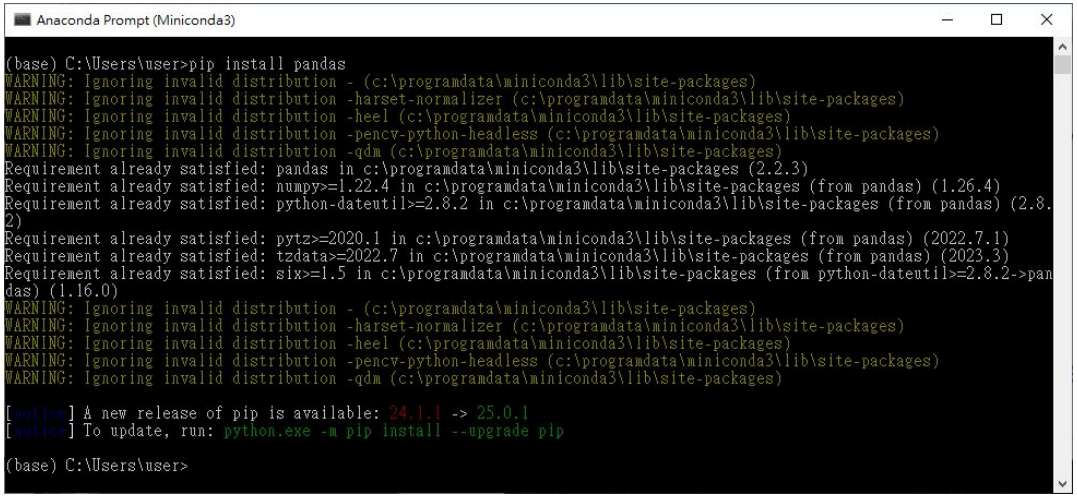
If you use python 3.7 or 3.8, you better install the version 2.1.0

For library installation, you can go to anaconda prompt and enter
“pip install pandas” or “pip install pandas==2.1.0”

Library Installation in Python

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

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



How to Manage Your Budget with a Simple Python Script

Open Jupyter notebook

1. Getting User Input

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

def get_user_input():
    """Get user input for income and expenses."""
    income = float(input("Enter your income: "))
    # Expecting expenses to be a dictionary input
    expenses = {}
    while True:
      category = input("Enter expense category (or 'done' to finish): ")
      if category.lower() == 'done':
            break
      amount = float(input(f"Enter amount for {category}: "))
      expenses[category] = amount
    return income, expenses
```

How to Manage Your Budget with a Simple Python Script

2. Calculating the Budget

```
def calculate_budget(income, expenses):
    """Calculate total expenses and balance."""
    total_expenses = sum(expenses.values())
    balance = income - total_expenses
    return total_expenses, balance
```

3. Displaying the Budget Summary

```
def display_budget_summary(income, total_expenses, balance):
    """Display the budget summary."""
    print(f"Income: ${income:.2f}")
    print(f"Total Expenses: ${total_expenses:.2f}")
    print(f"Balance: ${balance:.2f}")
```

How to Manage Your Budget with a Simple Python Script

4. Plotting the Expenses

```
def plot_expenses(expenses):
    """Plot the expenses as a bar chart."""
    df = pd.DataFrame(list(expenses.items()), columns=['Category', 'Amount'])
    df.plot(kind='bar', x='Category', y='Amount', legend=False)
    plt.ylabel('Amount ($)')
    plt.title('Expense Distribution')
    plt.show()
```

How to Manage Your Budget with a Simple Python Script

5. Main Function

