

# LGT3109

# Introduction to Coding for

# Business with Python

## (class 1)

Xiaoyu Wang  
Dept. of LMS, PolyU

# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

# Our Goal

- Grasp the fundamentals of Python language and the **basics of coding**.
- Read and analyze **basic** Python programs;
- Develop, test and debug **basic** Python programs.



# Our Goal

- Be familiar with and be able to apply **basic knowledge** and **skills** of Python programming for **basic business applications**.
- Understand **basic business applications** of Python programs;
- Apply Python programming for **basic business applications** in **task automation** and **data management**.



# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

# Basic Information

- Course name: Introduction to Coding for Business with Python.
- Delivery channel: Face to Face/Video recorded.
- Course Website: Poly U Blackboard (LEARN@PolyU 理學網)  
<http://learn.polyu.edu.hk>
- Instructor: WANG, Xiaoyu
- Email: [xiaoyuhk.wang@polyu.edu.hk](mailto:xiaoyuhk.wang@polyu.edu.hk)
- Office: M1007; Tel.: 2766-7369

# Basic Information-Blackboard

- Lectures, tutorials, and class videos.

The screenshot shows the Blackboard Home Page for a course titled "INTRODUCTION TO CODING FOR BUSINESS WITH PYTHON (LGT3109\_20251\_A)". The left sidebar contains links for Home Page, Announcements, Calendar, Syllabus & Schedule, Tools, Discussions, Groups, and Contacts. The main content area is titled "Home Page" and includes sections for "My Announcements" (with a message about no course or organization announcements), "My Tasks" (with a message about no tasks), and "Needs Attention".

INTRODUCTION TO  
CODING FOR BUSINESS  
WITH PYTHON  
(LGT3109\_20251\_A)

Home Page  
Announcements  
Calendar

---

Syllabus & Schedule  
Tools

---

Discussions  
Groups  
Contacts

Home Page

My Announcements

No Course or Organization Announcements

My Tasks

My Tasks:

Needs Attention

# Basic Information-Instructor

- Name: WANG, Xiaoyu
- I am an Assistant Professor in the Department of Logistics and Maritime Studies.
- My research and teaching focuses on Supply Chain Management in consumer privacy and innovative technologies such as FinTech.
- PhD in Operations Management, Olin Business School, WashU.
- Master in Operations Research, UNC at Chapel Hill.
- Bachelor in Finance, Guanghua School of Management, PKU.

# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

# Teaching Arrangement

- Lectures and Tutorials:

Lecture on every Wednesday (8:30am-10:20am), at MN102c

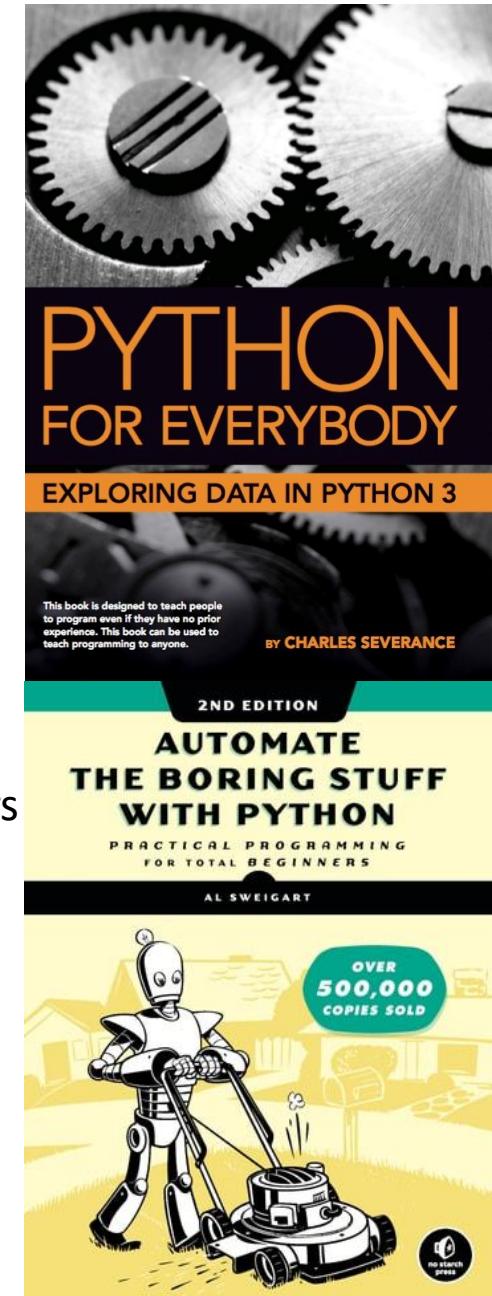
Tutorial on every Wednesday (10:30am-11:20am), at MN102c

- Office Hour:

Saturday afternoon (2:30pm-4:00pm)

# Teaching Arrangement-Textbook

- Python for Everybody: Exploring Data in Python 3
  - Charles Russell Severance, Aimee Andrion , Sue Blumenber, and Elliott Hauser
  - CreateSpace Independent Publishing Platform, 2016
  - Hardcopy available in Amazon
  - Available online for free: <https://www.py4e.com/book.php>
  - About python basics
- Automate the Boring Stuff with Python, 2nd Edition, Practical Programming for Total Beginners
  - Al Sweigart
  - No Starch Press, 2019
  - Hardcopy available in PolyU bookstore
  - Available online for free: <https://automatetheboringstuff.com/#toc>
  - About applications in task automation



# Teaching Arrangement-Schedule

## Tentative Schedule of LGT3109 (2025-26 Semester 1)

Week	Date	Topics	Action/Reminder
0	N/A	No teaching. You can test online teaching systems, download and install Anaconda Navigator, and set up the coding environment on your own devices.	<i>Read documents in "Week 0" at Blackboard</i>
1	Sep. 2	Lecture: Course Introduction: Python and applications to business <i>Tutorial 1: Coding environment for Python</i>	
2	Sep. 9	Lecture: Variables and Simple Data Types <i>Tutorial 2: Variables and Simple Data Types</i>	Due: Tutorial 1 Report
3	Sep. 16	Lecture: Basic Flow Control: Conditions and Flow Chart <i>Tutorial 3: Conditions and Flow Chart</i>	Due: Tutorial 2 Report
4	Sep. 23	Lecture: Functions <i>Tutorial 4: Functions</i>	Due: Tutorial 3 Report
5	Sep. 30	Lecture: Basic Flow Control: Iterations <i>Tutorial 5: Iterations</i>	Due: Tutorial 4 Report
6	Oct. 7	Mid-Autumn Festival	
7	Oct. 14	Lecture: String and Files <i>Tutorial 6: Strings and Files</i>	Due: Tutorial 5 Report
8	Oct. 21	Lecture: Lists <i>Tutorial 7: Lists</i>	Due: Tutorial 6 Report <i>Release: Individual Assignment</i>
9	Oct. 28	Lecture: Dictionaries and Other Structuring Data <i>Tutorial 8: Dictionaries and Other Structuring Data</i>	Due: Tutorial 7 Report
10	Nov. 4	Unscored In-class Exercises Mid-term Review	Due: Tutorial 8 Report
11	Nov. 11	Lecture: Task Automation for Business: Organizing Files <i>Tutorial 9: Organizing Files</i>	
12	Nov. 18	Lecture: Basic Data Management: Acquiring and Exploring Data <i>Tutorial 10: Acquiring and Exploring Data</i>	Due: Tutorial 9 Report
13	Nov. 25	Lecture: Testing and Debugging Python Programs and Course Review	Due: Tutorial 10 Report <i>Due: Individual Assignment</i>
Final Exam		TBD	

# Teaching Arrangement-Content

Key Aspects	Lecture and Tutorial Topics
Python Fundamentals	<p>Getting Started: What's the use of Python? How to type and execute Python programs?</p> <p>Variables and Simple Data Types</p> <p>Basic Flow Control</p> <p>Functions</p> <p>Strings, Lists, and Dictionaries</p> <p>Testing and Debugging Python Programs</p> <p>Reading, Writing , and Organizing Working Files</p>
Business Applications of Python: Task Automation in Business Operations	Acquiring and Exploring Data

# Teaching Arrangement-Other Issues

- Lecture slides will be available on [Blackboard](#).
  - Save you some time taking notes.
- Plagiarism is at **zero tolerance** (exams, tutorials, and assignments).
- Be **quiet** and do not **disturb** others in class.
  - You can eat food, drink water, go to restroom. However, do it gently!
  - You can ask question without raising your hand.

# Part 1: Course Introduction

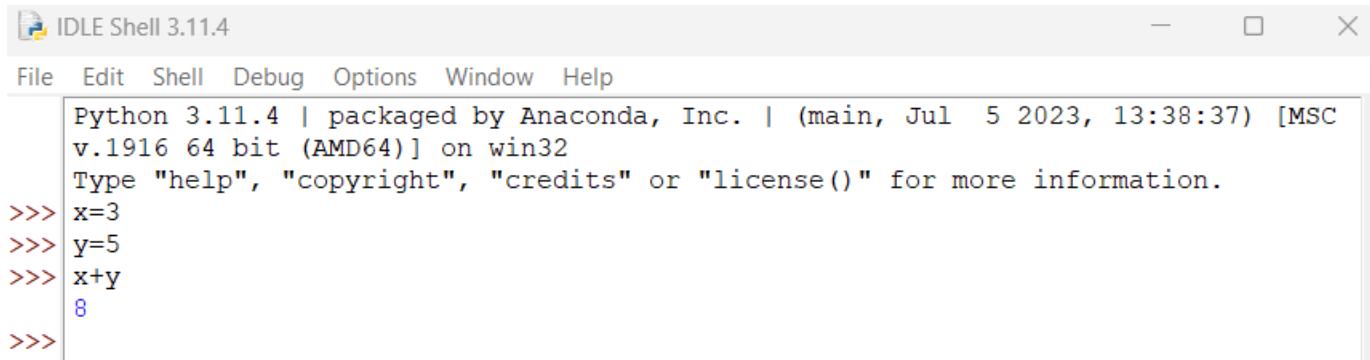
- The goal of the course
- Course basic information
- Arrangement for teaching
- **Coding environment**
- Lecture and tutorial
- Grading

# Software We Use-Shell/Editor

- Some definition clarifications.
- Are you confused with words like **Python**, **IDLE**, **Shell**, **Jupyter**...
- **Python** is a programming **language** (communicate with machine).
- A **shell** is a command-line **interface** that allows you to interact with the operating system.
- A **text editor** is a program that allows you to edit text files/scripts. Text editors usually provide more features and a more user-friendly interface.

# Software We Use-Shell/Editor

- In shell, code is executed one line at a time.
- In editor, multiply codes can be executed once.
- Editors allow you to open **multiple** files at once.

