

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

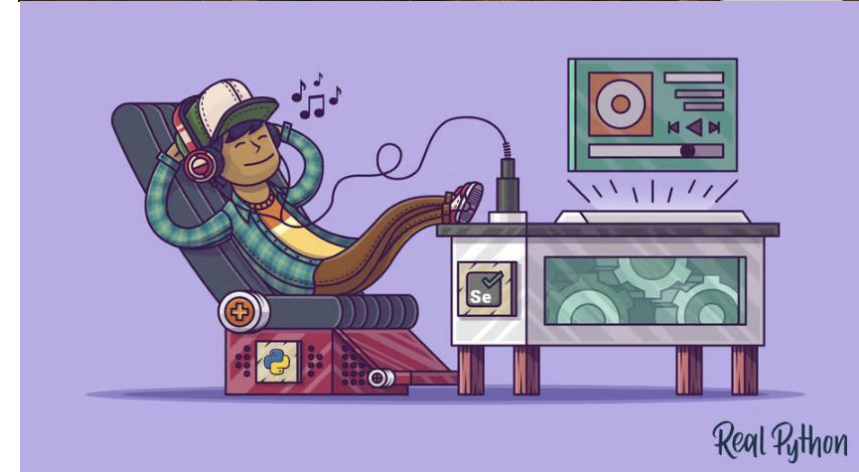
Our Goal

- 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

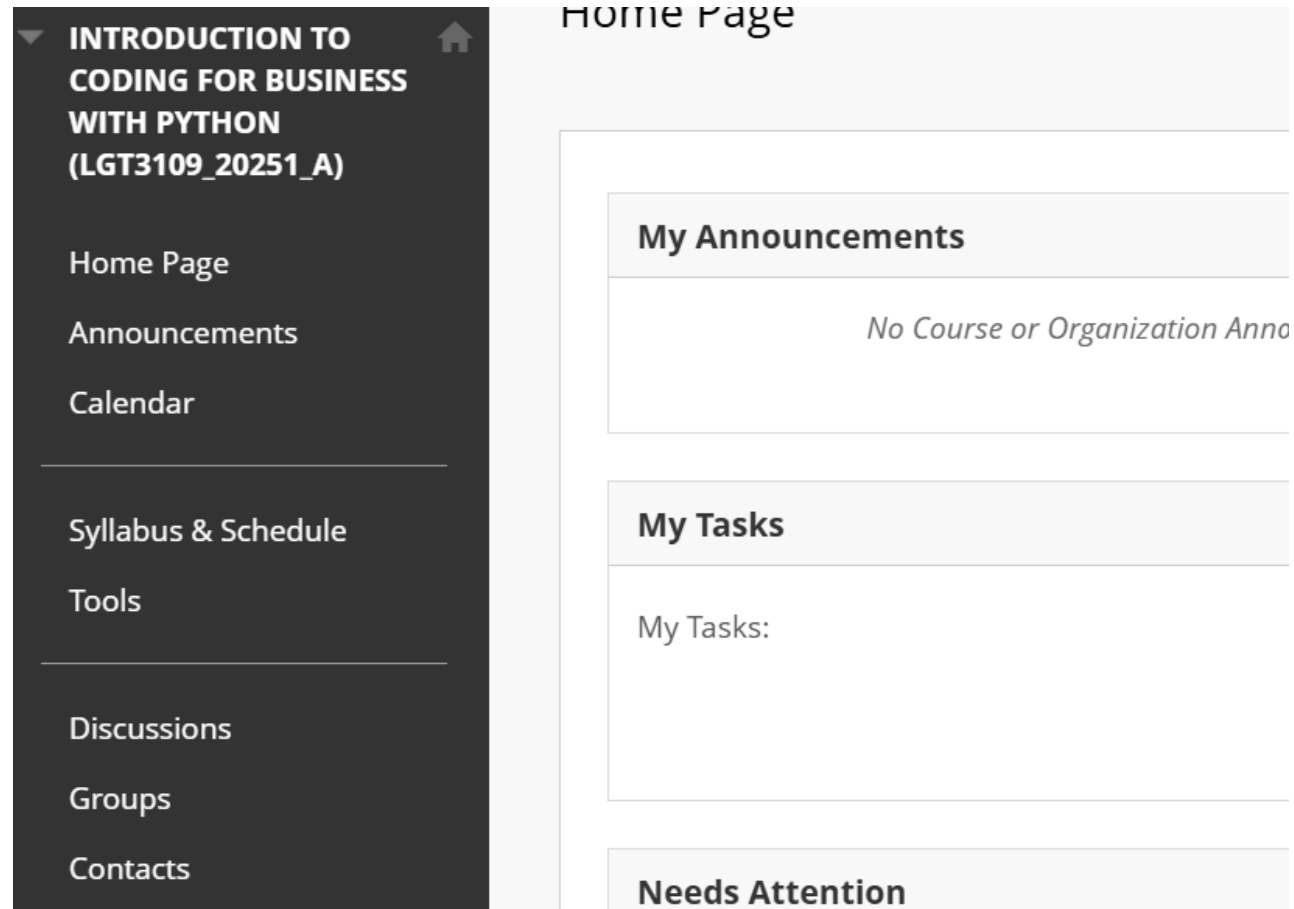
- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

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
- Office: M1007; Tel.: 2766-7369

Basic Information-Blackboard

- Lectures, tutorials, and class videos.



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

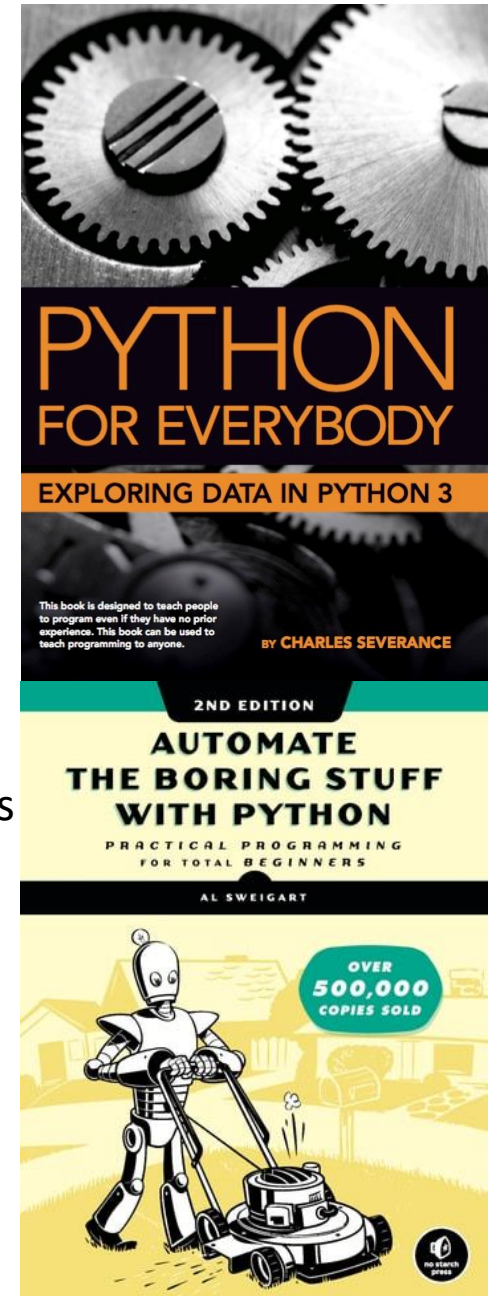
Tutorial on every Wednesday (10:30am-11:20am), at MN102c

- 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 | Getting Started: What's the use of Python? How to type and execute Python programs? |
| | Variables and Simple Data Types |
| | Basic Flow Control |
| | Functions |
| | Strings, Lists, and Dictionaries |
| | Testing and Debugging Python Programs |
| | Reading, Writing , and Organizing Working Files |
| 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

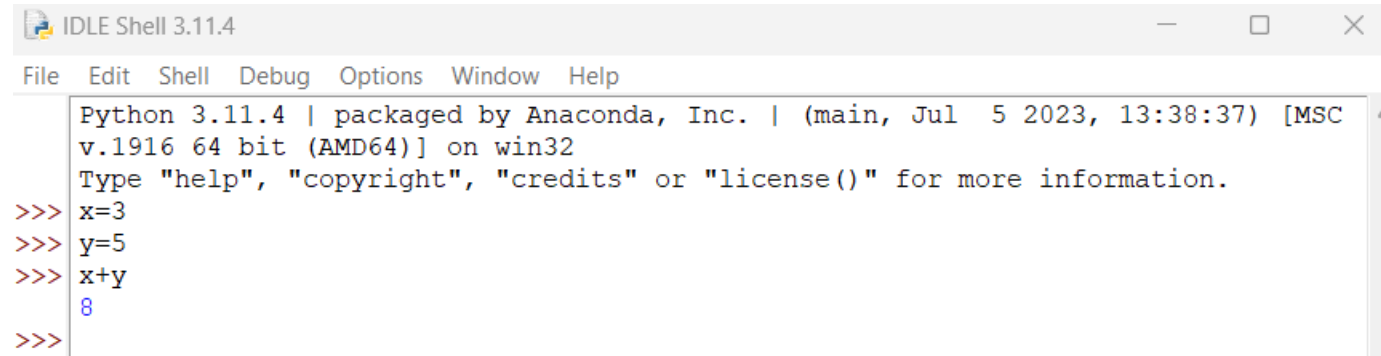
- The goal of the course
- Course basic information
- Arrangement for teaching
- Coding environment
- Lecture and tutorial
- Grading

Software We Use-Shell/Editor

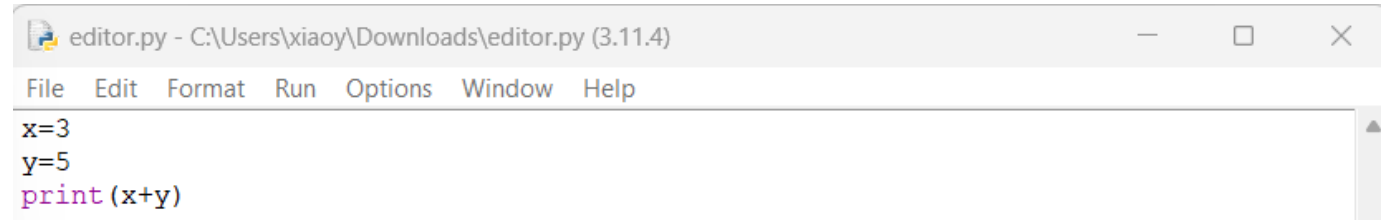
- Some definition clarifications.
- Are you confused with words like `Python`, `IDLE`, `Shell`, `Jupyter`...
- `Python` is a programming `language` (communicate with machine).
- A `shell` is a command-line `interface` that allows you to interact with the operating system.
- A `text editor` is a program 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.



