

EE 1004 Foundation of Information Systems and Data Analysis

Semester A 2022/23

Lab 2: Search Engine for Best Flight Route

This lab practice uses “booking the air ticket” for a trip to understand how to co-relate data, graph, matrix, and vector variables in a daily example. In this lab, you have to consider two issues during booking: the **price of the tickets** and the **flight duration**. You are required to **use graphs to build the data structure of the flight networks** and then utilize the graph to **find the best flight routes based on your desires**.



Objectives:

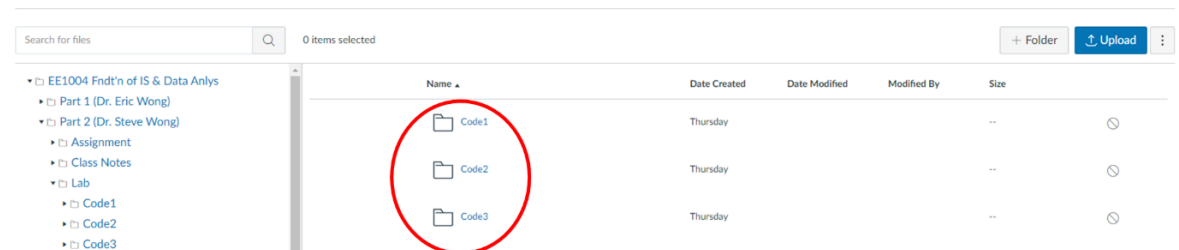
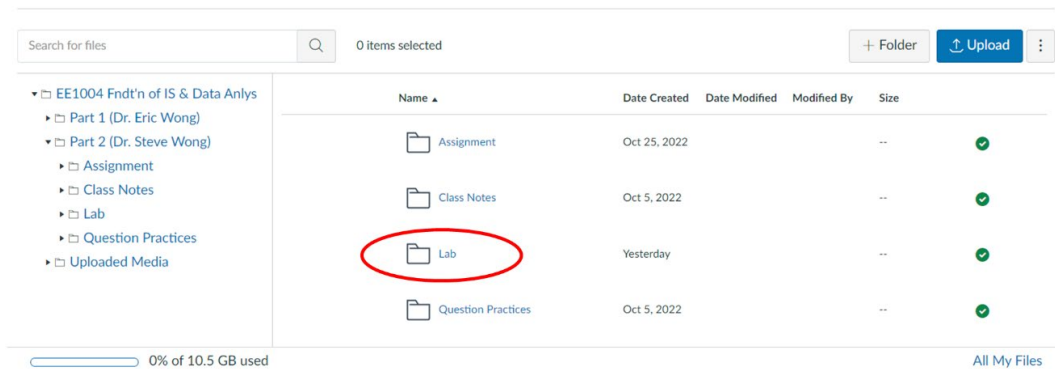
- **Recognize and extract the data** for distinct purposes,
- **Utilize the programming language**, i.e. Python, to build the data structure for manipulation,
- **Identify the representation of graph, matrix, and vector** in program syntax,
- **Implement Dijkstra’s algorithm** in graph theory for practical use cases.

Pre-requisites:

- **Python Code**

Python is a widely used, high-level, general-purpose programming language. Compared with traditional programming languages, such as C++ and Java, Python has a distinctive characteristic that emphasizes the readability of code with intuitive syntax and a comprehensive standard library. It is consistently rated as one of the most popular programming languages that beginners welcome and supports multiple programming paradigms for a variety of applications, including data analysis, web development, and artificial intelligence (AI) implementation.