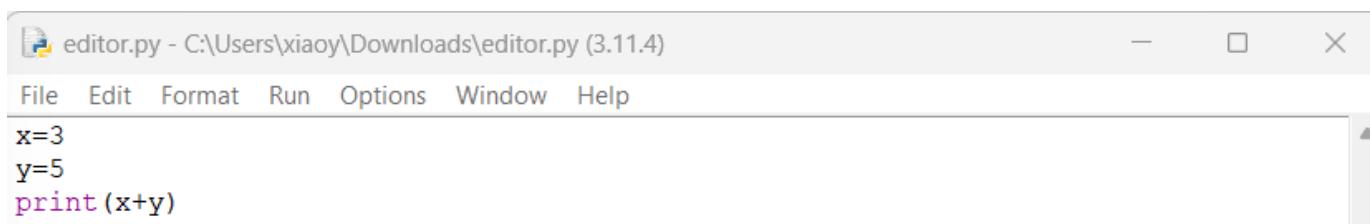


IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

```
>>> x=3
>>> y=5
>>> x+y
8
>>>
```



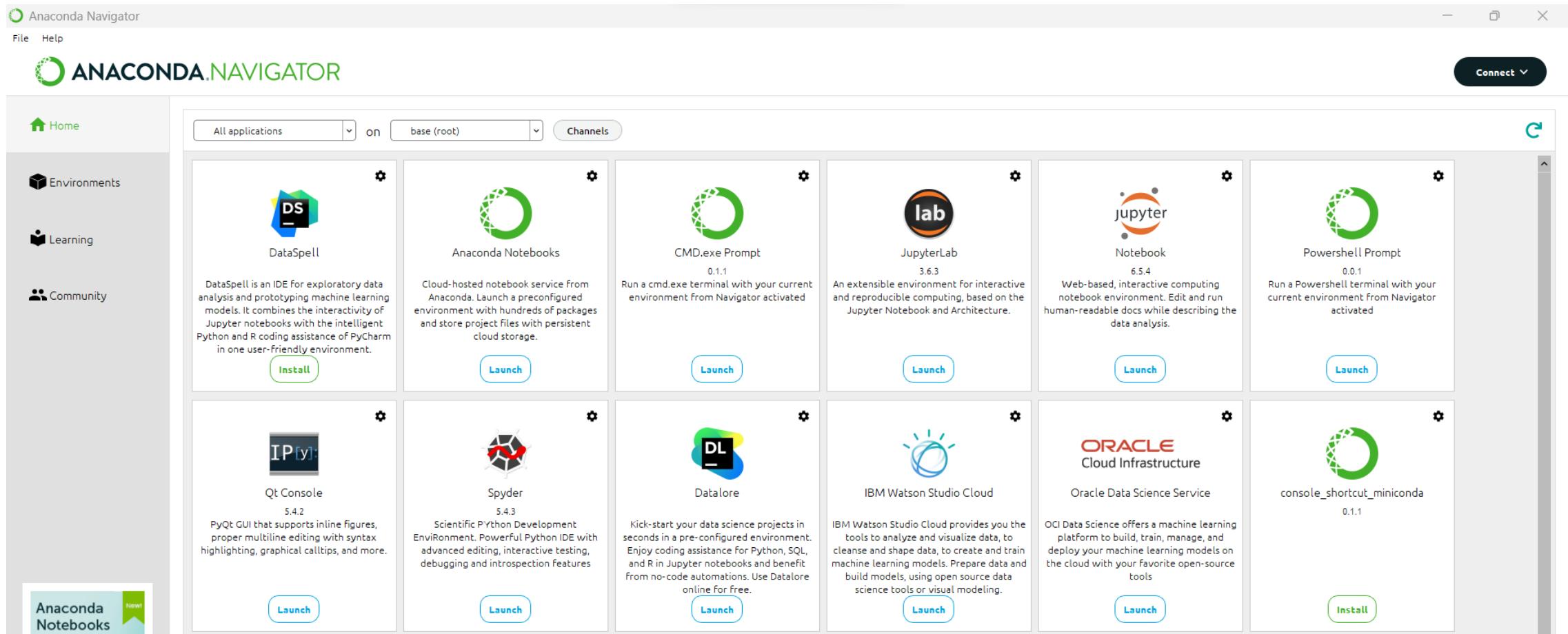
editor.py - C:\Users\xiaoy\Downloads\editor.py (3.11.4)

File Edit Format Run Options Window Help

```
x=3
y=5
print(x+y)
```

# Software We Use-Environment

- An **environment** is needed: **IDLE, Jupyter Notebook, Spyder...**



# Software We Use-Environment

- An **environment** is needed: **IDLE**, **Jupyter Notebook**, **Spyder**...
- A Python **shell** is a command-line interface.
- An **IDLE** (Integrated Development and Learning Environment) is a more advanced tool, which **includes** more features such as a **code editor**, **debugging tools**, and a built-in **Python shell**.

# Software We Use-Flow

- Anaconda Navigator

- Python and R environment and applications.
- Anaconda Navigator → Environment → Shell and Editor
- IDLE environment (Classroom) → Shell or Editor (Mouth)  
→ Python (Language)

# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

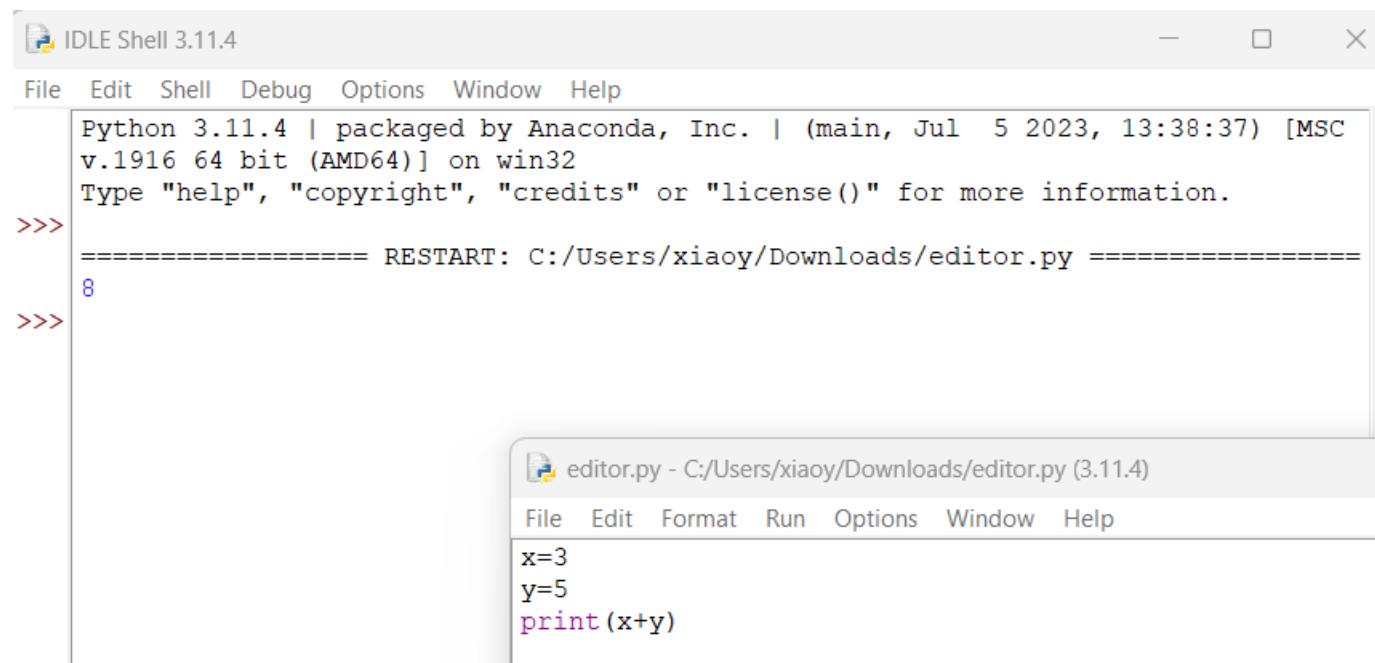
# Lectures and Tutorials

- Lecture + Tutorial per week
- The lecture focuses on **basic concepts**
- Tutorials focus on **coding exercises** with Python
- A report is required for **each** tutorial

# Lectures and Tutorials-Coding Environment

- For Lectures:

➤ **IDLE** (integrated development environment for Python)



The image shows two windows of the IDLE Python development environment. The top window is the 'IDLE Shell 3.11.4' window, which displays the Python interpreter's prompt ('>>>') and the output of the command 'print(x+y)'. The bottom window is the 'editor.py - C:/Users/xiaoy/Downloads/editor.py (3.11.4)' window, showing the source code of a Python file named 'editor.py'.

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/xiaoy/Downloads/editor.py =====

8

editor.py - C:/Users/xiaoy/Downloads/editor.py (3.11.4)

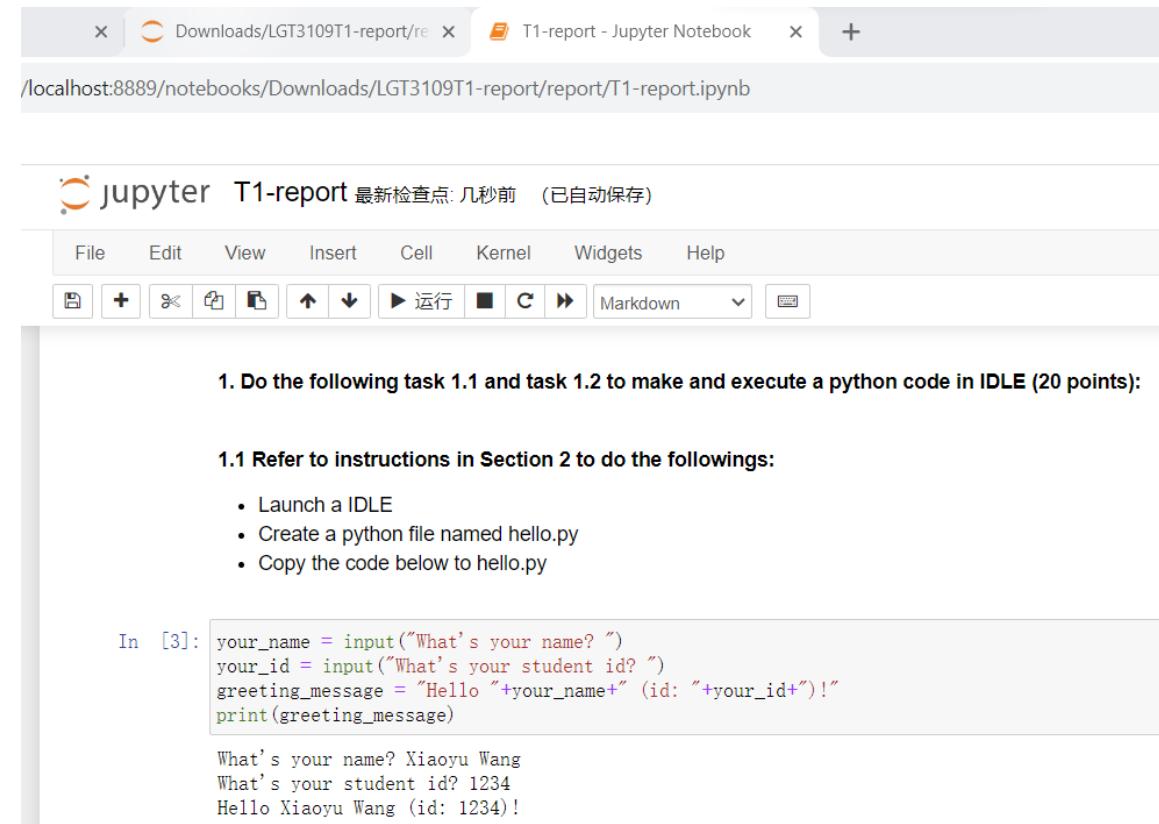
File Edit Format Run Options Window Help

```
x=3
y=5
print(x+y)
```