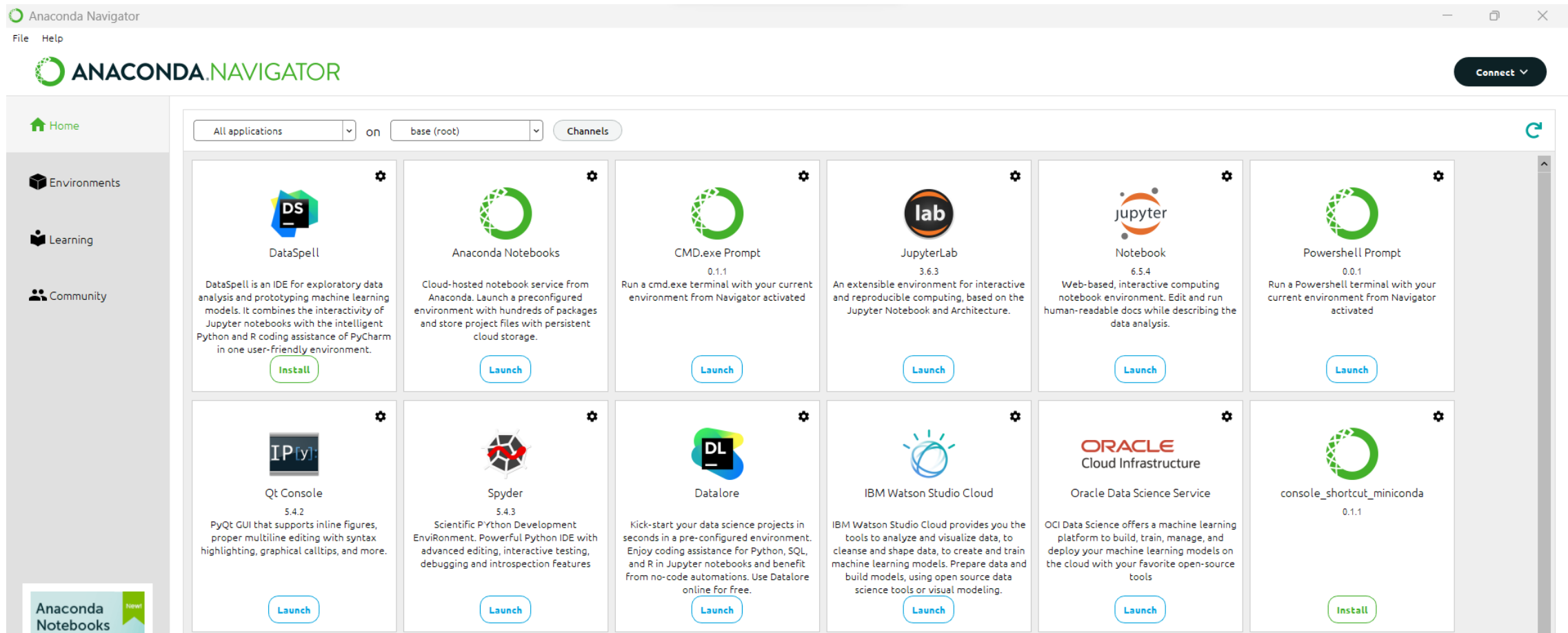
The screenshot shows the IDLE Shell 3.11.4 window. The title bar reads "IDLE Shell 3.11.4". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the Python 3.11.4 startup message: "Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32". Below this, it says "Type 'help', 'copyright', 'credits' or 'license()' for more information." The prompt ">>>" is followed by the code: `x=3`, `y=5`, `x+y`, and the output `8`.



The screenshot shows the IDLE editor window. The title bar reads "editor.py - C:\Users\xiaoy\Downloads\editor.py (3.11.4)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code: `x=3`, `y=5`, and `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

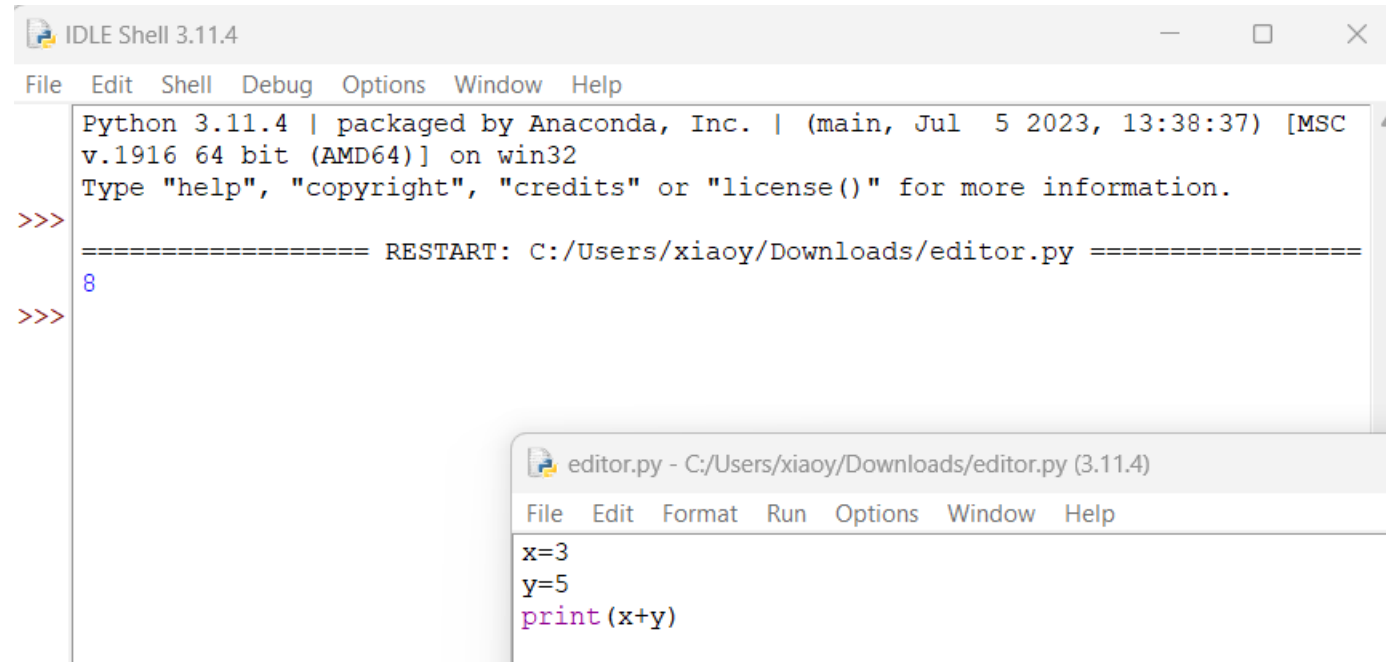
Lectures and Tutorials

- 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)



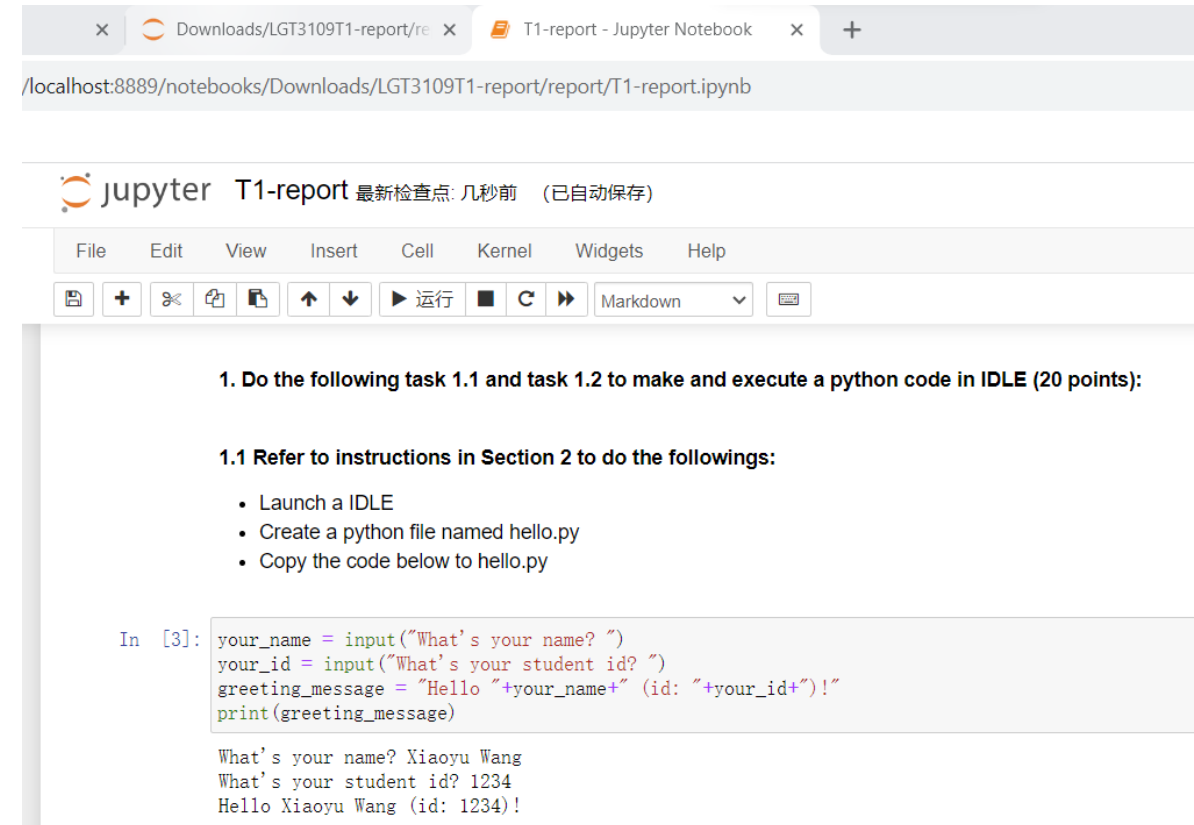
The screenshot displays the IDLE Python 3.11.4 environment. The main window, titled 'IDLE Shell 3.11.4', shows the Python interpreter's startup message: 'Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul 5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)] on win32'. It prompts the user to type 'help', 'copyright', 'credits', or 'license()' for more information. Below this, a prompt '8' is visible, and the output '8' is shown. A second window, titled 'editor.py - C:/Users/xiaoy/Downloads/editor.py (3.11.4)', is open in the foreground, displaying a simple Python script:

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

Lectures and Tutorials-Coding Environment

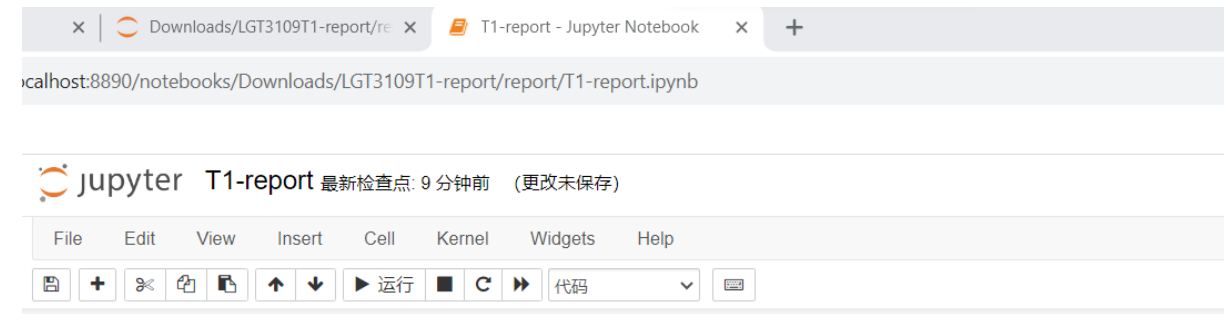
- For Tutorials:

➤ Jupyter Notebook (Interactive Development Environment)



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!**



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

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

Hello Xiaoyu Wang!

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

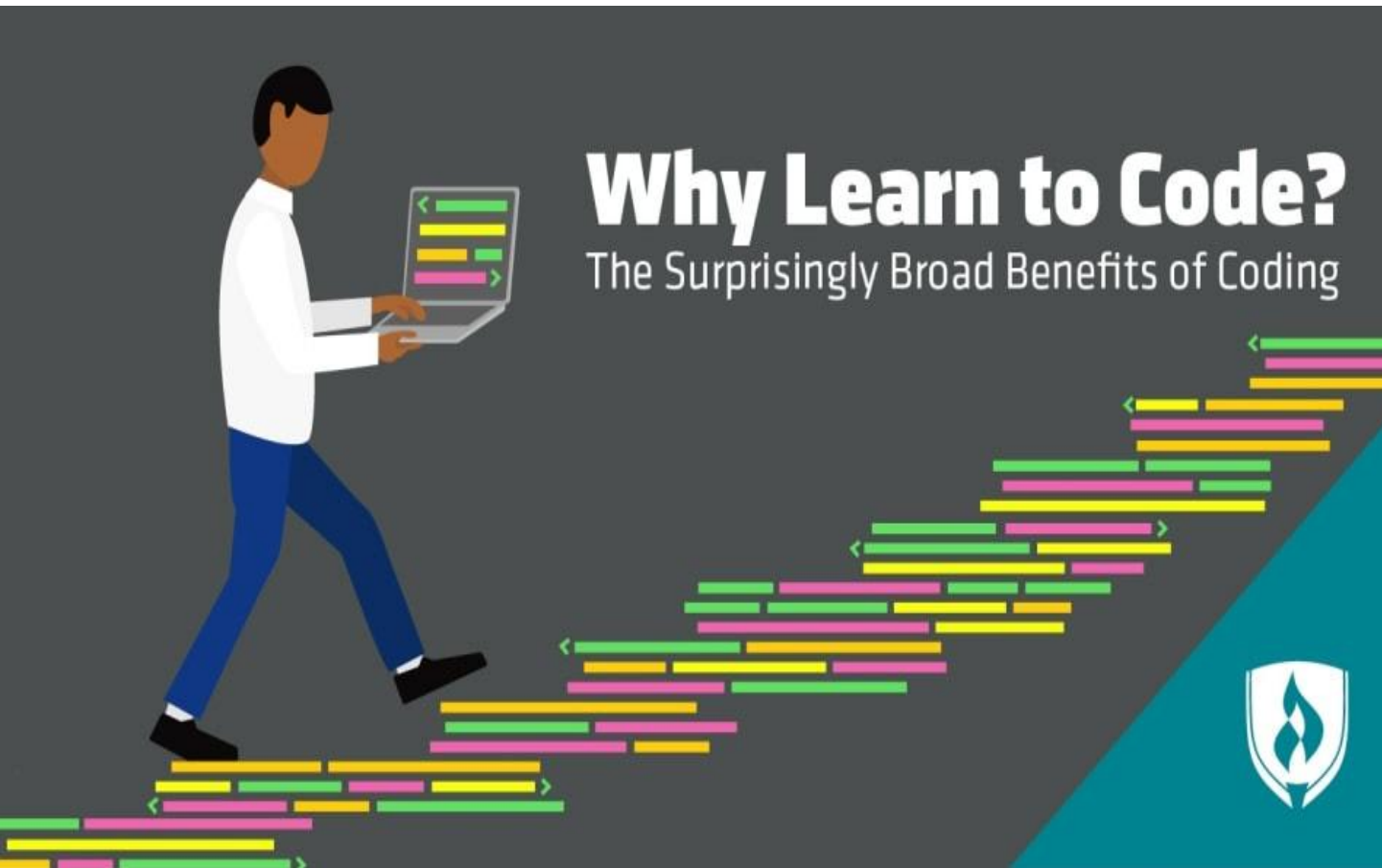
- Why coding
- What's code
- What's Python
- Components of Python Code

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?

A screenshot of a Python IDLE Shell window. The title bar says 'IDLE Shell 3.11.4'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following output:

```
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
>>>
```