```
income, expenses = get_user_input()
total_expenses, balance = calculate_budget(income, expenses)
display_budget_summary(income, total_expenses, balance)
plot_expenses(expenses)
```

How to Manage Your Budget with a Simple Python Script

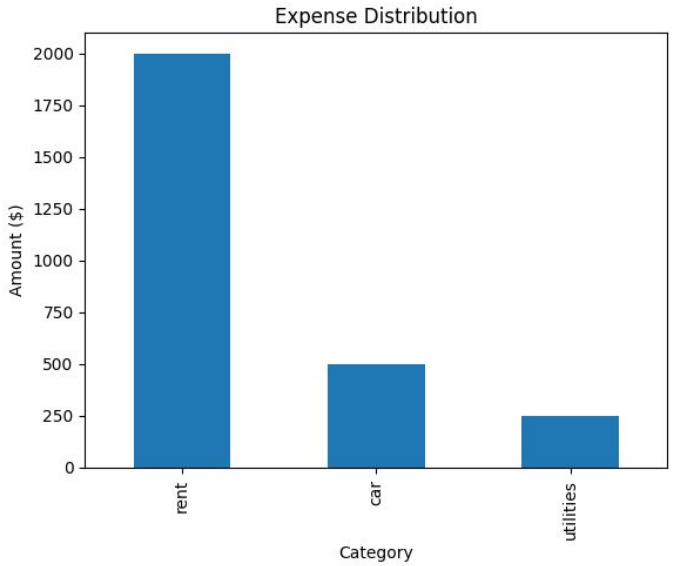
5. Main Function

Main Input

Enter your income: 10000
Enter expense category (or 'done' to finish): rent
Enter amount for rent: 2000
Enter expense category (or 'done' to finish): car
Enter amount for car: 500
Enter expense category (or 'done' to finish): utilities
Enter amount for utilities: 250
Enter expense category (or 'done' to finish): done

Result

Income: $10000.00
Total Expenses: $2750.00
Balance: $7250.00



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:

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

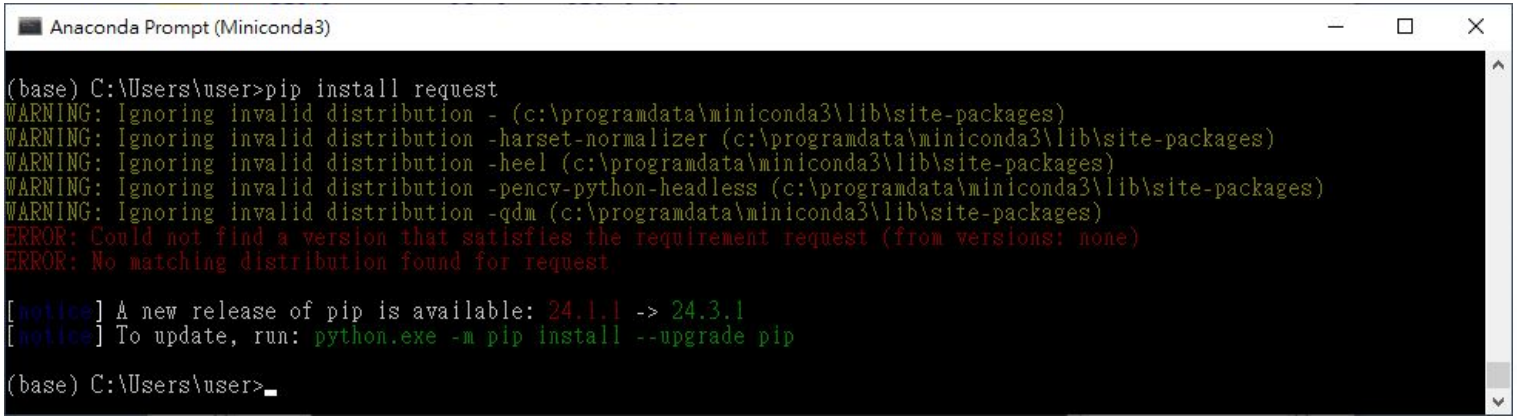
Data Sources	Files, APIs, Webpages (via Requests)
Data Parsing	Regular Expressions, Beautiful Soup
Data Structures/Storage	Traditional lists/dictionaries, PANDAS
Models	Linear Regression, Logistic Regression, kNN, etc

BeautifulSoup only concerns webpage data

Obtaining Data – Web Scraping

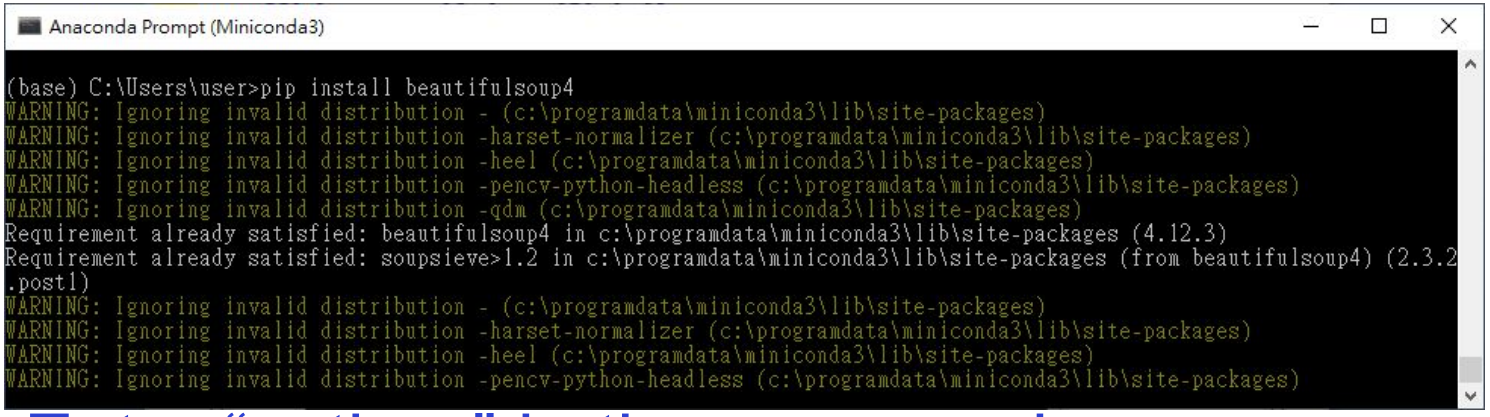
1. Open tools “Anaconda Prompt” which will direct you to the command prompt.

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

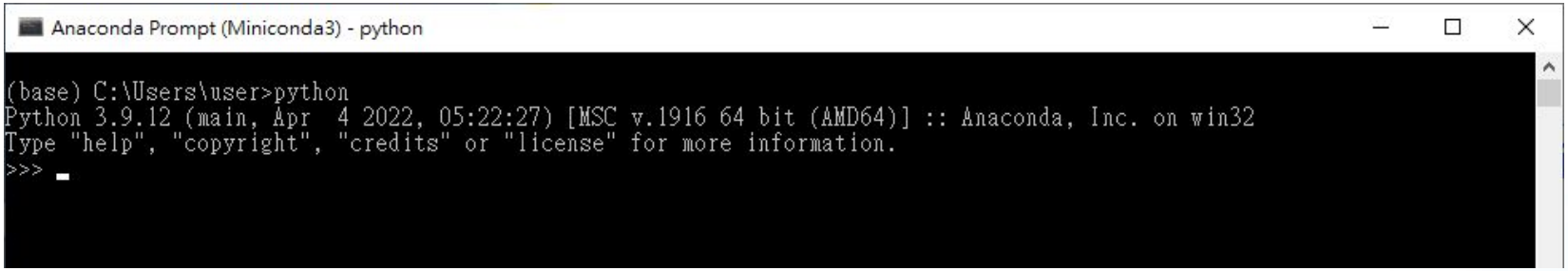


Obtaining Data – Web Scraping

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



4. 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
```