# Lectures and Tutorials-Coding Environment

- For Tutorials:

➤ **Jupyter Notebook (Interactive Development Environment)**



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

- Title Bar:** Downloads/LGT3109T1-report/re × T1-report - Jupyter Notebook × +  
/localhost:8889/notebooks/Downloads/LGT3109T1-report/report/T1-report.ipynb
- Toolbar:** File Edit View Insert Cell Kernel Widgets Help
- Cell Content:**
  1. Do the following task 1.1 and task 1.2 to make and execute a python code in IDLE (20 points):
  - 1.1 Refer to instructions in Section 2 to do the followings:
    - Launch a IDLE
    - Create a python file named hello.py
    - Copy the code below to hello.py

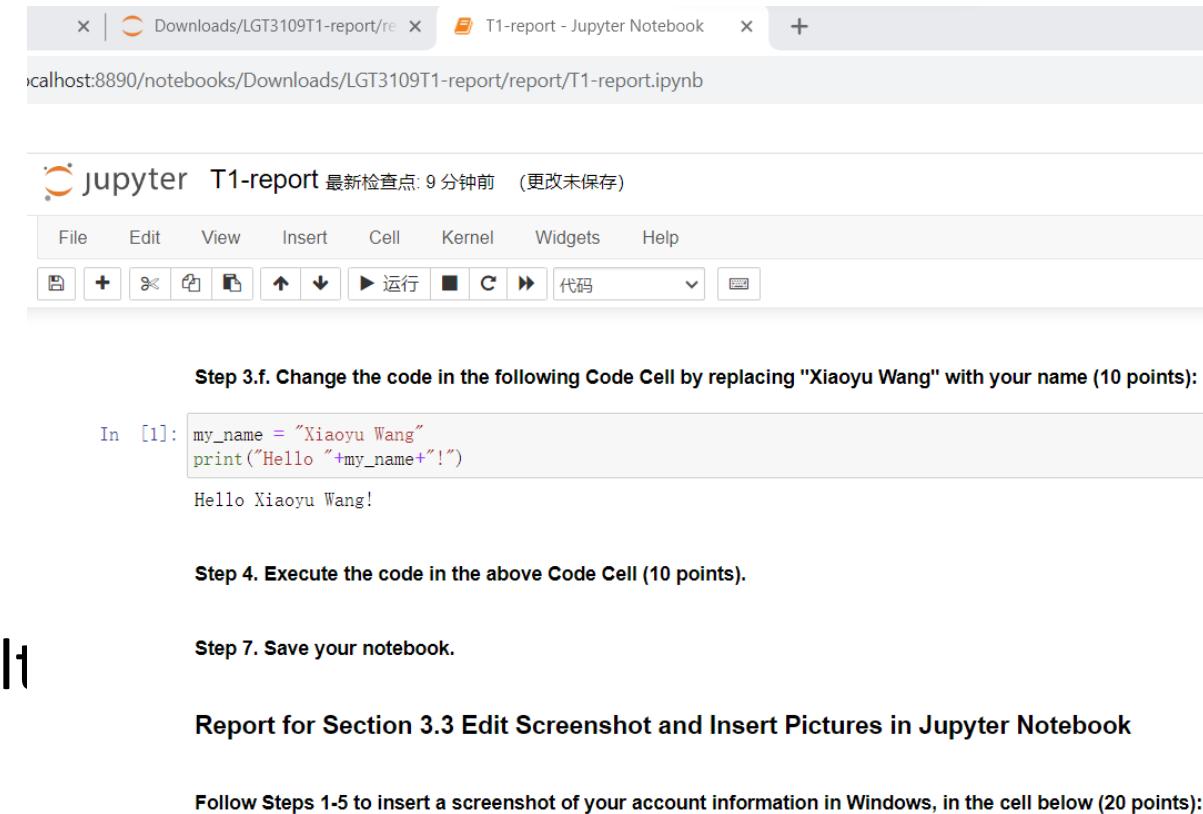
```
In [3]: your_name = input("What's your name? ")
your_id = input("What's your student id? ")
greeting_message = "Hello " + your_name + " (id: " + your_id +")!"
print(greeting_message)
```

Output:

```
What's your name? Xiaoyu Wang
What's your student id? 1234
Hello Xiaoyu Wang (id: 1234)!
```

# Lectures and Tutorials-Report Submission

- Submit a report for each tutorial on **Blackboard**.
- Complete each tutorial and report **independently**.
- A **sample report** is provided and you only need to fill in some of the contents (your answers, codes, result etc.) through **Jupyter Notebook**
- **No late submission will be accepted!**



The screenshot shows a Jupyter Notebook interface with the title "T1-report - Jupyter Notebook". The URL in the address bar is "localhost:8890/notebooks/Downloads/LGT3109T1-report/report/T1-report.ipynb". The notebook contains the following content:

```
In [1]: my_name = "Xiaoyu Wang"
print("Hello "+my_name+"!")
```

Hello Xiaoyu Wang!

Step 3.f. Change the code in the following Code Cell by replacing "Xiaoyu Wang" with your name (10 points):

Step 4. Execute the code in the above Code Cell (10 points).

Step 7. Save your notebook.

Report for Section 3.3 Edit Screenshot and Insert Pictures in Jupyter Notebook

Follow Steps 1-5 to insert a screenshot of your account information in Windows, in the cell below (20 points):

# Lectures and Tutorials-Tutorial Exercises

- Basics about Python Coding:

- Tutorial 1: Coding environment for Python
- Tutorial 2: Variables and simple data types
- Tutorial 3: Conditions and flowcharts
- Tutorial 4: Functions
- Tutorial 5: Iterations
- Tutorial 6: Strings and Files
- Tutorial 7: Lists
- Tutorial 8: Dictionaries and Structured Data

- Basic Applications:

- Tutorial 9: Organizing Files
- Tutorial 10: Acquiring and Exploring Data

# Part 1: Course Introduction

- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- **Grading**

# Grading

- Class Participation (discussion, question and answer, participation): 10%
- Tutorial Reports: 30%
- Individual Assignment: 10%
- Final Exam (Open Notes): 50%

# Grading- Class Participation 10%

- Class attendance.

# Grading-Tutorial Reports 30%

- A sample report including instructions will be provided
- Follow the instruction to do the tutorial, and fill in the missing contents in the sample report to make your report
  - Fill in the answers to questions
  - Insert/Change codes
  - Insert results (screenshots)
  - Late submission is not accepted

# Grading-Individual Assignment 10%

- Questions will be released in Week 8 (October 21)
- More than a month to complete the assignment (Due: November 25)
  - Late submission is not accepted.

# Grading-Final Exam 50%

- Final Exam is on computer, open notes, with an internet connection only allowing to access Blackboard
- In Week 10 (November 4), there will be some unscored in class exercises, with questions and formats like those of the final exam.

# Part 2: Motivation for Programming

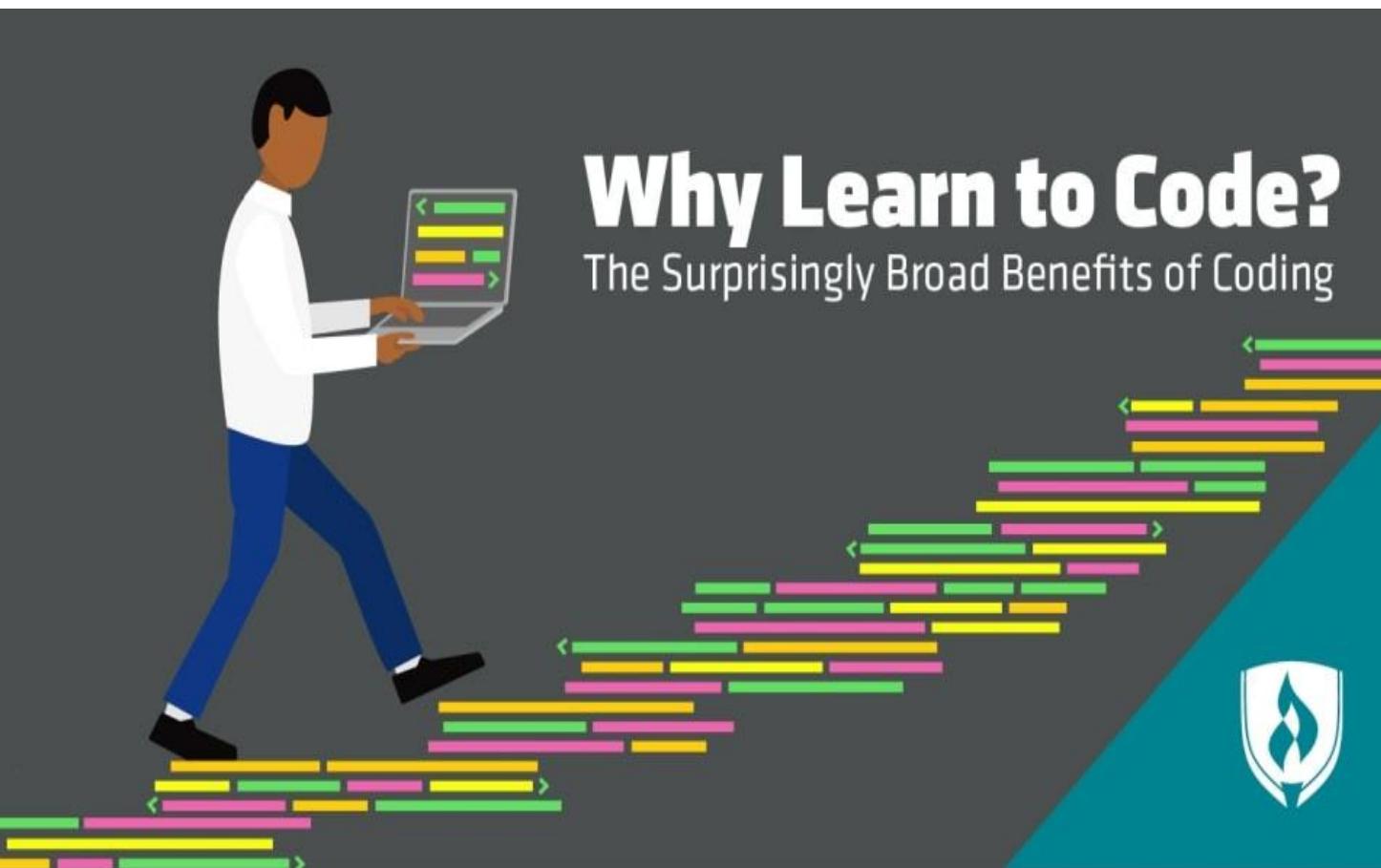
- Why coding
- What's code
- What's Python
- Components of Python Code

# Part 2: Motivation for Programming

- Why coding
- What's code
- What's Python
- Components of Python Code

# Why Coding

- Discussion: Why we want to learning coding and programming?