Throughout the lab session, we provide the **templates of the python code** for each experiment task **inside the Lab folder in Canvas** sequentially and included as follows:

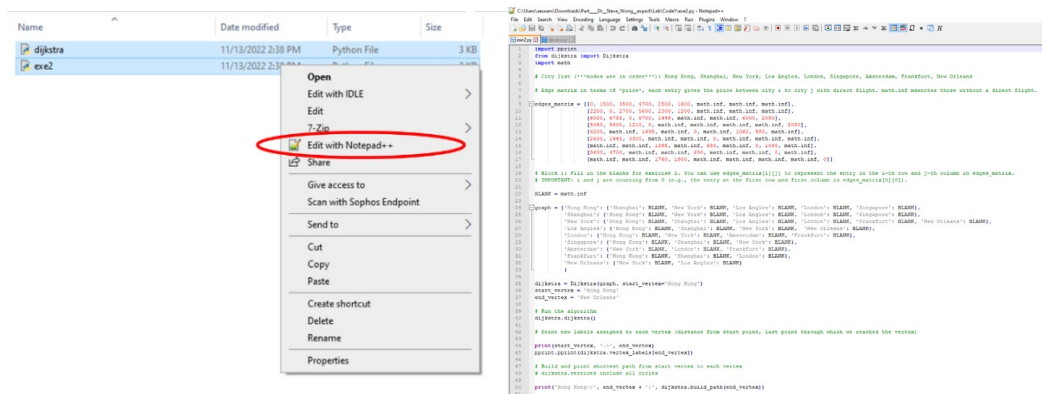


dijkstra.py: The file with the full implementation of the Dijkstra algorithm introduced in the previous lectures. The algorithm returns the shortest paths from one vertex to others in the given graph.

exe2/3/4.py: The script assigned for each exercise and included the Dijkstra algorithm (called from modules in dijkstra.py). It is a template provided to you for filling in the variable and initializing the graph and its corresponding matrices and thus achieving the results for each exercise.

example.py: The example module gives you simple examples of how to initialize the graph adjacent matrix, and the price/duration edge matrix following the program syntax.

To access or edit the provided **python code (.py)** and **data file (.md)**, please use the **built-in text editor (i.e. Notepad++)** in lab computers as shown in the following figures. **Select the desired files and right click** for calling the options menu, and click **Edit with Notepad++**.

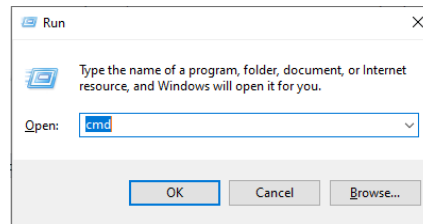


BE REMEMBER to save your code before the execution.

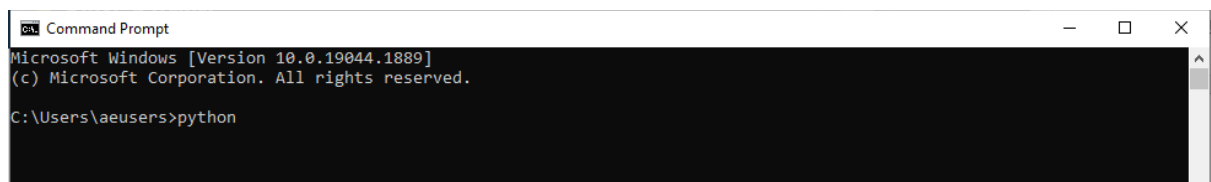
- Python Compiler/Interpreter

To execute your program code, Python Compiler/Interpreter to translate or compile the human-readable source code to machine code (i.e. binary – 1/0) for the processor in your computer. Except for the typical Integrated Development Environment (IDE), such as Microsoft Visual Studio, Eclipse, and PyCharm, the command prompt installed with an interpreter is the optimal solution for validating your program simply and intuitively.