Below the shell window, a separate window titled 'freq_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\teaching\L...' is open. It shows the source code of the script:

```
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()

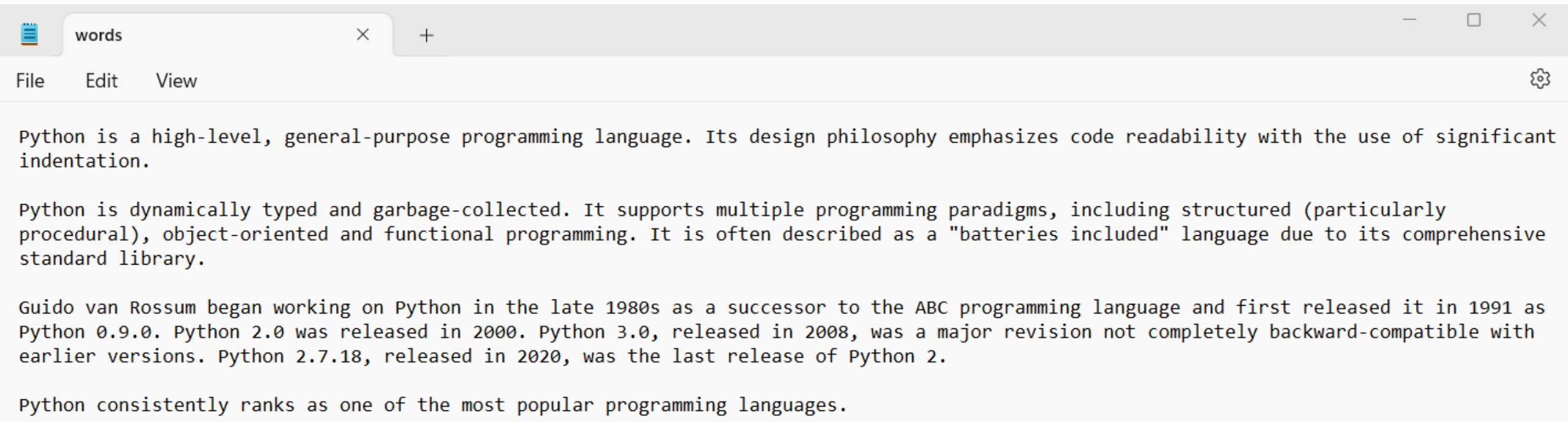
for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

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.



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.

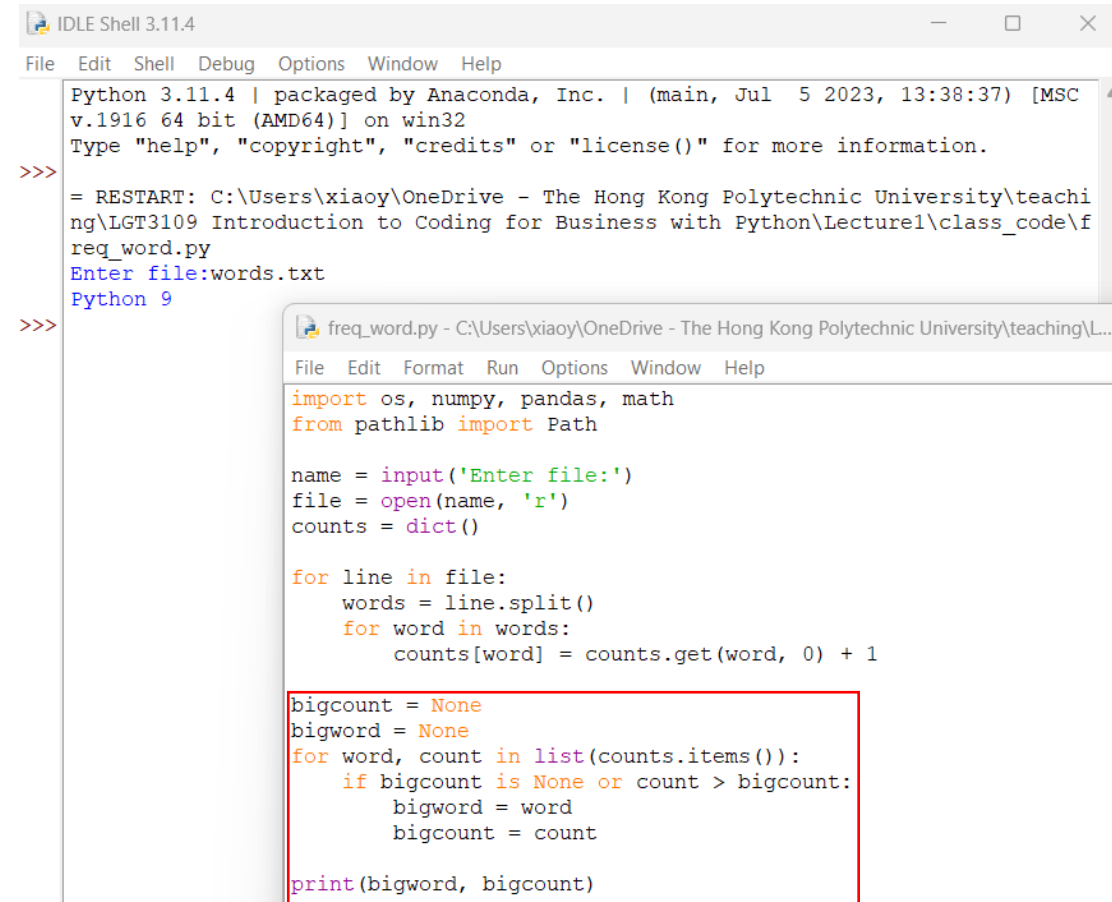
Why Coding-Case 1-Find the Largest

- 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 image shows two windows from the IDLE 3.11.4 environment. The top window is the IDLE Shell, displaying the Python 3.11.4 startup message and the command prompt. The bottom window is a Python script editor titled 'freq_word.py', showing the code for finding the most frequent word in a file. The code includes imports for 'os', 'numpy', 'pandas', 'math', and 'Path' from 'pathlib'. It prompts the user for a file name, opens the file, and iterates through its lines to count word frequencies. The final part of the code, which finds the word with the highest frequency, is highlighted with a red box.

```
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 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?

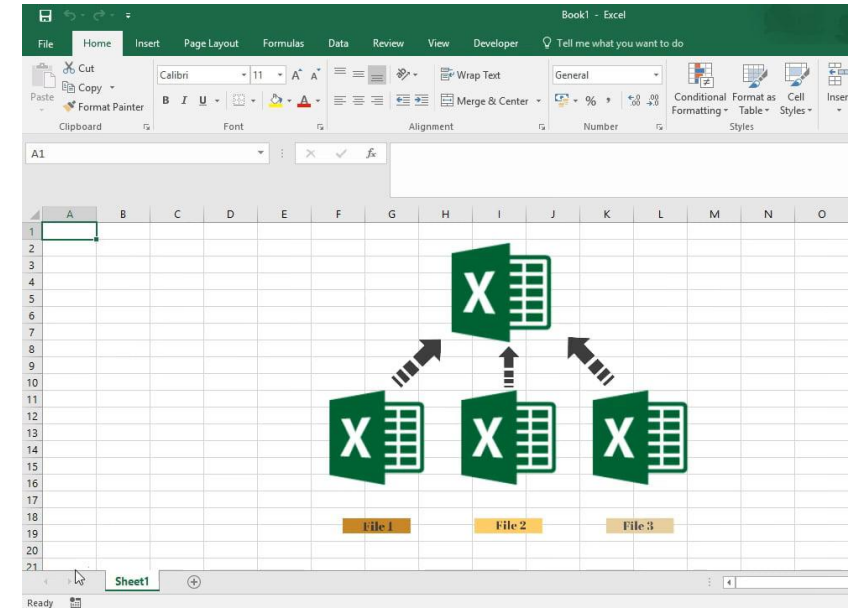
| | A | B | C | D | E | F | G | H |
|---|----|-----|----|-----|-----|-----|-----|-----|
| 1 | BU | OP | DP | JAN | FEB | MAR | APR | MAY |
| 2 | SH | HK | LA | 20 | 10 | 20 | 20 | 15 |
| 3 | SH | SH | LA | 20 | 5 | 20 | 20 | 15 |
| 4 | SH | SIN | SF | 5 | 10 | 20 | 5 | 15 |

| | 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?