The image shows a man in a white shirt and blue pants walking up a staircase. The stairs are composed of various colored horizontal bars (green, yellow, pink) with small arrows pointing in different directions, representing code snippets. To the right of the staircase is a computer screen displaying a Python IDLE Shell and a code editor window.

**IDLE Shell 3.11.4**

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\teaching\LGT3109 Introduction to Coding for Business with Python\Lecture1\class_code\freq_word.py
Enter file:words.txt
Python 9
```

**freq\_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\teaching\L...**

```
File Edit Format Run Options Window Help
import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

# Why Coding-Case 1 Counting Words

- Help me find the most frequently used word and the how many time it appears in the article.

The screenshot shows a window titled "words" with a toolbar at the top featuring a file icon, a search bar with the text "words", and a "+" button. Below the toolbar is a menu bar with "File", "Edit", "View", and a settings gear icon. The main content area contains four paragraphs of text:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages.

# Why Coding-Case 1-Human Way

- How do you find the word?
  - Read the passage from the beginning to the end.
  - Write down when you see a **new word**: (new word: **1**).
  - Add 1 seeing an **old word**. From (old word: **x**) to (old word: **x+1**).
  - After finishing counting, pick the one most counted.

# Why Coding-Case 1-Find the Largest

- After reading all the article, computer has a full list.

- List: {[“Python”,9], [“is”,3],.....}
- Pick up the one with the largest count number.

```
bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

The screenshot shows two windows from the Python IDLE Shell. The top window is titled 'IDLE Shell 3.11.4' and displays the command-line interface. The bottom window is titled 'freq\_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\teaching\LGT3109 Introduction to Coding for Business with Python\Lecture1\class\_code\freq\_word.py' and shows the source code for a script named 'freq\_word.py'. A red box highlights the final part of the code where the logic for finding the largest count is implemented.

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\teaching\LGT3109 Introduction to Coding for Business with Python\Lecture1\class_code\freq_word.py
Enter file:words.txt
Python 9

>>>
import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

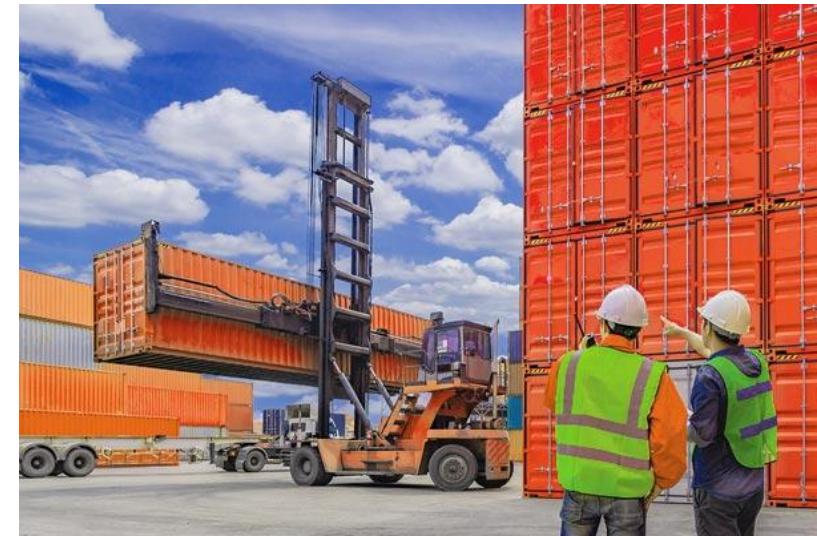
for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

# Why Coding-Case 2 Combine File

- Foxconn assembles and exports PC worldwide.
- Request for information on cargo demand: how many to ship.  
(10 containers from SH to SF in Feb.)
- Request for quotation: what's the price to ship. (1000HKD/container)
- Allocation and Contract.
- Execution: Shipment of products.



# Why Coding-Case 2-Demand Information

- Each business unit needs to submit forecasted demand requests of every month for each pair of origin and destination ports

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	BU	OP	DP	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
2	SH	HK	LA	20	10	20	20	15	15	30	30	40	40	20	10
3	SH	SH	LA	20	5	20	20	15	10	30	30	40	40	20	10
4	SH	SIN	SF	5	10	20	5	15	15	30	20	40	40	20	20

- BU: business unit ID.
- OP: original port
- DP: destination port
- JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC: the forecasted number of containers of each month

# Why Coding-Case 2-Mission Combine

- As a procurement manager, you receive demand requests in **excel files** from 100 business units of your company worldwide.
- To prepare documents for transportation service procurement, you need to first **combine** all the requests into **one excel file**.
- How are you going to do it?

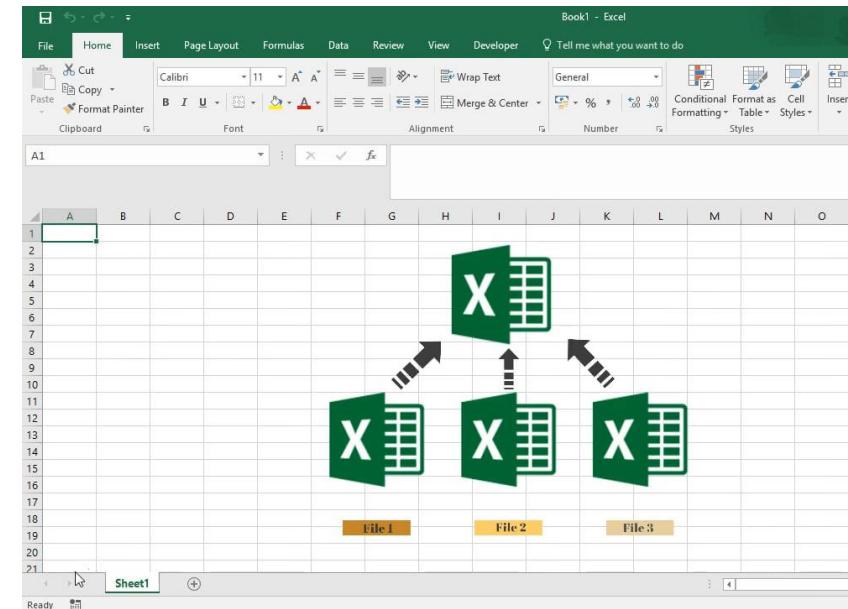
	A	B	C	D	E	F	G	H
1	BU	OP	DP	JAN	FEB	MAR	APR	MAY
2	SH	HK	LA	20	10	20	20	15
3	SH	SH	LA	20	5	20	20	15
4	SH	SIN	SF	5	10	20	5	15

	A	B	C	D	E	F	G	H
1	BU	OP	DP	JAN	FEB	MAR	APR	MAY
2	SZ	HK	LA	20	10	20	20	15
3	SZ	SH	LA	20	5	20	20	15
4	SZ	SIN	SF	5	10	20	5	15

	A	B	C	D	E	F	G	H
1	BU	OP	DP	JAN	FEB	MAR	APR	MAY
2	SH	HK	LA	20	10	20	20	15
3	SH	SH	LA	20	5	20	20	15
4	SH	SIN	SF	5	10	20	5	15
5	SZ	HK	LA	20	10	20	20	15
6	SZ	SH	LA	20	5	20	20	15
7	SZ	SIN	SF	5	10	20	5	15

# Why Coding-Case 2-How Do you Achieve

- Problem: **combine** 100 excel files into one excel file.
- Discussion: How to do?



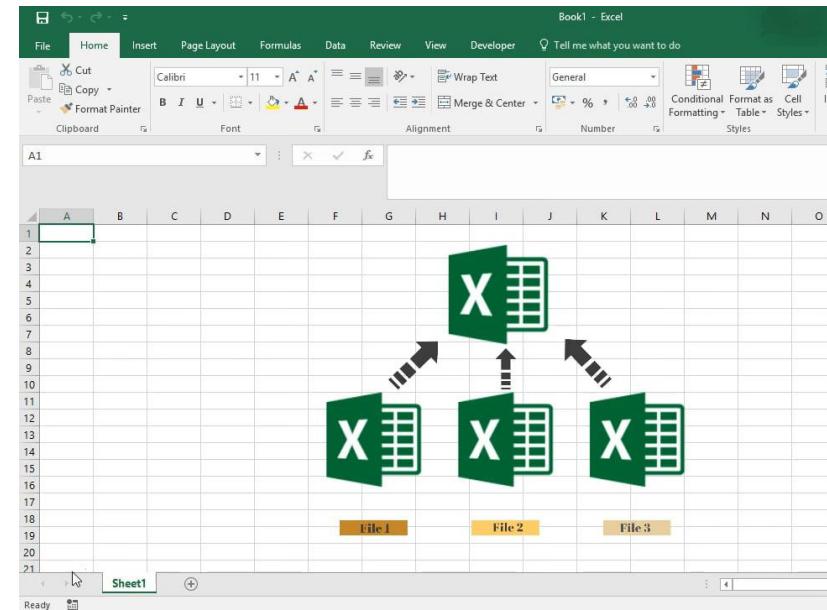
# Why Coding-Case 2-How Do you Achieve

- Problem: **combine** 100 excel files into one excel file.
- Discussion: How to do?

- Manually combine.
- Use software like Ultimate Suite. (costly)
- Can do that in Excel. (complex)

<https://techengage.com/tools-merge-excel-files/>

- Python!



# Why Coding-Case 2-Python Code

- Piece of cake!

➤ Read data.

```
import os
import pandas as pd
cwd = os.path.abspath('')
files = os.listdir(cwd)
```

➤ Handle data in order.

```
df = pd.DataFrame()
for file in files:
```

➤ Process data and combine.