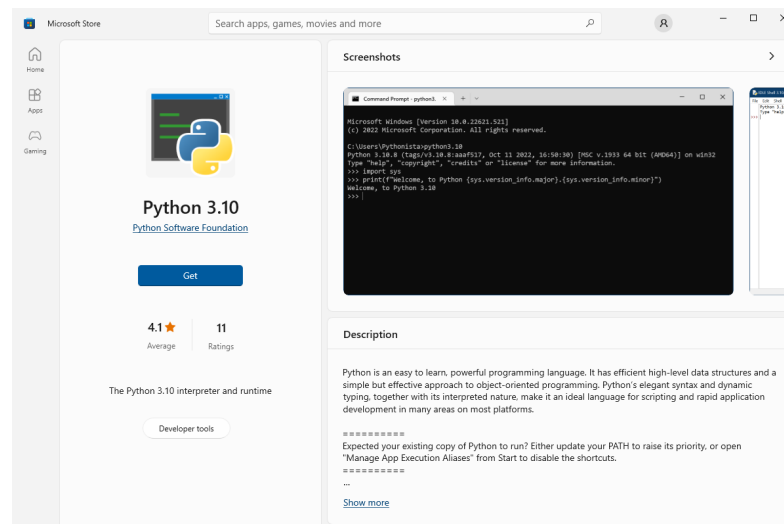
Step 1: Press **Window Logo Key + R** to open the “Run” box, type **cmd** in the bar, and then press **Enter** as shown below.



Step 2: Type the “**python**” command on the cmd window to check whether there is any version of Python interpreter on your computer.



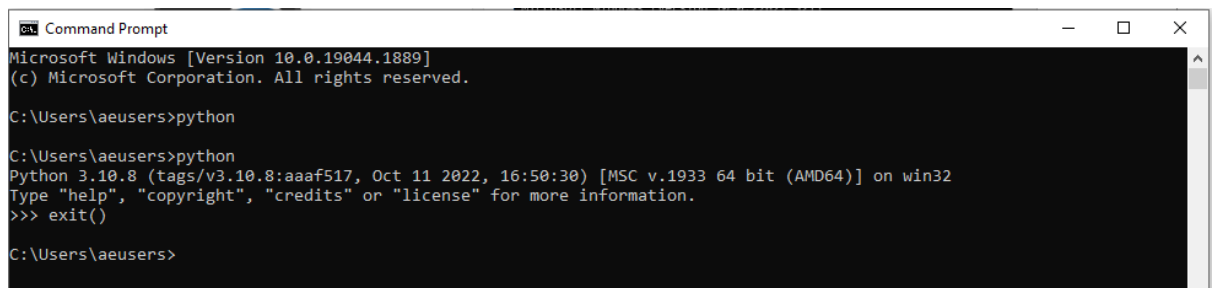
If your computer does not have any version of Python, then the Microsoft Store app will automatically pop out and take you to the latest version of Python in the store.



Step 3: Press the “**Get**” button on the Python App page, and wait for the download and installation. Once the installation is complete, the “**Get**” button will turn into the text of “**You can find these apps in your All apps list**” on the app page, and typing the “**python**” command on the cmd window will return the version of Python interpreter installed as shown below.



Step 4: Type the “**exit()**” command to close the python interpreter on the cmd window. Congratulations! You are ready for the following exercises.



Exercise 1:

For your lovely Christmas trip, you are interested in **9 cities** including Hong Kong, Shanghai, New York, Los Angeles, London, Singapore, Amsterdam, Frankfurt, and New Orleans, and searching for the cheapest or shortest flight between those cities.



Hong Kong



Shanghai



New York



Los Angeles



London



Singapore



Amsterdam



Frankfurt



New Orleans

Based on the **flight plan** attached in the **appendix**, we suggest using the graphs to represent the flight networks between 9 specified cities. Specifically, **each vertex** represents a **city**, and **the edge** denotes **the flight price between cities**.

To **model the flight network** in your program, the **graph representations** are required below:

Adjacent matrix: The adjacent matrix gives the graph skeleton, where each entry $(i, j) = 0/1$ denotes whether there exists a direct flight from City i to City j .

Edge matrix: The edge matrix gives the flight price among cities, where each entry (i, j) corresponds to the ticket price from City i to City j .

Write down the weighted digraph and the corresponding matrices of the flight network for 9 cities.

Exercise 2:

Now, you are going to search for **the cheapest flight between 9 cities**.

- Step 1: Download the provided source code from canvas

Under the “Lab” folder in Canvas, download the entire “Code 1” folder (including 3 python codes of “dijkstra.py”, “exe2.py”, and “example.py”) to your desired destination.