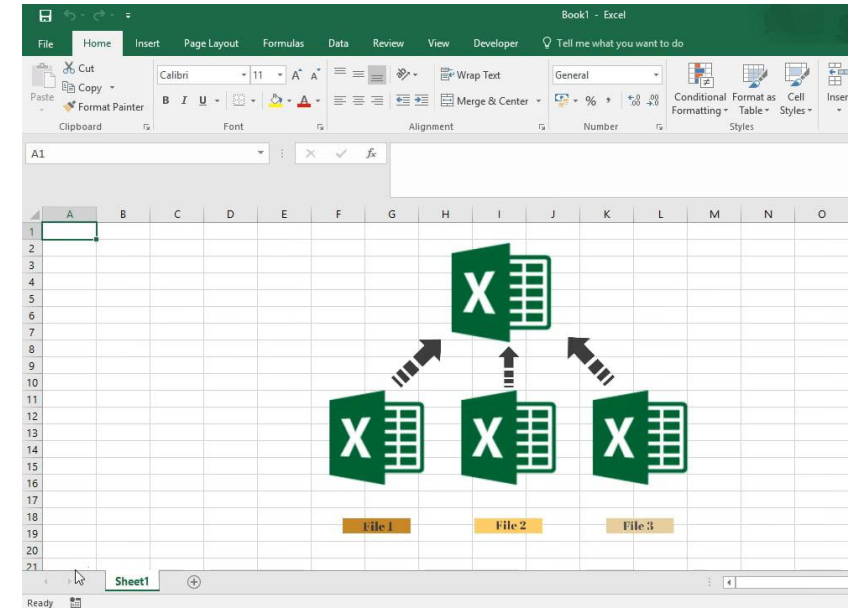
Why Coding-Case 2-How Do you Achieve

- 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)
```

```
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)
```

➤ Handle data in order.

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

➤ Process data and combine.

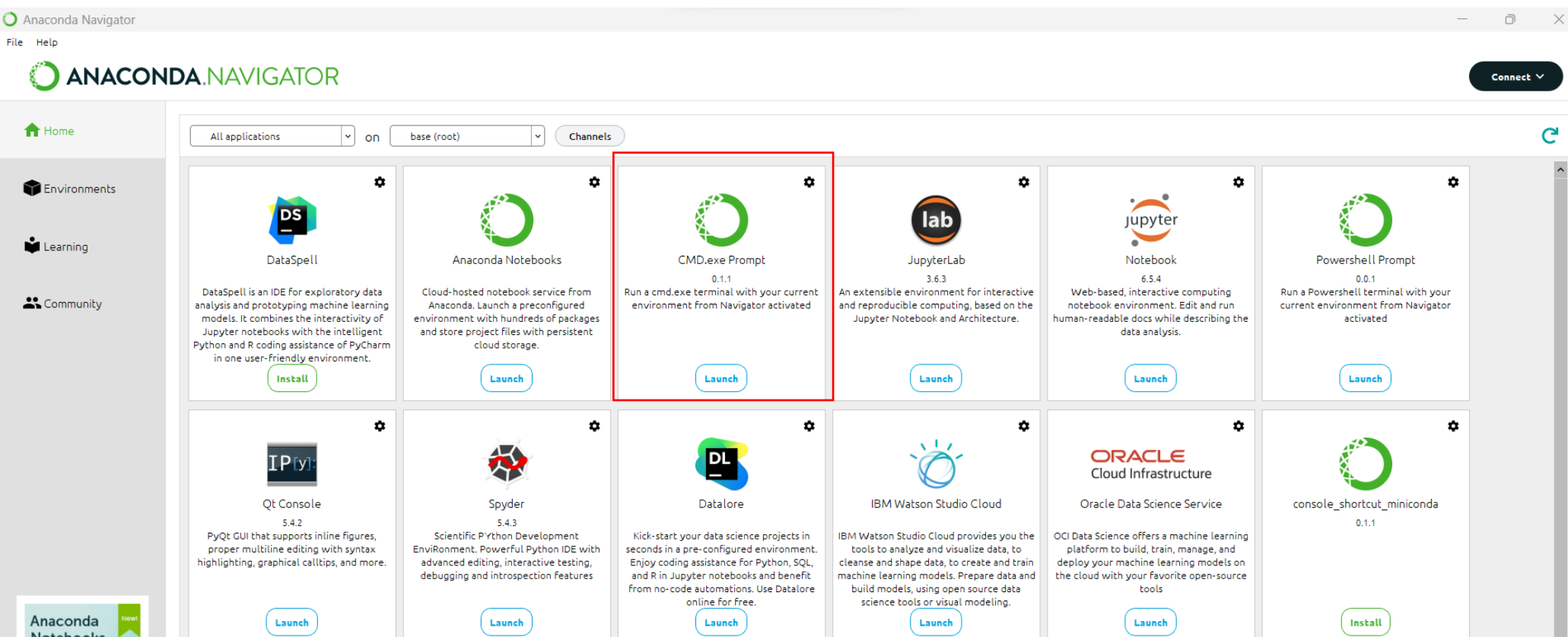
```
if file.startswith('Demand') and file.endswith('.xlsx'):
    df = pd.concat([df, pd.read_excel(file)], ignore_index=True)
```

➤ Export file.

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

Why Coding-Case 2-How to Run

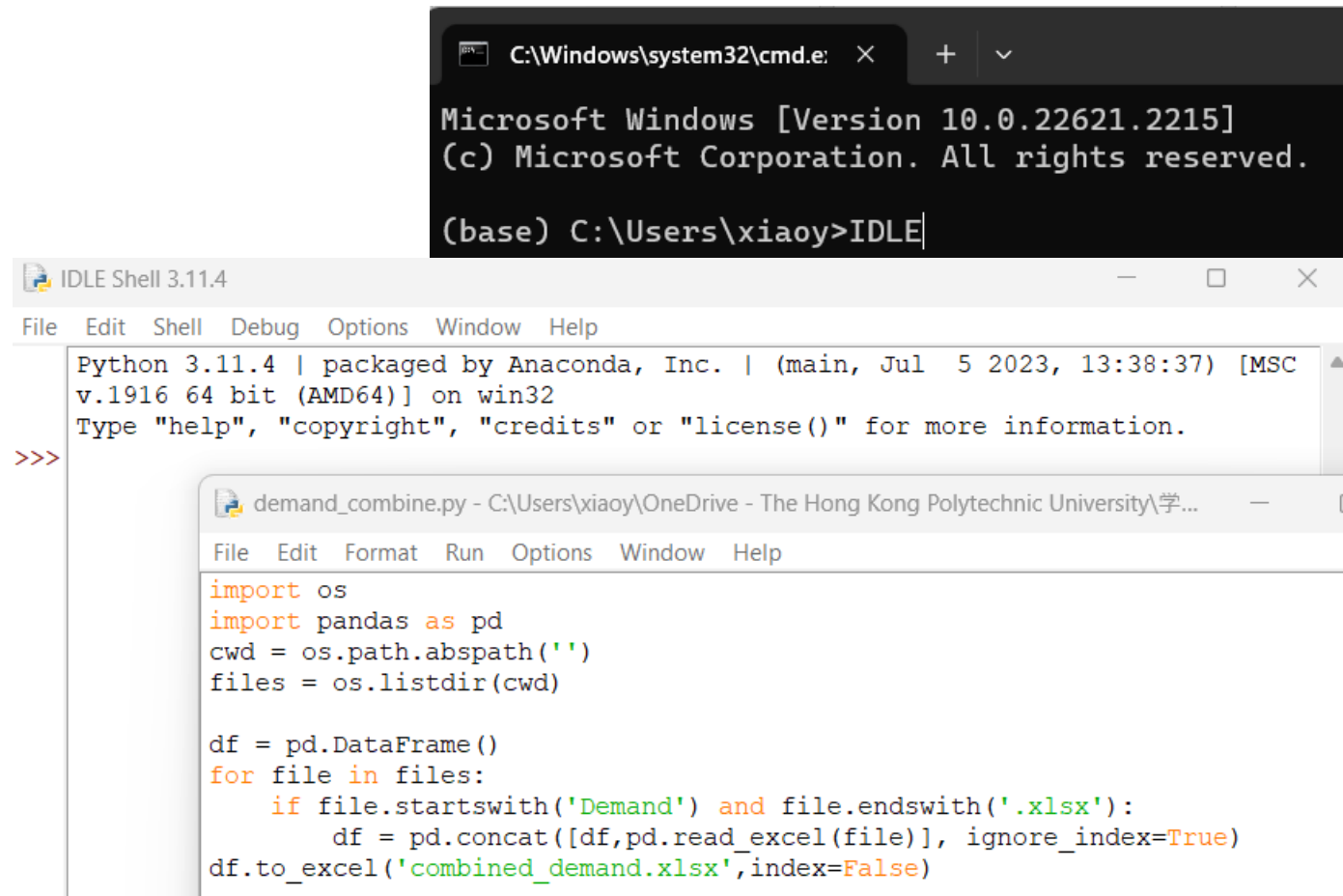
- Open Anaconda Navigator.
- Open the terminal CMD.exe Prompt.



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.



The image shows a Windows terminal window and an IDLE Shell window. The terminal window is titled "C:\Windows\system32\cmd.e" and displays the command prompt output: "Microsoft Windows [Version 10.0.22621.2215] (c) Microsoft Corporation. All rights reserved. (base) C:\Users\xiaoy>IDLE". The IDLE Shell window is titled "IDLE Shell 3.11.4" and shows the Python 3.11.4 prompt: ">>>". Below the IDLE Shell window, a code editor window titled "demand_combine.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechnic University\学..." displays the following Python code:

```
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;  
}
```



The illustration shows a person with dark hair and glasses, wearing a blue shirt, sitting at a dark desk. In front of them is a laptop with a light pink screen displaying the code symbols '</>'. The background is a light blue gradient, and the person is surrounded by floating CSS code snippets, including rules for 'table', 'button', 'a:link', 'img', 'iframe', 'blockquote', and 'pre'.

Part 2: Motivation for Programming

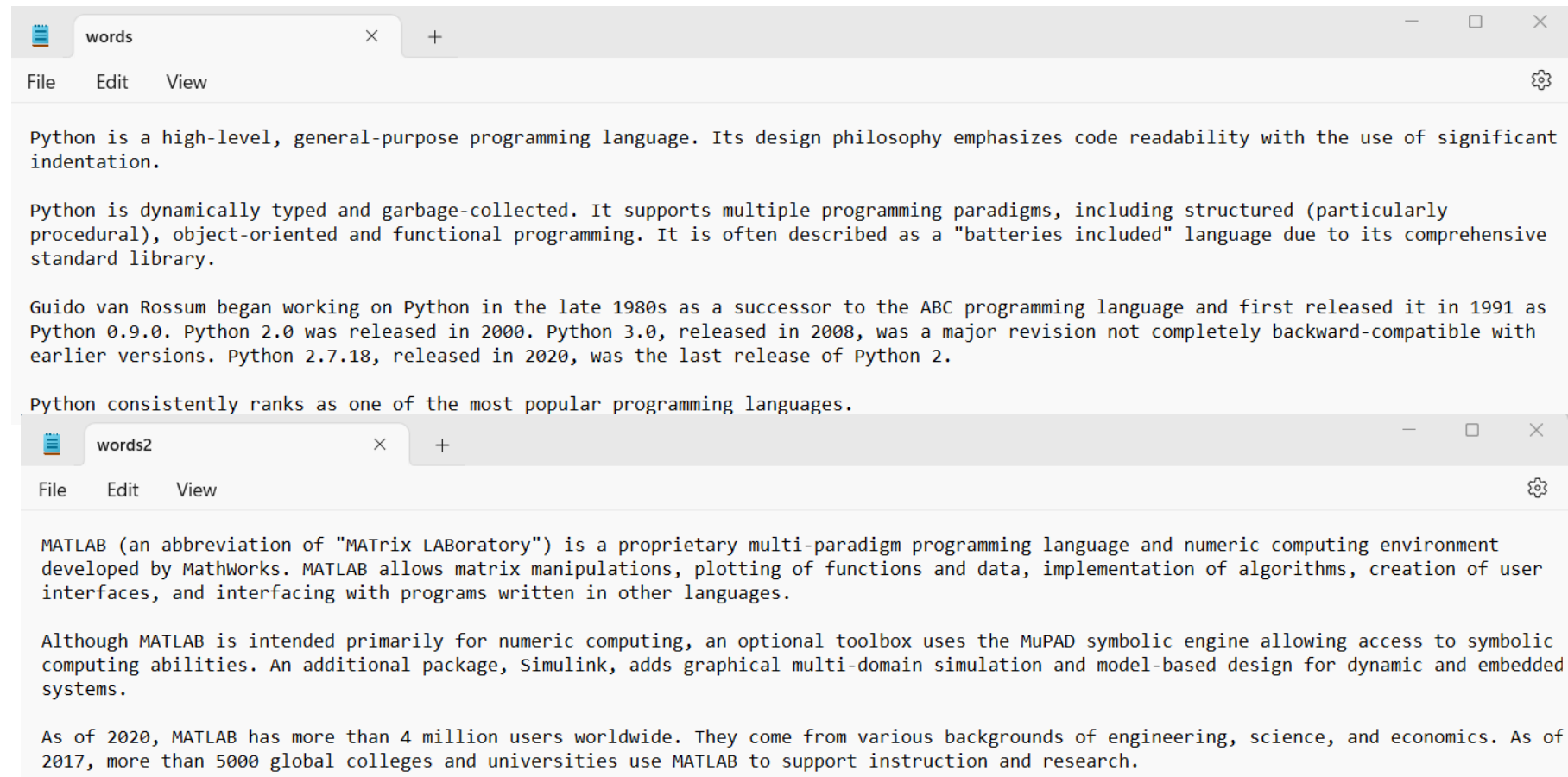
- Why coding
- What's code
- What's Python
- Components of Python Code

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

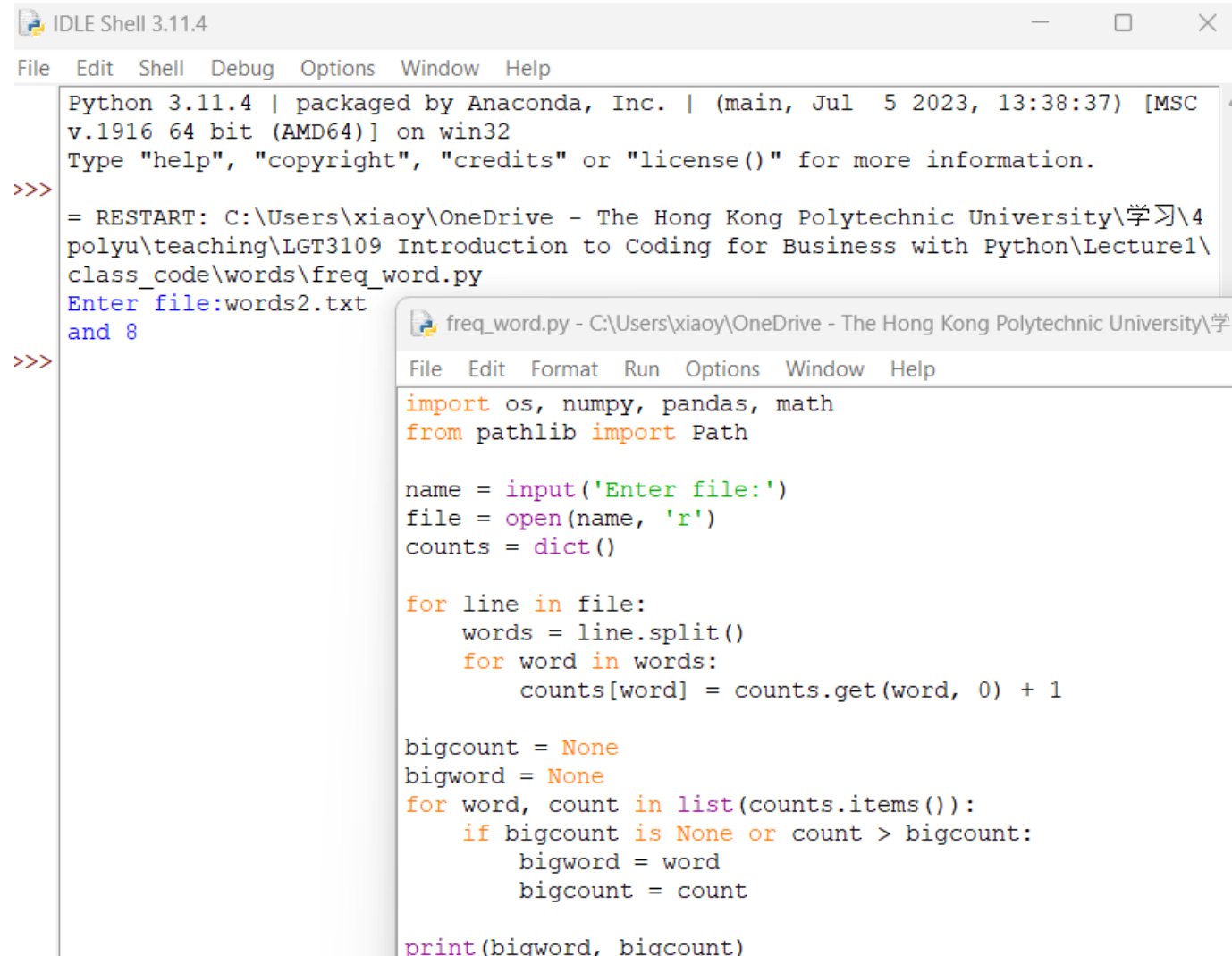
What's Code-Case 1 Revisit

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



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 screenshot shows two windows from the Python IDLE 3.11.4 environment. The top window is the 'IDLE Shell 3.11.4' which displays the Python startup message and the execution of a script. The bottom window is the 'freq_word.py' editor showing the source code of the script.