```
if file.startswith('Demand') and file.endswith('.xlsx'):
    df = pd.concat([df,pd.read_excel(file)], ignore_index=True)
```

➤ Export file.

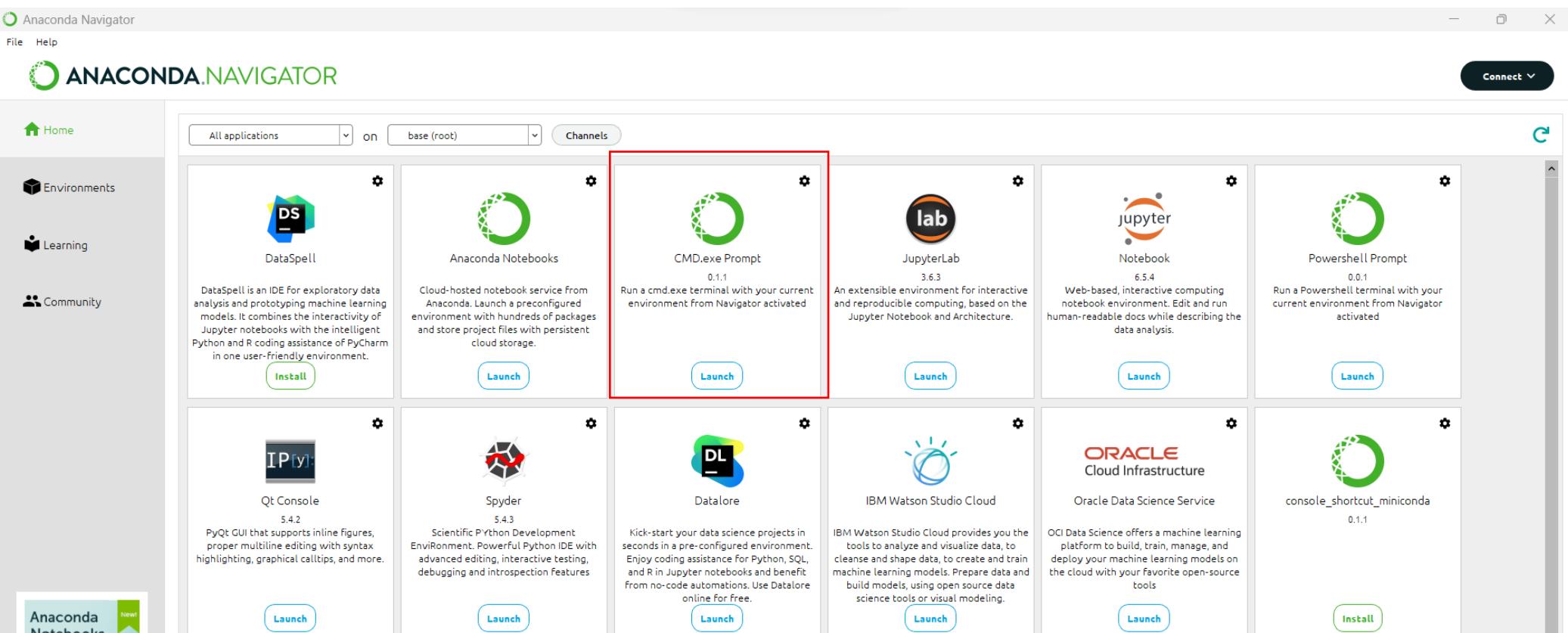
```
df.to_excel('combined_demand.xlsx',index=False)
```

```
File Edit Format Run Options Window Help
import os
import pandas as pd
cwd = os.path.abspath('')
files = os.listdir(cwd)

df = pd.DataFrame()
for file in files:
    if file.startswith('Demand') and file.endswith('.xlsx'):
        df = pd.concat([df,pd.read_excel(file)], ignore_index=True)
df.to_excel('combined_demand.xlsx',index=False)
```

# Why Coding-Case 2-How to Run

- Open Anaconda Navigator.
- Open the terminal CMD.exe Prompt.



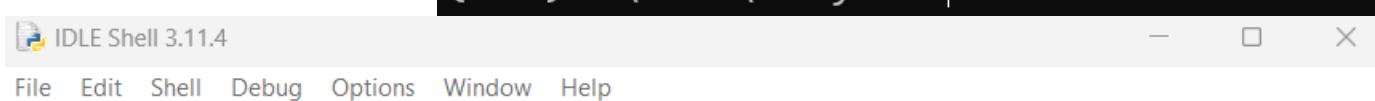
# Why Coding-Case 2-How to Run

- In terminal, open IDLE.
- In IDLE, open the code file.
- Run the code.

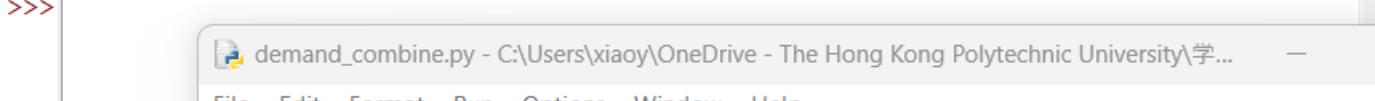
➤**Attention:** by default, the path is the where your file is.



```
C:\Windows\system32\cmd.exe + ^ Microsoft Windows [Version 10.0.22621.2215] (c) Microsoft Corporation. All rights reserved. (base) C:\Users\xiaoy>IDLE
```



```
IDLE Shell 3.11.4 File Edit Shell Debug Options Window Help Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license()" for more information.
```



```
demand_combine.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学... File Edit Format Run Options Window Help >>> import os import pandas as pd cwd = os.path.abspath('') files = os.listdir(cwd)
```

```
df = pd.DataFrame() for file in files: if file.startswith('Demand') and file.endswith('.xlsx'): df = pd.concat([df,pd.read_excel(file)], ignore_index=True) df.to_excel('combined_demand.xlsx',index=False)
```

# Why Coding

- Discussion: Why we want to learning coding and programming?
- Making livings (develop software)
- Saving costs and time
- Solving problems
- Having fun
- Helping others

```
content: "";
content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
button, input, select, textarea { margin: 0 }
:focus { outline: 0 }
a:link { -webkit-tap-highlight-color: #FF5E99 }
img, video, object, embed {
    max-width: 100%;
    height: auto!important;
}
iframe { max-width: 100% }
blockquote {
    font-style: italic;
    font-weight: normal;
    font-family: Georgia,Serif;
    font-size: 15px;
    padding: 0 10px 20px 27px;
    position: relative;
    margin-top: 25px;
}
blockquote:after {
    position: absolute;
    content: '"';
}
blockquote p { margin-bottom: 10px }
strong, b { font-weight: bold }
em, i, cite {
    font-style: normal;
    font-family: arial;
}
small { font-size: 100% }
figure { margin: 10px 0 }
code, pre {
    font-family: monospace,consolas,sans-serif;
    font-weight: normal;
    font-style: normal;
}
pre {
    margin: 5px 0 20px 0;
    line-height: 1.3em;
    padding: 8px 10px;
    overflow: auto;
}
img {
    border: 1px solid black;
    border-radius: 10px;
    width: 100px;
    height: 100px;
    margin: 0 8px;
    vertical-align: middle;
    font-size: 1.5;
}
code {
    border: 1px solid black;
    padding: 0 2px;
    background-color: black;
    color: white;
}
```



# Part 2: Motivation for Programming

- Why coding
- What's code
- What's Python
- Components of Python Code

# What's Code

- Code, program, and software:
  - Code: a sequence of stored **instructions**.
  - Program: a piece of code or a **set of instructions** that performs a given task.
  - Software: a **set of programs** used to perform a complex task.
  - Software > Program > Code

# What's Code-Case 1 Revisit

- We encode and solve a problem. The coded program **saves time and energy, and it is reusable!**

The image shows two separate code editor windows side-by-side. The top window is titled 'words' and contains text about Python. The bottom window is titled 'words2' and contains text about MATLAB.

**words:**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages.

**words2:**

MATLAB (an abbreviation of "MATrix LABoratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

As of 2020, MATLAB has more than 4 million users worldwide. They come from various backgrounds of engineering, science, and economics. As of 2017, more than 5000 global colleges and universities use MATLAB to support instruction and research.

# What's Code-Case 1 Revisit

- Just input the file needed.
- The **same** code, but with **Different** input file.
- This coded program instructs PC **hardware** to want in order.

The image shows a screenshot of a Windows desktop environment. On the left, the Python IDLE Shell 3.11.4 window is open, displaying a command-line interface. The command `freq_word.py` is run, followed by the command `Enter file:words2.txt and 8`. On the right, a code editor window titled "freq\_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学..." displays the Python script source code. The script uses file processing and a dictionary to count word frequencies and find the most frequent word.

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学习\4
polyu\teaching\LGT3109 Introduction to Coding for Business with Python\Lecture1\
class_code\words\freq_word.py
Enter file:words2.txt
and 8

>>>

import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

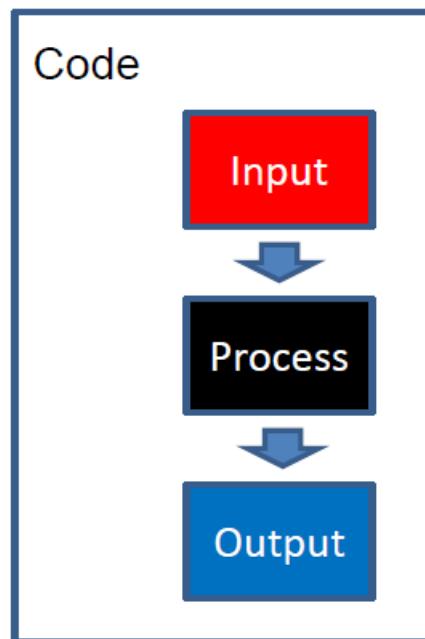
for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

# What's Code

- **Code** is a sequence of **instructions** for computers to work as the tool.
- **Coding** is to build tools for **automation** of information processing.



```
name = input('Enter file:')  
file = open(name, 'r')  
  
counts = dict()  
for line in file:  
    words = line.split()  
    for word in words:  
        counts[word] = counts.get(word, 0) + 1  
  
bigcount = None  
bigword = None  
for word, count in list(counts.items()):  
    if bigcount is None or count > bigcount:  
        bigword = word  
        bigcount = count  
  
print(bigword, bigcount)
```

# Part 2: Motivation for Programming

- Why coding
- What's code
- **What's Python**
- Components of Python Code

# What's Python

- **Code** is a sequence of **instructions** for computers to work as the tool.
- **Coding** is to build tools for **automation** of information processing.
- **Python** is a coding language for **expression** of the instructions



# What's Python-Advantage

- Python is easy to learn, and code is very readable.
- Python code is more concise than most of the programming languages.
- Python is open source and can be used for free.
- Python provides extensive support libraries of reusable codes.
- The popularity of Python is growing rapidly.