202209EE1004 > Files > Part 2 (Dr. Steve Wong) > Lab > Code1

Search for files	Q	0 Items selected	+ Folder	Upload	
Name	Date Created	Date Modified	Modified By	Size	
dijkstra.py	Thursday	Thursday	Fan FENG	3 KB	✓
example.py	4:26pm	4:26pm	Fan FENG	917 bytes	✓
exe2.py	Thursday	Thursday	Fan FENG	3 KB	✓

Open up all the python files with Notepad++ as introduced in the Pre-requisites session.

- Step 2: Fill in the “BLANK” of the *graph* in “exe2.py”

In “exe2.py”, a completed matrix named *edge_matrix* is provided with the *number* denoting the price of the direct flight between 2 cities, and *math.inf* denoted that there is no direct flight between cities. The row of the matrix represented the start point of your flight, while the destination of your flight is depicted as the matrix column.

Refer to the given *edge_matrix* and follow the instruction in the “exe2.py”, please try to replace the “BLANK” in the *graph* with the corresponding entries in *edge_matrix*.

```
23
24 graph = {'Hong Kong': {'Shanghai': BLANK, 'New York': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Singapore': BLANK},
25          'Shanghai': {'Hong Kong': BLANK, 'New York': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Singapore': BLANK},
26          'New York': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Frankfurt': BLANK, 'New Orleans': BLANK},
27          'Los Angeles': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK, 'New Orleans': BLANK},
28          'London': {'Hong Kong': BLANK, 'New York': BLANK, 'Amsterdam': BLANK, 'Frankfurt': BLANK},
29          'Singapore': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK},
30          'Amsterdam': {'New York': BLANK, 'London': BLANK, 'Frankfurt': BLANK},
31          'Frankfurt': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
32          'New Orleans': {'New York': BLANK, 'Los Angeles': BLANK}
33 }
```

Furthermore, “example.py” provided a simple demonstration of 3 cities with completed *edge_matrix* and *graph* whereby you take it as a reference for filling in the variables of the *graph* in “exe2.py”

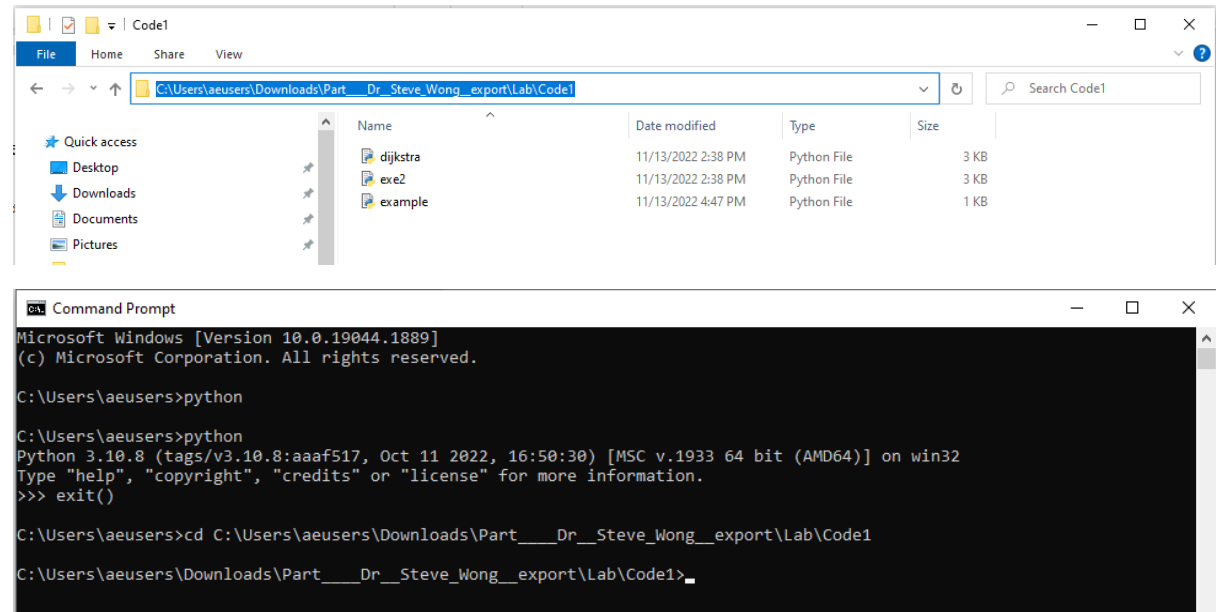
```
7 edges_matrix = [[0, 1500, 3500],
8                 [2200, 0, 2700],
9                 [6000, 6755, 0]]
10
11 graph = {'Hong Kong': {'Shanghai': edges_matrix[0][1], 'New York': edges_matrix[0][2]},
12          'Shanghai': {'Hong Kong': edges_matrix[1][0], 'New York': edges_matrix[1][2]},
13          'New York': {'Hong Kong': edges_matrix[2][0], 'Shanghai': edges_matrix[2][1]}
14
15 }
```