Obtaining Data – Web Scraping

Open Chrome Browser and enter https://www.hk01.com/

Right Click and Select "Inspect"

Select the Content that you want to extract

Obtaining Data – Web Scraping

Right Click and click “Edit As HTML”
Copy part of the code until “/div>

Obtaining Data – Web Scraping

Open Perplexity at https://www.perplexity.ai/

Enter prompt "write python beatifulsoap to extract data from website url https://www.hk01.com/ with html code {htmlcode}"

Where {htmlcode} = " <button data-testid="hamburger-menu-button" class="box-border w-6 px-0 py-3 m-0 transition-all duration-[0.6s] ease-[cubic-bezier(0.19,1,0.22,1)] [transform:translateZ(0)] border-none outline-none rounded-none cursor-pointer shadow-none bg-transparent focus:shadow-none focus:outline-none" aria-label="mobile-menu"><span class="relative block h-px rounded-[1px] transition-all duration-[0.6s] ease-[cubic-bezier(0.19,1,0.22,1)] w-[70%] left-[15%] bg-black/[0.78] before:w-full after:w-full before:absolute after:absolute before:left-0 after:left-0 before:content-[' '] after:content-[' '] before:h-px after:h-px before:rounded-[1px] after:rounded-[1px] before:transition-all before:duration-[0.6s] before:ease-[cubic-bezier(0.19,1,0.22,1)] after:transition-all after:duration-[0.6s] after:ease-[cubic-bezier(0.19,1,0.22,1)] before:bg-black/[0.78] after:bg-black/[0.78] before:-top-1.5 before:rotate-0 after:top-1.5 after:rotate-0"></button> "

Obtaining Data – Web Scraping

In original perplexity prompt, type the following prompt "Add code that save it into csv file"

	A
1	text,url,class
2	,/,
3	港聞,/zone/1/%E6%B8%AF%E8%81%9E,
4	兩會焦點,/issue/10355/%E5%85%A9%E6%9C%832025-%E5%9C%8B%E9%9A%9B%E5%B1%80%
5	娛樂,/zone/2/%E5%A8%9B%E6%A8%82,
6	最平酒店,https://clk.omgt3.com/?AID=2200313&PID=55029&uid=content&uid2=channel_旅遊_editor
7	國際,/zone/4/%E5%9C%8B%E9%9A%9B,
8	即時,/latest,
9	熱榜,/hot,
10	生活,/zone/8/%E7%94%9F%E6%B4%BB,
11	科技,/zone/11/%E7%A7%91%E6%8A%80%E7%8E%A9%E7%89%A9,
12	中國,/zone/5/%E4%B8%AD%E5%9C%8B,
13	體育,/zone/3/%E9%AB%94%E8%82%B2,
14	01深圳,/zone/25/01%E6%B7%B1%E5%9C%B3,
15	經濟,/zone/14/%E7%B6%93%E6%BF%9F,
16	觀點,/zone/12/%E8%A7%80%E9%BB%9E,
17	健康,/zone/24/%E5%81%A5%E5%BA%B7,
18	好食玩飛,/zone/19/%E5%A5%BD%E9%A3%9F%E7%8E%A9%E9%A3%9B,
19	女生,/zone/6/%E5%A5%B3%E7%94%9F,
20	熱話,/zone/7/%E7%86%B1%E8%A9%B1,
21	藝文格物,/zone/9/%E8%97%9D%E6%96%87%E6%A0%BC%E7%89%A9,
22	社區,/zone/10/%E7%A4%BE%E5%8D%80,
23	教育,/zone/23/%E6%95%99%E8%82%B2,
24	简,/,"font-light, pointer-events-none, !text-light-n5"
25	
26	
27	Mobile Menu Details:
28	Exists: True

Practice 2 - Quick Test

Which Data Type is Structured/Unstructed, Discret/Continuous?

E-mails	
Digital Images	
Stock Market Logs	
Historical Gold Prices	
Credit Approval Records	
Social Media Friend Relationships	
Tweets & Trending Topics	
Sales Records	

Source: Practical Data Analysis - Second Edition Hector Cuesta, Dr. Sampath Kumar OACKT Publishing BIRMINGHAM - MUMBAI

Data Collection – Fetching Data using Pandas

What / Why?