```
print("Hello World")
```

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

C

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```

C++

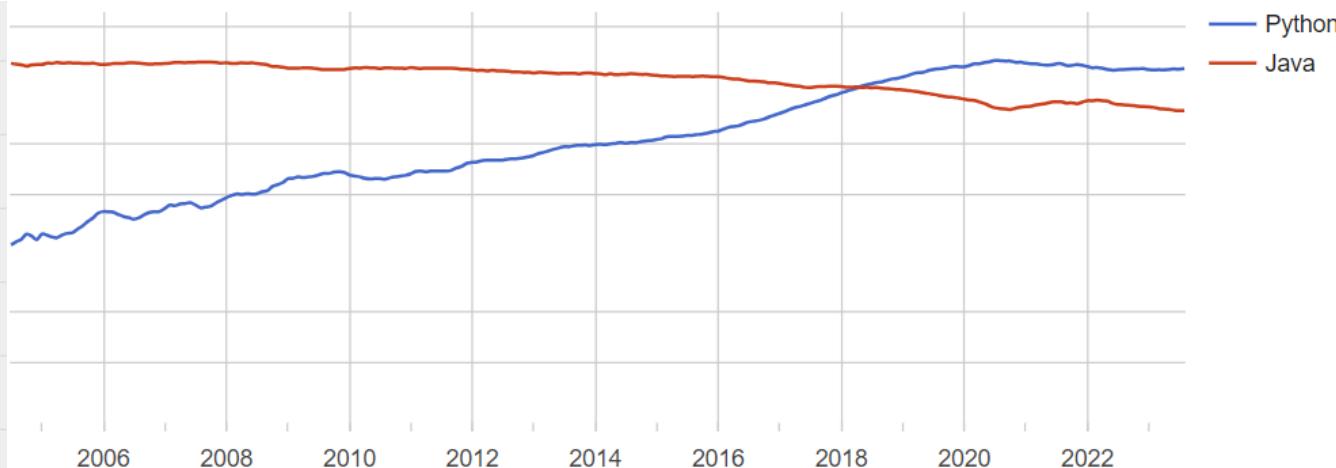
```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Prints the string to the
        console.
    }
}
```

Java

# What's Python-Popularity

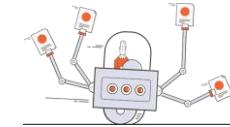
- PYPL index:

Worldwide, Aug 2023 :				
Rank	Change	Language	Share	1-year trend
1		Python	28.04 %	+0.3 %
2		Java	15.78 %	-1.3 %
3		JavaScript	9.27 %	-0.2 %
4		C#	6.77 %	-0.2 %
5		C/C++	6.59 %	+0.4 %
6		PHP	5.01 %	-0.4 %
7		R	4.35 %	+0.0 %
8		TypeScript	3.09 %	+0.3 %
9	↑↑	Swift	2.54 %	+0.5 %
10		Objective-C	2.15 %	+0.1 %



# What's Python-Wide Application

- **Web\Software Development:**
  - Instagram, Google, Netflix, Uber, Dropbox ...
- **Task Automation:**
  - Python is easy to build little scripts to automate tasks.
- **Data science:**
  - Plenty of Python modules: Data analysis/visualizations.
- Machine learning, Blockchain, Fintech...



# What's Python-Syntax Errors

- Python language is a way to communicate our instructions to Python.
- At start we will make lots of mistakes.
- Learn by doing!



```
>>> x=3
>>> y=5
>>> x>y
SyntaxError: invalid syntax
```

```
>>> x=3
>>> y=5
>>> x<y
True
>>> x>7
False
>>>
```

# Part 2: Motivation for Programming

- Why coding
- What's code
- What's Python
- Components of Python Code

# Components of Python Code

- Vocabulary/Words:

- Variables and Reserved words

- Sentence structure:

- Statement and script

The image shows two windows from the IDLE Python shell. The top window is the IDLE Shell 3.11.4 interface, which displays the Python version, build information, and a help message. It also shows the current working directory and a command-line input area with three '>>>' prompts. The bottom window is a code editor showing the contents of a file named 'freq\_word.py'. The code uses file input, string splitting, dictionaries, and loops to count word frequencies from a text file and print the most frequent word.

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
= RESTART: C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学习\4
polyu\teaching\LGT3109 Introduction to Coding for Business with Python\Lecture1\
class_code\words\freq_word.py
Enter file:words2.txt
and 8

>>>

import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

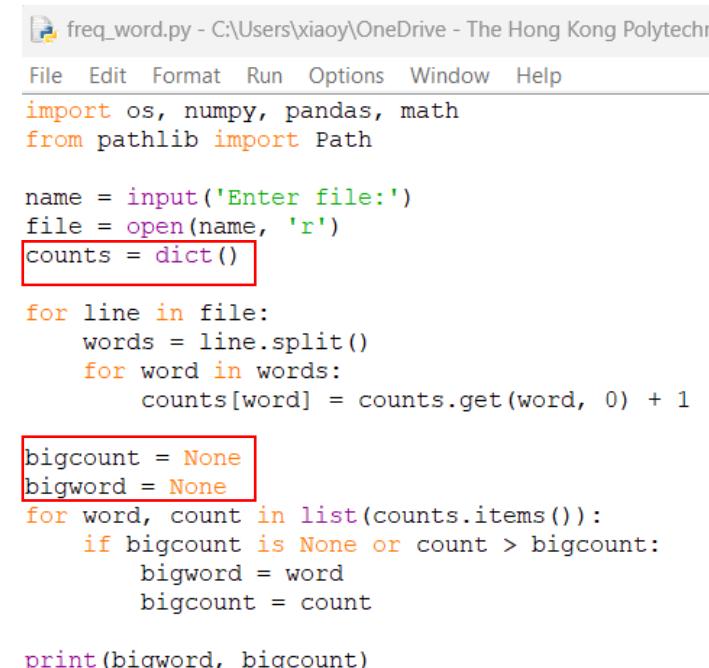
bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

# Components of Python Code-Vocabulary

- Variable:
  - A **variable** is a named place in the memory where a programmer can store data and later retrieve the data using the variable name.
  - You choose variable **names** and can change the contents.
- Counts start from null
- After 1<sup>st</sup> iteration, counts ={["Python",1]}
- After 2<sup>nd</sup> iteration, counts={["Python",1],["is",1]}

```
>>> x=1
>>> x
1
>>> x=2
>>> x
2
>>>
```



```
freq_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University
File Edit Format Run Options Window Help
import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

# Components of Python Code-Vocabulary

- You cannot use reserved words as variable names!

False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	
break	except	in	raise	

- Otherwise, syntax errors

```
>>> false=3
>>> false
3
>>> False=3
SyntaxError: cannot assign to False
>>>
```

# Components of Python Code-Sentence

- A **statement** is a unit of code (or instruction) that Python interpreter can execute.
- A **script** contains a **sequence** of statements.

`x = 2` ← Assignment statement

`x = x + 2` ← Assignment with expression

`print(x)` ← Print statement

Variable

Operator

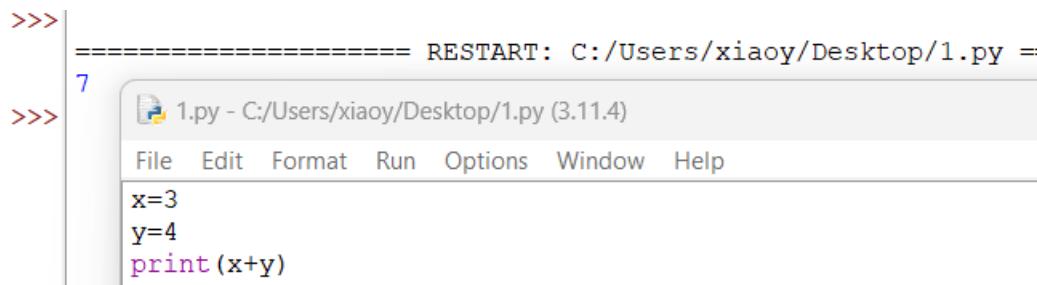
Constant

Function

# Components of Python Code-Write Script

- Shell or text editor?
- Interactive Python (**Shell**)
  - You type directly to Python **one line at a time**, and it responds.
- Script (**editor**)
  - You enter a **sequence of statements** (script) into a file using a text editor and tell Python to execute the statements in the file.

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5  
v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more  
    >>> x=3  
    >>> y=4  
    >>> print(x+y)  
    7
```

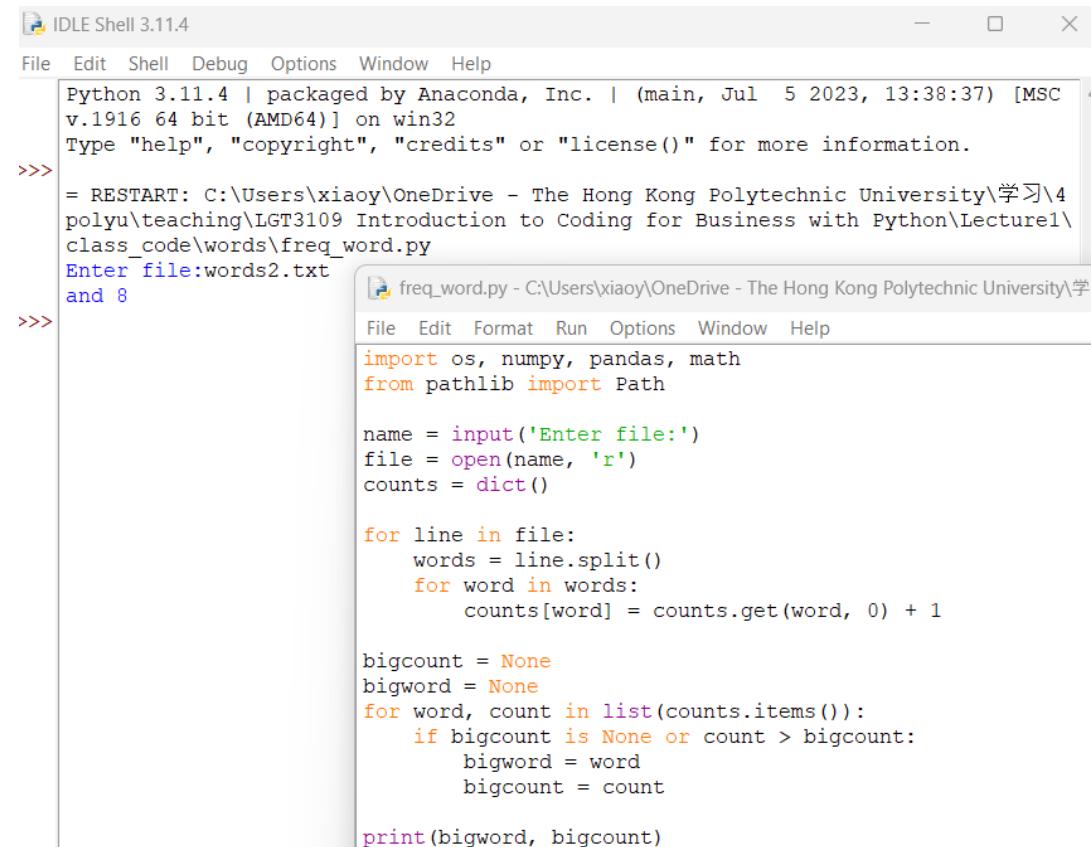


# Components of Python Code-Write Script

- Interactive Python is best for programs of 3 or 4 lines.
- Most programs are much longer, so we type them into a file and tell Python to run the commands in the file.
- As a convention, we add “**.py**” as the suffix on the end of these files to indicate they contain Python. (For example, **words.py**)
- **Attention:** “.txt” will do as well. “.py” is preferred.

# Components of Python Code-Program Steps

- Like a recipe or installation instructions, a program is a **sequence** of steps to be done in order.
- Some steps are **conditional** they may be skipped.
- Sometimes a step or group of steps is to be **repeated**.



The image shows a screenshot of the Python IDLE environment. On the left, there is a script editor window titled "freq\_word.py" containing Python code. On the right, there is a shell window titled "IDLE Shell 3.11.4" showing the execution of the script.