And, save your finalized “exe2.py” for the execution in the command prompt.

- **Step 3: Direct the command prompt to the directory you stored the python codes**

Copy the directory path where you stored the python codes, and type the “cd (your directory path)\Code 1” command on the cmd window

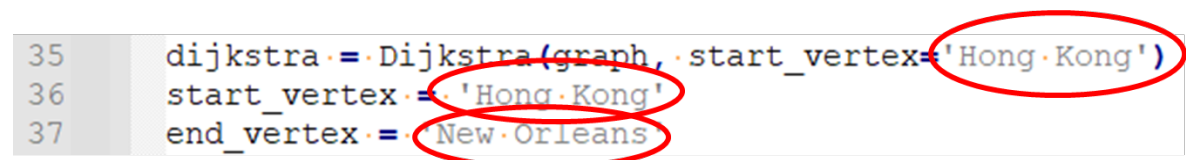
e.g. C:\Users\aeusers\Downloads\Part____Dr__Steve_Wong__export\Lab\Code1



- **Step 4: Run the “exe2.py” to obtain the results**

Type the “python exe2.py” to validate your program code, and obtain the results for the checkpoint.

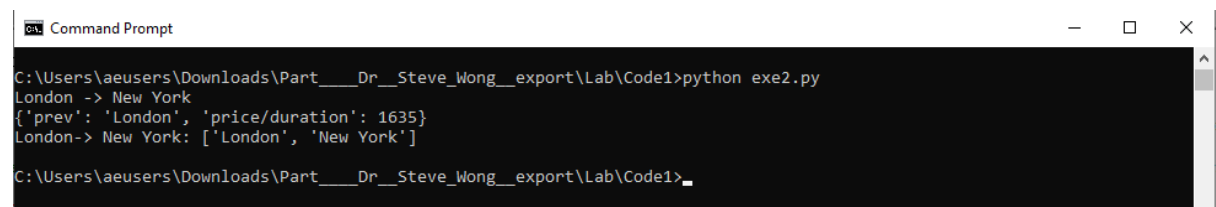
Modify the code with different *start_vertex* and *end_vertex* to obtain different results



e.g. The Cheapest flight from London to New York

d (London, Hong Kong) = \$1635

Shortest path = London → New York



Checkpoint 1: Run the code and find the cheapest flight from New Orleans to Singapore.

Exercise 3:

Sequentially, you are looking for the **shortest flight between 9 cities**.

- **Step 1: Download the provided source code and duration information from canvas**
Download the entire “Code 2” folder (including 3 files of “dijkstra.py”, “exe3.py”, and “duration.md”) to your desired destination. And, open all files with Notepad++.