```
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
>>>
```

```
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()

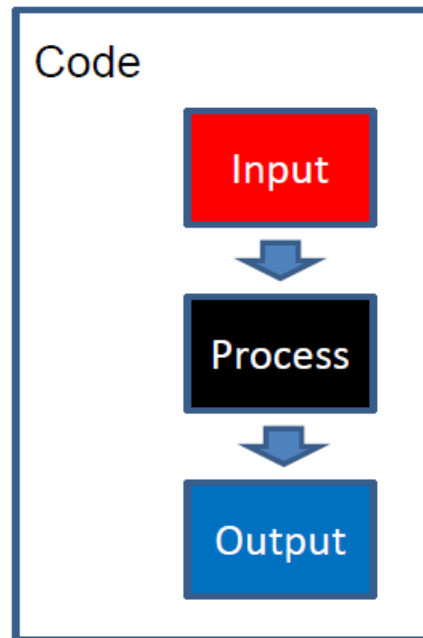
for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

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)
```

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.

Python

```
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.
    }
}
```

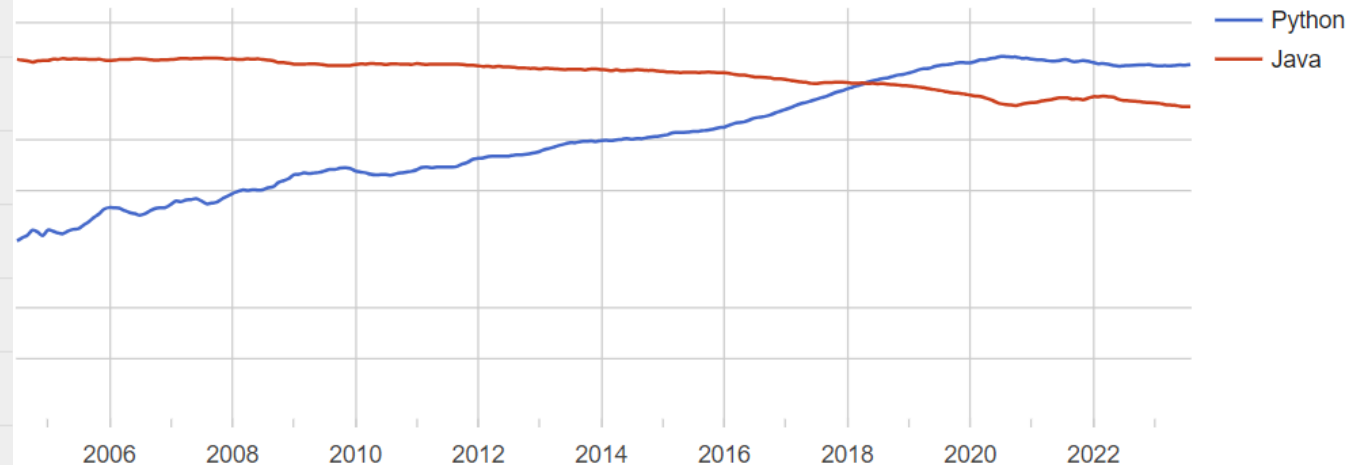
Java

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 ...

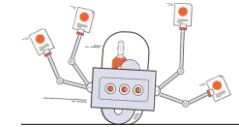
- Task Automation:

- 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
>>>
```

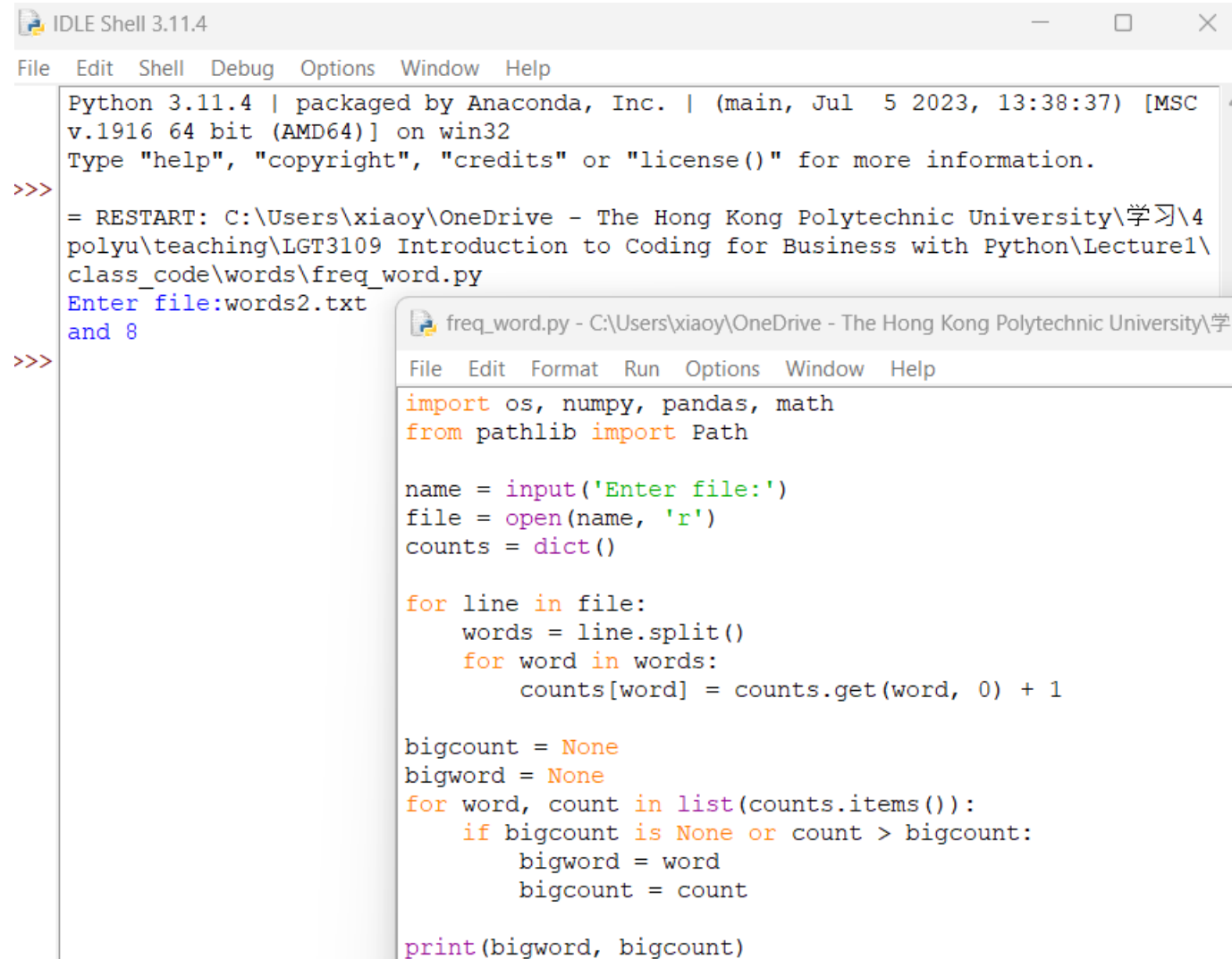
```
>>> x=3
>>> y=5
>>> x<y
True
>>> x>7
False
>>>
```

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 Python IDLE 3.11.4 environment. The top window is the 'IDLE Shell 3.11.4' which displays the Python startup message and the execution of a script. The bottom window is the 'freq_word.py' editor showing the source code of the script.

```
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
>>>
```

```
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)
```

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.