```
freq_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学...\\
File Edit Format Run Options Window Help
import os, numpy, pandas, math
from pathlib import Path

name = input('Enter file:')
file = open(name, 'r')
counts = dict()

for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

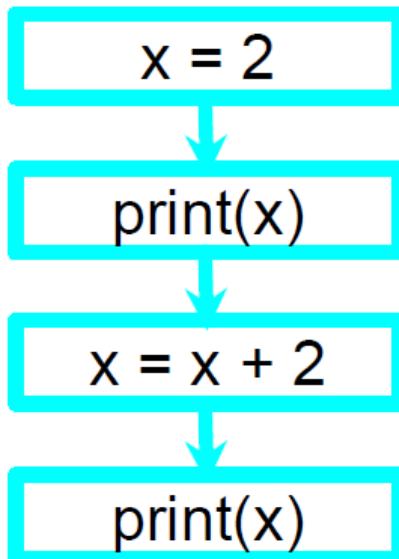
bigcount = None
bigword = None
for word, count in list(counts.items()):
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)

>>>
= RESTART: C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学...\\
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC
v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> Enter file:words2.txt
and 8
```

# Components of Python Code-Program Steps

- Sequential steps



Program:

```
x = 2
print(x)
x = x + 2
print(x)
```

Output:

2

4

- When a program is running, it flows **from one step to the next**. We encode in order and set up paths for the program to follow.

# Components of Python Code-Program Steps

- Conditional steps

Program:

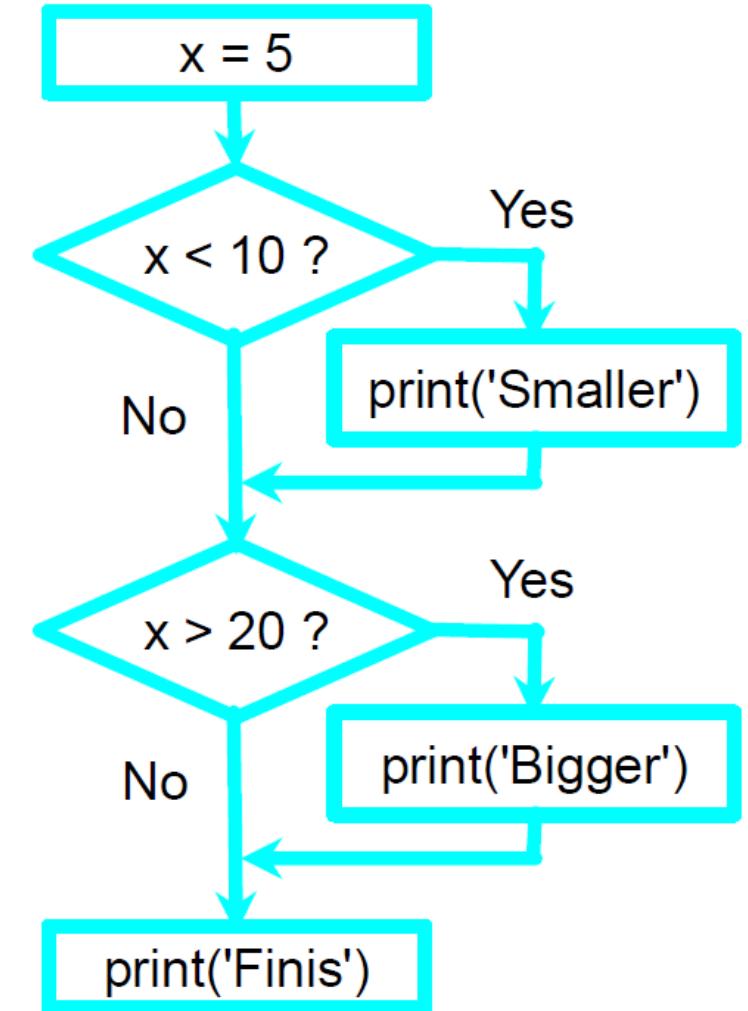
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Output:

Smaller  
Finis

- Some steps are executed only when **certain conditions** are satisfied or not satisfied.



# Components of Python Code-Program Steps

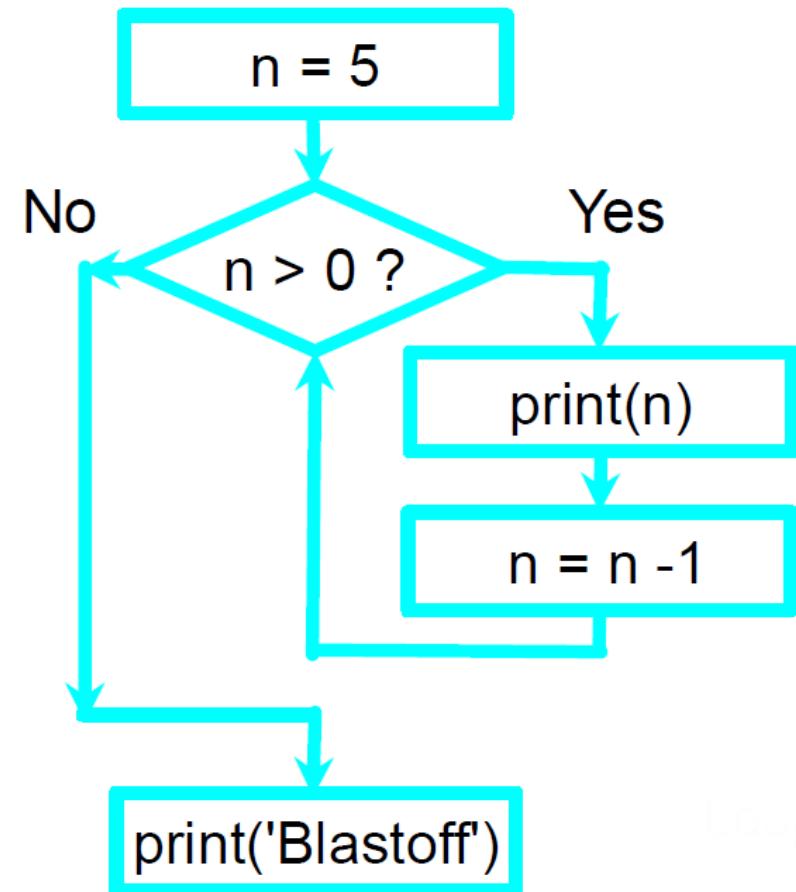
- Repeated steps

Program:

```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Blastoff!')
```

Output:

5  
4  
3  
2  
1  
Blastoff!



- Loops (**repeated steps**) have iteration variables that change each time through a loop.

# Components of Python Code-Program Steps

- Case 1 revisit

➤ Sequential steps

```
import os, numpy, pandas, math  
from pathlib import Path
```

```
name = input('Enter file:')  
file = open(name, 'r')  
counts = dict()
```

```
for line in file:  
    words = line.split()  
    for word in words:  
        counts[word] = counts.get(word, 0) + 1
```

➤ Repeated steps

```
bigcount = None  
bigword = None  
for word, count in list(counts.items()):  
    if bigcount is None or count > bigcount:  
        bigword = word  
        bigcount = count  
  
print(bigword, bigcount)
```

➤ Conditional steps

# Components of Python Code-Program Steps

- A short Python “Story” about how to count words in a file.
- A *word* to read data.
- A *sentence* to update counts.
- A *paragraph* to find the largest item in a list.

```
import os, numpy, pandas, math  
from pathlib import Path
```

```
name = input('Enter file:')  
file = open(name, 'r')  
counts = dict()
```

```
for line in file:  
    words = line.split()  
    for word in words:  
        counts[word] = counts.get(word, 0) + 1
```

```
bigcount = None  
bigword = None  
for word, count in list(counts.items()):  
    if bigcount is None or count > bigcount:  
        bigword = word  
        bigcount = count
```