202209EE1004 > Files > Part 2 (Dr. Steve Wong) > Lab > Code2

Search for files	Q	0 items selected	+ Folder	Upload	
- EE1004 Fnd'n of IS & Data Anlys - Part 1 (Dr. Eric Wong) - Part 2 (Dr. Steve Wong) - Assignment - Class Notes - Lab - Code1 - Code2 - Code3					
Name	Date Created	Date Modified	Modified By	Size	
dijkstra.py	Thursday	Thursday	Fan FENG	3 KB	✓
duration.md	Thursday	Thursday	Fan FENG	820 bytes	✓
exe3.py	Thursday	Thursday	Fan FENG	2 KB	✓

- **Step 2: Create the edge matrix for the flight duration between 9 cities, and fill in the “BLANK” of the *graph_duration* in “exe3.py”**

Create another edge matrix named *edge_matrix_duration* in “exe3.py”. Compared with the *edge_matrix* in “exe2.py”, the *edge_matrix_duration* owns the same syntax except the *number* is used to denote the duration of the direct flight between 2 cities. The duration of the direct flight across 9 cities can be found in the “duration.md”.

<pre> 9 edges_matrix_duration = [], 10 11 12 13 14 15 16 17 18 </pre>	<pre> 1 Hong Kong to: 2 Shanghai: 2.75 3 New York: 15.67 4 Los Angeles: 12.75 5 London: 14.5 6 Singapore: 4 7 8 Shanghai to: 9 Hong Kong: 2.75 10 New York: 14.75 11 Los Angeles: 12.5 12 London: 15.0 13 Singapore: 5.5 14 </pre>
--	--

And thus, the “BLANKS” of *graph_duration* in “exe3.py” should be filled with the corresponding entries in *edge_matrix_duration*. Be aware of the notation of entries in *edge_matrix_duration*.

```

24 graph_duration = {'Hong Kong': {'Shanghai': BLANK, 'New York': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Singapore': BLANK},
25                  'Shanghai': {'Hong Kong': BLANK, 'New York': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Singapore': BLANK},
26                  'New York': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'Los Angeles': BLANK, 'London': BLANK, 'Frankfurt': BLANK, 'New Orleans': BLANK},
27                  'Los Angeles': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK, 'New Orleans': BLANK},
28                  'London': {'Hong Kong': BLANK, 'New York': BLANK, 'Amsterdam': BLANK, 'Frankfurt': BLANK},
29                  'Singapore': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK},
30                  'Amsterdam': {'New York': BLANK, 'London': BLANK, 'Frankfurt': BLANK},
31                  'Frankfurt': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
32                  'New Orleans': {'New York': BLANK, 'Los Angeles': BLANK}
33

```

Remember to save your “exe3.py” before the execution in the command prompt.

- **Step 3: Direct the command prompt to the directory you stored the python codes**

Copy the directory path where you stored the python codes, and type the “cd (your directory path)\Code 2” command on the cmd window

e.g. C:\Users\aeusers\Downloads\Part____Dr__Steve_Wong__export\Lab\Code2

- **Step 4: Run the “exe3.py” to obtain the results**

Type the “python exe3.py” to validate your program code, and obtain the results for the checkpoint.

e.g. The shortest duration from Amsterdam to Hong Kong

d(Amsterdam, Hong Kong) = 24.83 hours

Shortest path = Amsterdam → New York → Hong Kong



```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Downloads\Lab\Code2>python exe3.py
Amsterdam -> Hong Kong
{'prev': 'New York', 'price/duration': 24.83}
Amsterdam-> Hong Kong: ['Amsterdam', 'New York', 'Hong Kong']
C:\Users\User\Downloads\Lab\Code2>
```

Checkpoint 2: Run the code and find the flight with the shortest duration from Hong Kong to New Orleans.

Exercise 4:

Alternatively, your family/friends suggested **two cities including Taipei and Tokyo** for your consideration. In addition to the aforementioned 9 cities, you are requested to **find out the cheapest and shortest flight** of the revised flight network including Taipei and Tokyo.



Taipei



Tokyo

- **Step 1: Download the provided source code and duration information from canvas**

Download the entire “Code 3” folder (including 3 files of “disjkstra.py”, “exe4.py”, and “Taipei&Tokyo.md”) to your desired destination. And, open all files with Notepad++.