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

➤ Counts start from null

➤ After 1st iteration, counts = [{"Python",1}]

➤ After 2nd iteration, counts=[{"Python",1},{"is",1}]

```
freq_word.py - C:\Users\xiaoy\OneDrive - The Hong Kong Polytechn
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

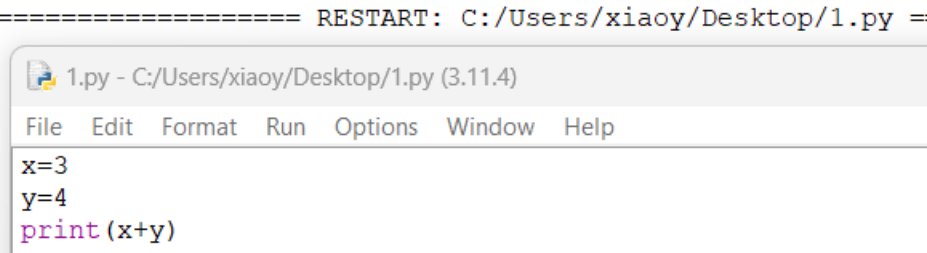
Components of Python Code-Write Script

- Shell or text editor?
- Interactive Python (**Shell**)
- You type directly to Python **one line at a time**, and it responds.

```
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
```

- 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.

```
>>>  
7  
>>>
```



===== RESTART: C:/Users/xiaoy/Desktop/1.py =====

1.py - C:/Users/xiaoy/Desktop/1.py (3.11.4)

File Edit Format Run Options Window Help

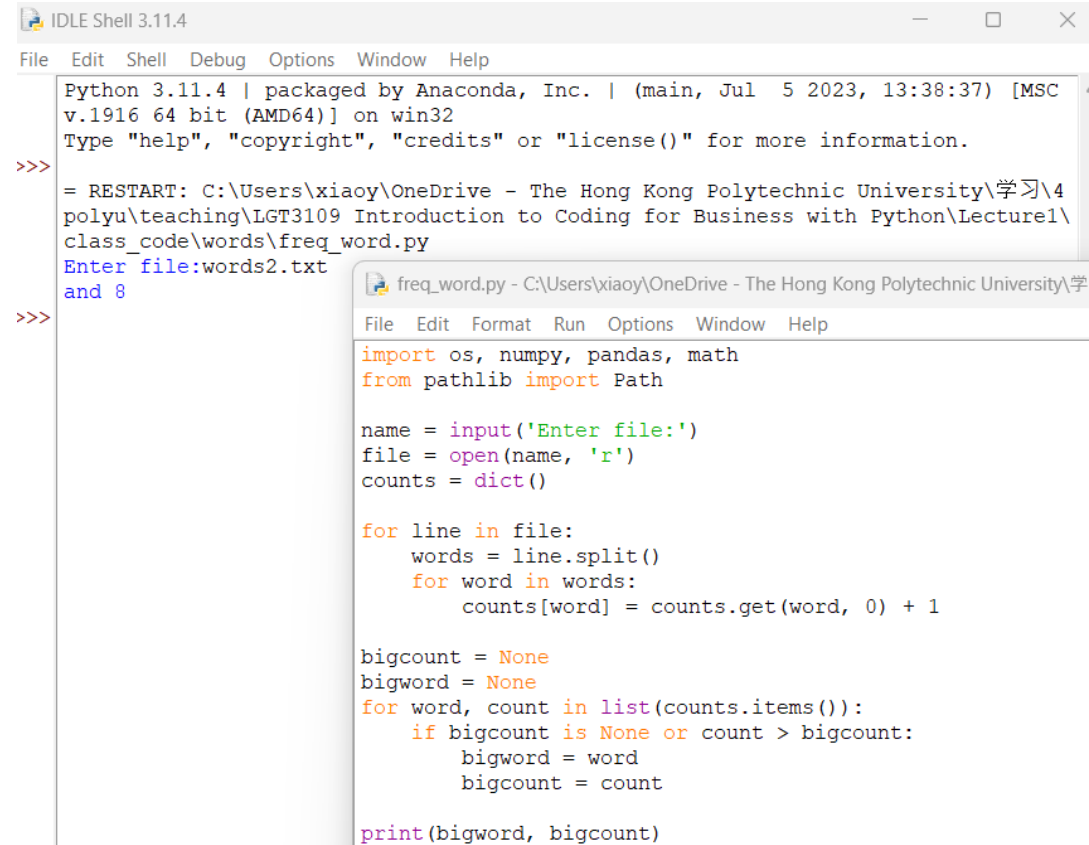
```
x=3  
y=4  
print(x+y)
```

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**.



```
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
>>>
```

```
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()

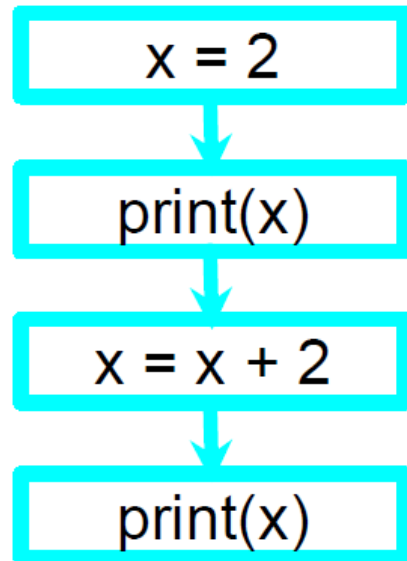
for line in file:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

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)
```

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:

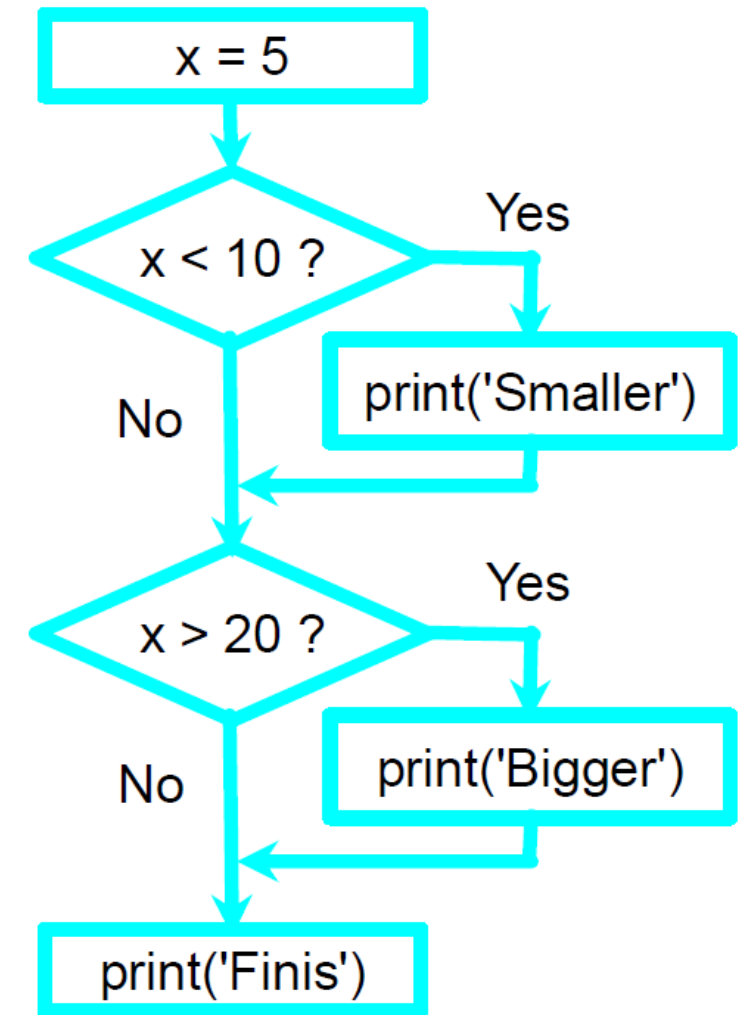
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

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

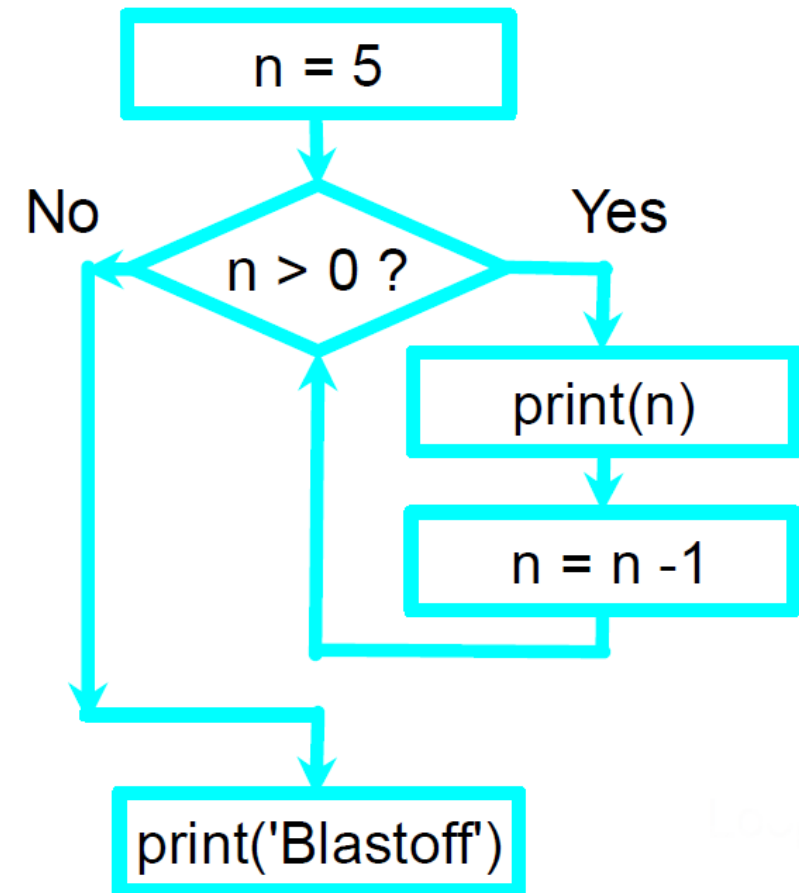
Program:

```
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

- Repeated steps

- Conditional 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
```

```
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-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) 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' section. The left sidebar menu is highlighted with a red box, showing the following items: 'Getting Started', 'Getting started with Anaconda', 'Managing Terms of Service on the command line', 'Anaconda Distribution' (expanded), 'Overview', 'System Requirements', 'Installing Anaconda Distribution', 'Advanced Installation' (expanded), 'Overview', 'Installing with silent mode', 'Using older versions of Anaconda Distribution' (highlighted with a red box), 'Installing the anaconda metapackage', 'Installing for multiple users', 'Installing Anaconda on an air-gapped machine', 'Anaconda Distribution release notes', and 'Uninstalling Anaconda Distribution'. The main content area is titled 'Installing older versions of Anaconda Distribution'. It includes a warning box stating that older versions are still subject to Anaconda's Terms of Service. Below this, it states that older versions are still available for download. A table lists the available architectures and their corresponding file names. The word 'archive' in the text 'These versions can be downloaded from our archive' is highlighted with a red box. A final warning box at the bottom states that older versions are no longer supported and may not be compatible with the current operating system.

ANACONDA

Q Search... Ctrl K Ask AI

Pricing Download > ⚙

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

LINUX AWS GRAVITON Z/ARM 64 aarch64.SFI

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

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](#).

On this page

[Installing older versions of Anaconda Distribution](#)

Tutorial 1-Jupyter notebook

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

Tutorial 1-Jupyter notebook

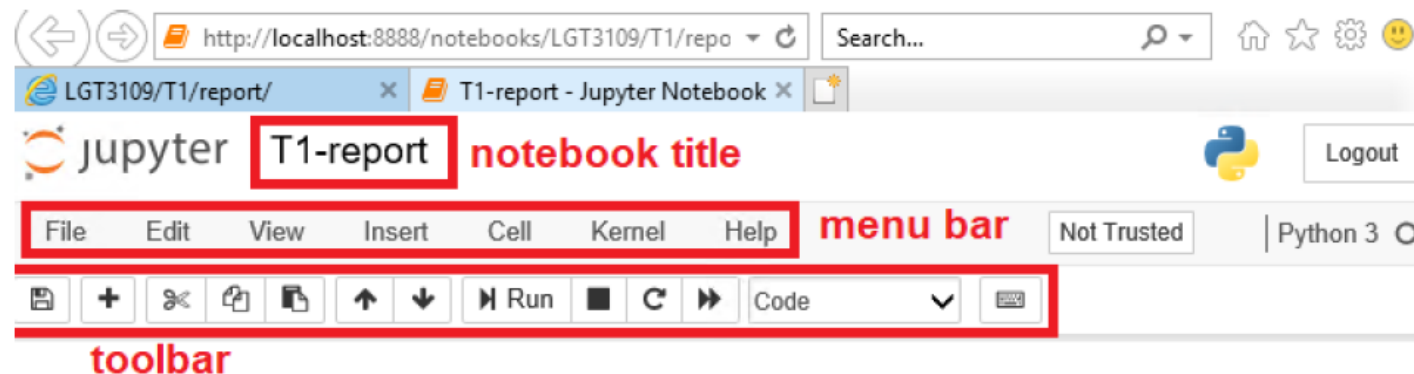
- Header

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



Tutorial 1-Jupyter notebook

- Edit

Edit Mode

A screenshot of a Jupyter notebook cell in Edit Mode. The cell has a green border and contains the text 'In []: a = 10|' where the cursor is at the end of the line.

```
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

A screenshot of a Jupyter notebook cell in Command Mode. The cell has a grey border and contains the text 'In []: a = 10' without a cursor.

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

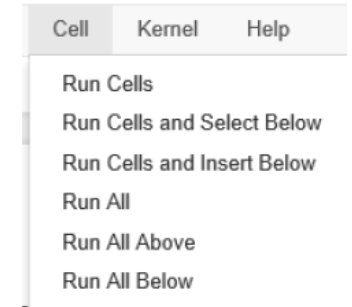
- is indicated by a grey cell border.
- In command mode: the structure of the notebook can be modified as a whole, but the text in individual cells cannot be changed.

Tutorial 1-Jupyter notebook

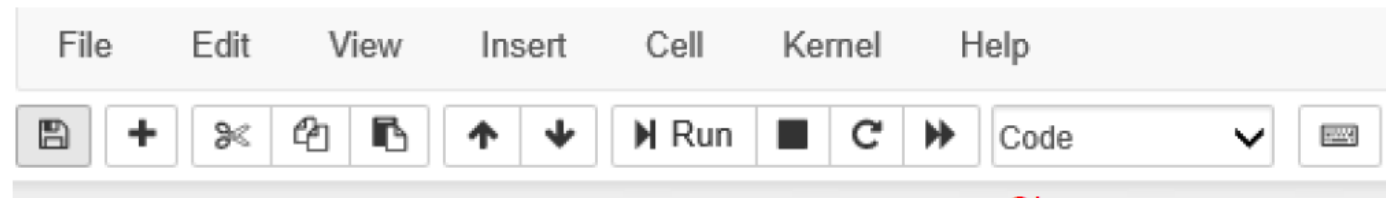
- Mouse navigator

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)



```
In [2]: print("I'm a code cell")  
I'm a code cell
```



save new delet copy paste move run code in Change
cell cell e/cut cells cells cells Cell and cell type
cell cell up/down insert a cell below

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

Help

User Interface Tour

Keyboard Shortcuts

Edit Keyboard Shortcuts

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

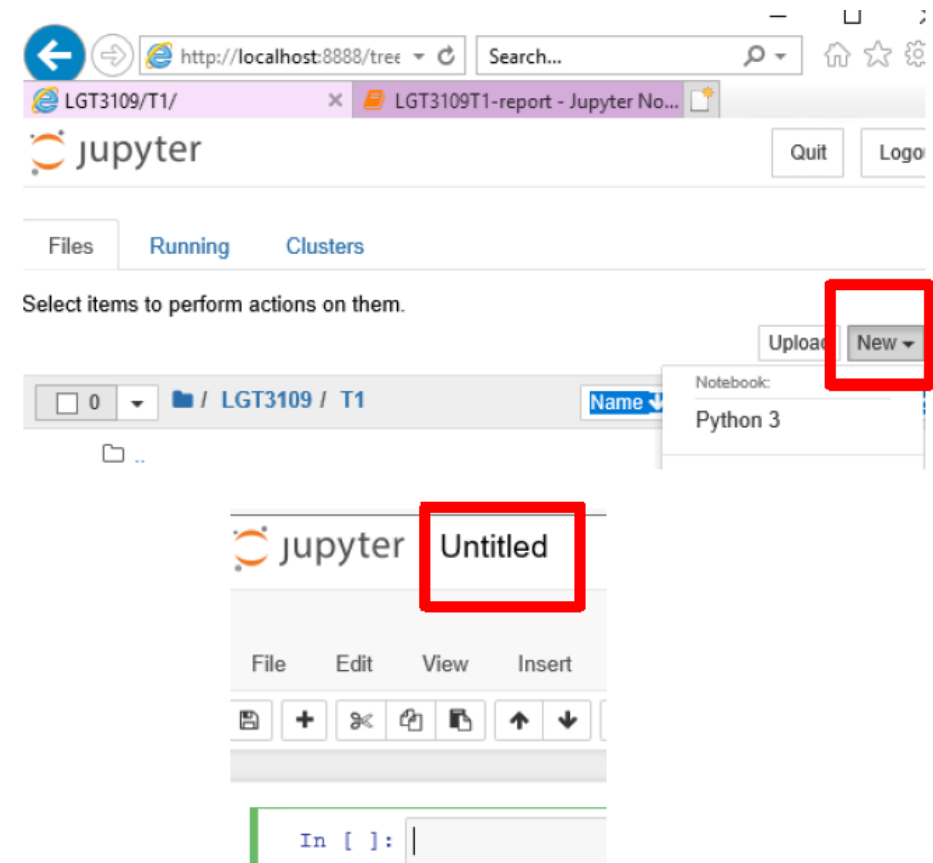
`Enter`: enter edit mode

`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)
 - Use tag to insert a figure in a markdown cell
 -

```

```