- Pandas is an *open-source* Python library (anyone can contribute)
- Allows for high-performance, easy-to-use data structures and data analysis
- Unlike NumPy library which provides multi-dimensional arrays, Pandas provides 2D table object called **DataFrame**
  (akin to a spreadsheet with column names and row labels).
- Used by *a lot* of people

Data Collection – Fetching Data using Pandas

How

- import **pandas** library (convenient to rename it)
- Use **read_csv()** function

```
import pandas as pd
dataframe = pd.read_csv("yourfile.csv")
```

Common Panda functions

High-level viewing:

- head() – first N observations
- tail() – last N observations
- columns() – names of the columns
- describe() – statistics of the quantitative data
- dtypes() – the data types of the columns

Data Collection – Fetching Data using Pandas

Common Panda functions

Accessing/processing:

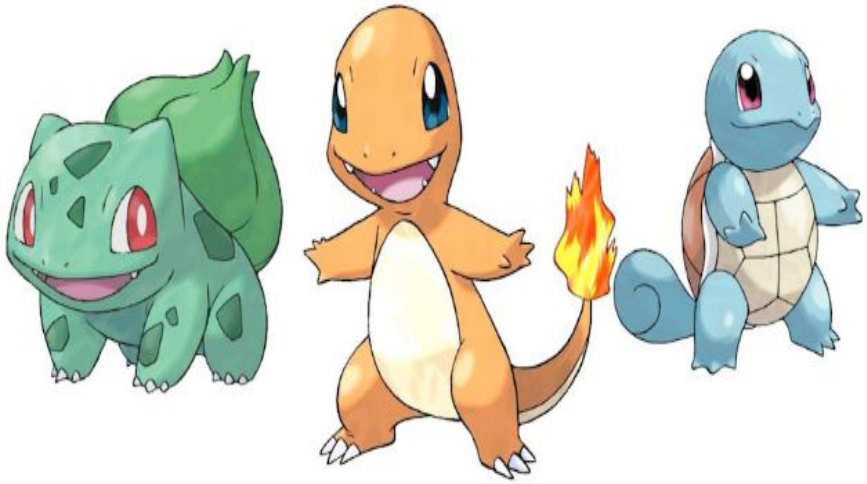
- df[“column_name”]
- Df.column_name
- .max(), .min(), .idxmax(), .idxmin()
- <dataframe> <conditional statement>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()

Download the data

Download the Pokemon dataset from:

https://github.com/innoviai/ipa_courses/blob/main/Lecture%203/pokemon_dataset.csv

Save the pokemon dataset file in a location you'll remember.



Reading in the data

First we import the Python packages we are going to use.

Then we use Pandas to load in the dataset as a data frame.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

df1 = pd.read_csv("filepath/to/where/data/is/stored/pokemon_dataset.csv",
index_col=0, encoding="ISO-8859-1")
```

NOTE: The argument `index_col` argument states that we'll treat the first column of the dataset as the ID column.

NOTE: The encoding argument allows us to by pass an input error created by special characters in the data set.

Examine the data set

```
print(df1.head())
```

```
         Name Type 1  Type 2  Total  ...  Sp. Def  Speed  Stage  Legendary
#                                    ...
1   Bulbasaur  Grass  Poison    318  ...       65     45      1      False
2     Ivysaur  Grass  Poison    405  ...       80     60      2      False
3    Venusaur  Grass  Poison    525  ...      100     80      3      False
4  Charmander   Fire     NaN    309  ...       50     65      1      False
5  Charmeleon   Fire     NaN    405  ...       65     80      2      False
```

```
print(df1.describe())
```

```
            Total          HP      Attack  ...     Sp. Def       Speed       Stage
count  151.00000  151.000000  151.000000  ...  151.000000  151.000000  151.000000
mean   407.07947   64.211921   72.549669  ...   66.019868   68.933775    1.582781
std     99.74384   28.590117   26.596162  ...   24.197926   26.746880    0.676832
min    195.00000   10.000000    5.000000  ...   20.000000   15.000000    1.000000
25%    320.00000   45.000000   51.000000  ...   49.000000   46.500000    1.000000
50%    405.00000   60.000000   70.000000  ...   65.000000   70.000000    1.000000
75%    490.00000   80.000000   90.000000  ...   80.000000   90.000000    2.000000
max    680.00000  250.000000  134.000000  ...  125.000000  140.000000    3.000000
```

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

THANK YOU