202209EE1004 > Files > Part 2 (Dr. Steve Wong) > Lab > Code3

Search for files	Q	0 items selected	+ Folder	Upload	
Name					
Date Created	Date Modified	Modified By	Size		
dijkstra.py	Thursday	Thursday	Fan FENG	3 KB	✓
exe4.py	Thursday	Thursday	Fan FENG	4 KB	✓
Taipei&Tokyo.md	Thursday	Thursday	Fan FENG	804 bytes	✓

- **Step 2: Modify the edge matrices in “exe2.py” and “exe3.py” by the addition of the flight information of Taipei & Tokyo, and fill in the “BLANK” of the *graph_duration* and *graph_price* in “exe4.py”**

Based on edge matrices in “exe2.py” and “exe3.py”, new matrices named *edge_matrix_price* and *edge_matrix_duration* are required to modify and adapt the new flight network including the new vertices of Taipei and Tokyo. “Taipei&Tokyo.md” contained the flight price and duration of the direct flight in Taipei and Tokyo, and the entries of *edge_matrix_price* and *edge_matrix_duration* should be adjusted and modified accordingly.

```

8 edges_matrix_price = [],
9
10
11
12
13
14
15
16
17
18
19

```

```

35 edges_matrix_duration = [],
36
37
38
39
40
41
42
43
44
45
46

```

<pre> 1 # Price 2 Taipei: 3 Shanghai -> Taipei: 1500 4 Hong Kong -> Taipei: 900 5 New York -> Taipei: 3700 6 Tokyo -> Taipei: 2300 7 Taipei -> Hong Kong: 1450 8 Taipei -> Shanghai: 1800 9 Taipei -> London: 2360 10 11 12 Tokyo: 13 Shanghai -> Tokyo: 3300 14 Hong Kong -> Tokyo: 1200 15 Los Angles -> Tokyo: 5580 16 Tokyo -> Hong Kong: 2900 17 Tokyo -> New York: 4500 18 Tokyo -> Taipei: 2300 19 </pre>	<pre> 21 # Duration 22 Taipei: 23 Shanghai -> Taipei: 2.5 24 Hong Kong -> Taipei: 1.9 25 New York -> Taipei: 12.67 26 Tokyo -> Taipei: 2.8 27 Taipei -> Hong Kong: 1.67 28 Taipei -> Shanghai: 2.4 29 Taipei -> London: 16.0 30 31 32 Tokyo: 33 Shanghai -> Tokyo: 3.5 34 Hong Kong -> Tokyo: 3.67 35 Los Angles -> Tokyo: 15.67 36 Tokyo -> Hong Kong: 4.0 37 Tokyo -> New York: 12.25 38 Tokyo -> Taipei: 2.8 39 </pre>
---	--

Correspondingly, the “BLANKS” of *graph_price* and *graph_duration* should be filled with the corresponding entries in *edge_matrix_price* and *edge_matrix_duration* respectively. Be aware of the notation of entries in *edge_matrix_price* and *edge_matrix_duration*.

```

22 graph_price = {'Hong Kong': {'Shanghai': BLANK, 'New York': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
23               {'Shanghai': {'Hong Kong': BLANK, 'New York': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
24               {'New York': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
25               {'Los Angles': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK, 'New Orleans': BLANK, 'Tokyo': BLANK},
26               {'London': {'Hong Kong': BLANK, 'New York': BLANK, 'Amsterdam': BLANK, 'Frankfurt': BLANK},
27               {'Singapore': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK},
28               {'Amsterdam': {'New York': BLANK, 'London': BLANK, 'Frankfurt': BLANK},
29               {'Frankfurt': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
30               {'New Orleans': {'New York': BLANK, 'Los Angles': BLANK},
31               {'Taipei': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
32               {'Tokyo': {'Hong Kong': BLANK, 'New York': BLANK, 'Taipei': BLANK}
33               }
34
49 graph_duration = {'Hong Kong': {'Shanghai': BLANK, 'New York': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
50                  {'Shanghai': {'Hong Kong': BLANK, 'New York': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
51                  {'New York': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'Los Angles': BLANK, 'London': BLANK, 'Singapore': BLANK, 'Taipei': BLANK, 'Tokyo': BLANK},
52                  {'Los Angles': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK, 'New Orleans': BLANK, 'Tokyo': BLANK},
53                  {'London': {'Hong Kong': BLANK, 'New York': BLANK, 'Amsterdam': BLANK, 'Frankfurt': BLANK},
54                  {'Singapore': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'New York': BLANK},
55                  {'Amsterdam': {'New York': BLANK, 'London': BLANK, 'Frankfurt': BLANK},
56                  {'Frankfurt': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
57                  {'New Orleans': {'New York': BLANK, 'Los Angles': BLANK},
58                  {'Taipei': {'Hong Kong': BLANK, 'Shanghai': BLANK, 'London': BLANK},
59                  {'Tokyo': {'Hong Kong': BLANK, 'New York': BLANK, 'Taipei': BLANK}
60                  }
61