```
print(bigword, bigcount)
```

# Acknowledgement

- Acknowledgements / Contributions
- These slides are Copyright 2010-Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.
- Initial Development: Charles Severance, University of Michigan School of Information
- Further Development: Zhou Xu, Hong Kong Polytechnic University
- Continuous development: Xiaoyu Wang, Hong Kong Polytechnic University

# Tutorial 1

- Download and install Anaconda Navigator.
  - Windows: <https://docs.anaconda.com/anaconda/install/windows/>
  - macOS: <https://docs.anaconda.com/free/anaconda/install/mac-os/>
- How to open Python shell/editor.
- How to execute editor script.
- How to save file.

# Tutorial 1

- Use previous version!

The screenshot shows the Anaconda website's 'Getting Started' page. A red box highlights the 'Using older versions of Anaconda Distribution' link in the sidebar under the 'Anaconda Distribution' section. The main content area is titled 'Installing older versions of Anaconda Distribution' and includes a note about older versions being subject to Anaconda's Terms of Service. It lists file name endings for different architectures and a warning about the risks of using old versions.

ANACONDA.

Home Getting Started Tools Package Security Manager Data Science & AI Workbench Reference

LINUX AVVS Graviton 2/ARM 04 aarch64.sh

Getting Started

Getting started with Anaconda  
Managing Terms of Service on the command line

Anaconda Distribution

- Overview
- System Requirements
- Installing Anaconda Distribution

Advanced Installation

- Overview
- Installing with silent mode
- Using older versions of Anaconda Distribution**

Installing the anaconda metapackage  
Installing for multiple users  
Installing Anaconda on an air-gapped machine  
Anaconda Distribution release notes  
Uninstalling Anaconda Distribution

Search... Ctrl K Ask AI Pricing Download ☰

On this page

Installing older versions of Anaconda Distribution

ⓘ Older versions of Anaconda Distribution are still subject to Anaconda's [Terms of Service](#).

Older versions of Anaconda Distribution are still available for:

Architecture	File name ending
Windows 32-bit	x86.exe
macOS Intel	x86_64.pkg, x86_64.sh
Linux 32-bit Intel or AMD	x86.sh
Linux IBM Power CPU	ppc64le.sh
Linux IBM Z	s390x.sh

These versions can be downloaded from our [archive](#). However, these installations are no longer supported by Anaconda and most are no longer receiving security updates.

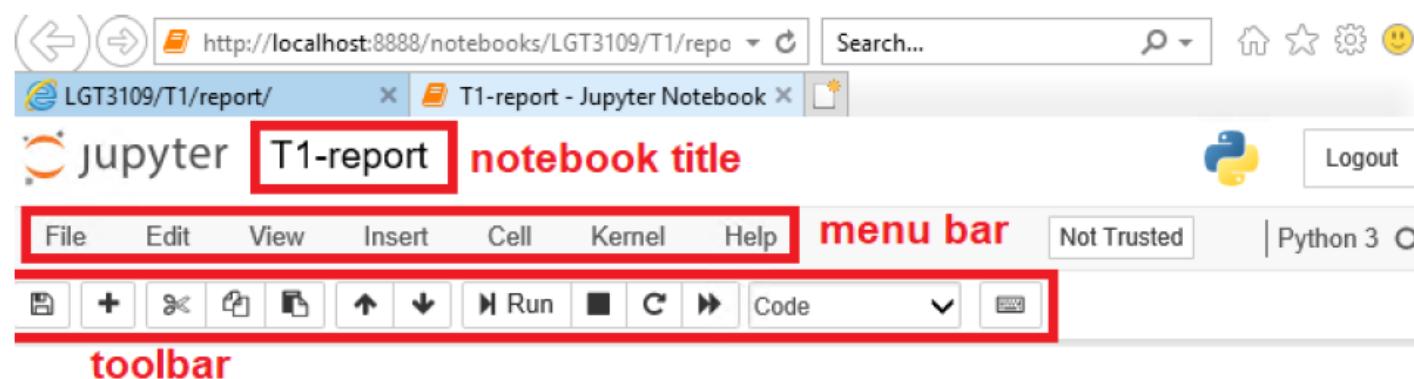
⚠ You might not be able to use conda to update or install packages beyond what is included in the Anaconda Distribution version you've installed. This is because Anaconda Distribution provides a static bundle of packages from the time of release, but the repository continues to be updated over time. Older package versions may be removed or replaced, and the versions present in the repository might not be compatible with your operating system. For package lists for each version of Anaconda Distribution, see the [Anaconda Distribution release notes](#).

# Tutorial 1-Jupyter notebook

- How to launch Jupyter notebook.
- How to open/rename a file.

# Tutorial 1-Jupyter notebook

- Header At the top of the notebook document is a header which contains the notebook title, a menu bar, and toolbar.  
This header remains fixed at the top of the screen, even as the body of the notebook is scrolled.  
The title can be edited in-place (which renames the notebook file)  
The menu bar and toolbar contain a variety of actions which control notebook navigation and document structure.



# Tutorial 1-Jupyter notebook

- Body
  - Markdown Cells: Used to build a nicely formatted texts
  - Code cells: Used to define the computational code
    - Two forms:
      - input cell : the user types the code to be executed
      - output cell : the result of the executed code is shown
    - Raw cells
      - text needs to be included in raw form, without execution or transformation

The screenshot shows a Jupyter Notebook interface with three cells:

- Markdown cell:** Displays the text "I'm a markdown cell."
- Code cell:** Displays the code "In [2]: print("I'm a code cell")" and its output "I'm a code cell".
- Raw cell:** Displays the text "I'm a \*\*raw\*\* cell".

# Tutorial 1-Jupyter notebook

- Edit      **Edit Mode**



```
In [ ]: a = 10
```

- indicated by a green cell border and a prompt showing in the editor area.
- When a cell is in edit mode, you can type into the cell, like a normal text editor.

- Command Mode



```
In [ ]: a = 10
```

- is indicated by a grey cell border.
- In command mode: the structure of the notebook can be modified as a whole, but the text in individual cells cannot be changed.

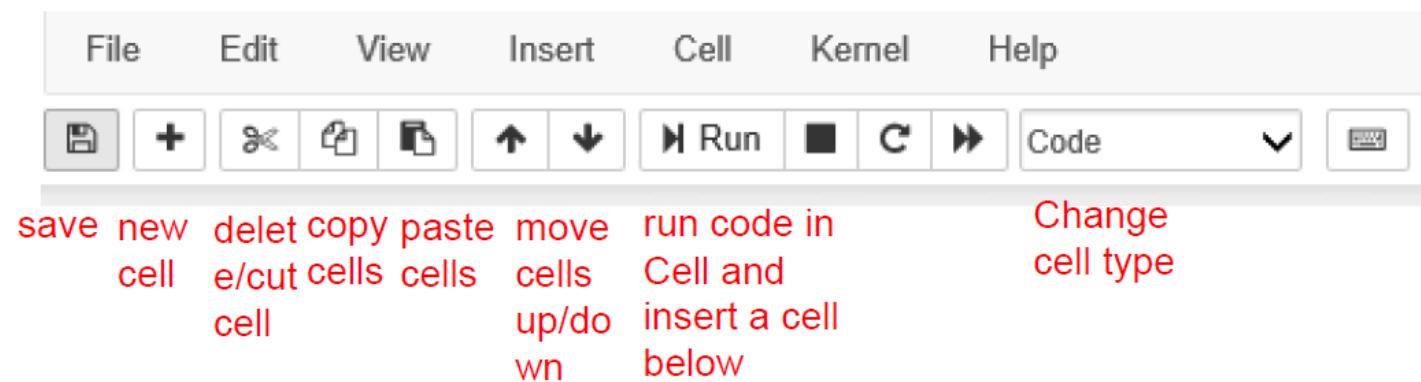
# Tutorial 1-Jupyter notebook

- Mouse navigator

## Mouse Navigation

- cells can be selected by clicking on them
- cell actions usually apply to the currently selected cell
  - To run the code in a cell, select it and then click “Run” in the toolbar (or Run Cells in Menu Cell)

The screenshot shows the Jupyter Notebook interface. At the top is a menu bar with 'Cell', 'Kernel', and 'Help'. Below the menu is a vertical toolbar with icons for file operations like save, new, delete, copy, paste, move, and cell type changes. A horizontal toolbar below the menu includes icons for file, edit, view, insert, cell, kernel, help, and various cell selection and modification tools. On the right, a code cell contains the Python command `In [2]: print("I'm a code cell")`, which has been executed and displayed as the output `I'm a code cell`.



# Tutorial 1-Jupyter notebook

- Keyboard navigator      **Keyboard Navigation**
  - The most important keyboard shortcuts are
    - Enter - enters edit mode
    - Esc - enters command mode
  - Other useful shortcuts in command mode (these can be viewed in the notebook at time via the Help→Keyboard Shortcuts menu item):
    - Shift + Enter: Run Cell
    - Y: to code
    - M: to markdown
    - R: to raw
    - B: insert a cell below
    - D,D: delete the selected cell
    - Ctrl+S: save notebook

The screenshot shows the Jupyter Notebook's Help menu with the 'Keyboard Shortcuts' option highlighted. Below the menu, a detailed list of keyboard commands for Command Mode is provided.

**Keyboard shortcuts**

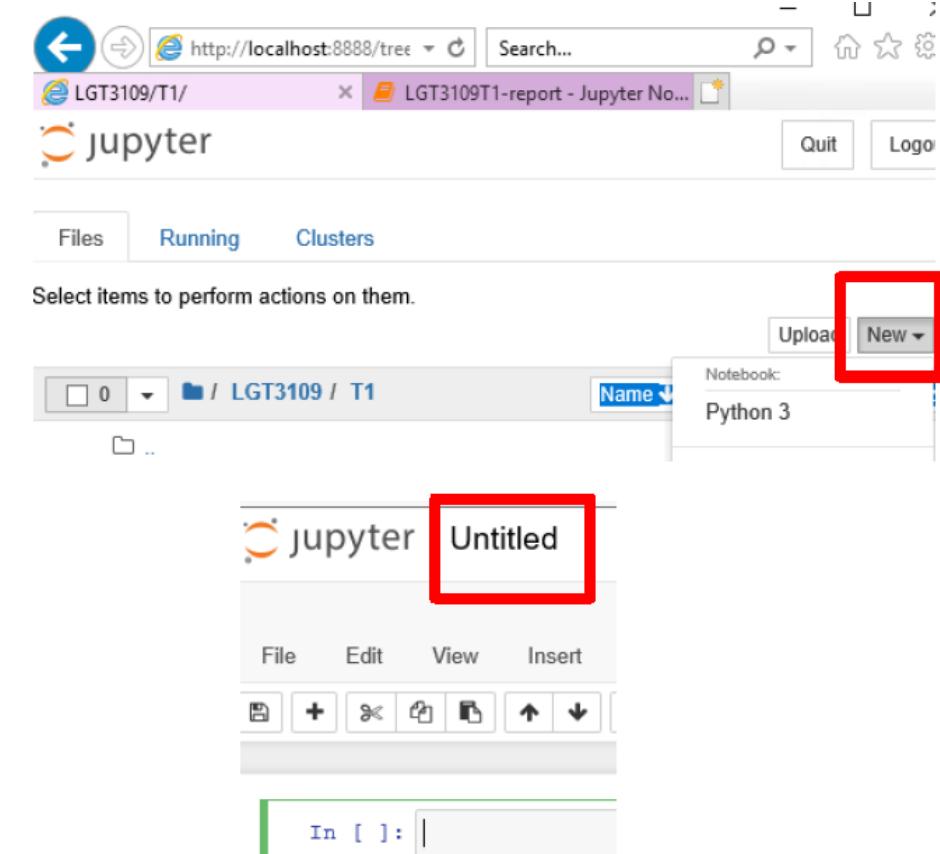
The Jupyter Notebook has two different keyboard input modes. Edit mode is indicated by a green cell border. Command mode is indicated by a grey cell border with a blue left margin.

**Command Mode (press `Esc` to enable)**

- `F`: find and replace
- `Ctrl+Shift+F`: open the command palette
- `Ctrl+Shift+P`: open the command palette
- `Enter`: enter edit mode
- `P`: open the command palette
- `Shift-Enter`: run cell, select below

# Tutorial 1-Jupyter notebook

- Create a notebook
  - Go to the working directory – /LGT3109/T1
  - Click “New” in the toolbar and choose “Python 3”
  - Now you can edit the new notebook
    - E.g., to rename the notebook file name, you can change the title (which is “Untitled” by default)



# Tutorial 1-Jupyter notebook

- Insert picture

Insert the saved picture in Jupyter Notebook (Use of Menu Bar)

- Go to Jupyter notebook of the tutorial report
- Select the cell below in the jupyter notebook

Follow Steps 1-5 to insert a screenshot of your account information in Windows, in the cell below (20 points):

\*Insert the screenshot of your account information here:\*

- Click the selected cell, click “Insert Image” in Edit Menu of the menu bar, click “Browser” to locate the picture to be inserted, click OK
- The image is inserted and can also be resized

! [account\_info.PNG] (attachment:account\_info.PNG)

# Tutorial 1-Jupyter notebook

- Insert picture
- Insert the saved picture in Jupyter Notebook (Use of < img >)
  - Use <img> tag to insert a figure in a markdown cell
  - <img src = file\_name width= width of the figure>

```

```