```

- **Step 3: Direct the command prompt to the directory you stored the python codes**

Copy the directory path where you stored the python codes, and type the “cd (your directory path)\Code 3” command on the cmd window

e.g. C:\Users\aeusers\Downloads\Part___Dr__Steve_Wong__export\Lab\Code3

- **Step 4: Run the “exe4.py” to obtain the results**

Type the “python exe4.py” to validate your program code, and obtain the results for the checkpoint.

e.g. The **cheapest flight** from **Los Angles** to **Taipei**

d (Los Angles, Taipei) = \$4910

Shortest path = Los Angles → New York → Taipei

The **shortest duration** from **Tokyo** to **Amsterdam**

d(Tokyo, Amsterdam) = 20 hours

Shortest path = Tokyo → Hong Kong → London → Amsterdam

```
C:\WINDOWS\system32\cmd.exe
C:\Users\User\Downloads\Lab\Code3>python exe4.py
Lowest Price:
Los Angeles -> Taipei
{'prev': 'New York', 'price/duration': 4910}
Los Angeles-> Taipei: ['Los Angeles', 'New York', 'Taipei']
Shortest Duration:
Tokyo -> Amsterdam
{'prev': 'London', 'price/duration': 20.0}
Tokyo-> Amsterdam: ['Tokyo', 'Hong Kong', 'London', 'Amsterdam']
C:\Users\User\Downloads\Lab\Code3>
```

Checkpoint 3: Run the code and find 1) the cheapest price from New Orleans to Taipei; 2) the flight with the shortest duration from New Orleans to Tokyo.

Lab Report:

Students are required to submit an individual lab report to Canvas in week (TBC). All reports should be typed on A4-size paper. And the contents of the lab report should be included in the following:

- The results/answers of all exercises, with detailed program code of the extracted data (Exercise 2/3/4) and detail steps/graphs (Exercise 1). Present the results with a screen capture or typed content clearly and concisely.
- Find out the cheapest flight with no more than 2 stops from New Orleans to Hong Kong. You can calculate it based on the graph using the Dijkstra algorithm. Please show the calculation steps in your report.

Appendix:

Flight Plan – Direct Flights

From Amsterdam:

Flight			Price (\$)
Amsterdam	→	Canada	3200
Amsterdam	→	Denmark	500
Amsterdam	→	Frankfurt	1080
Amsterdam	→	Finland	400
Amsterdam	→	London	580
Amsterdam	→	New Zealand	4450
Amsterdam	→	New York	1395

From Australia:

Flight			Price (\$)
Australia	→	Chiang Mai	2600
Australia	→	Denmark	4200
Australia	→	Las Vegas	5100
Australia	→	London	3540
Australia	→	Paris	2890
Australia	→	Rome	1780

From Canada:

Flight			Price (\$)
Canada	→	Finland	3350
Canada	→	Hong Kong	4340
Canada	→	Los Angeles	2400
Canada	→	Las Vegas	2150

From Denmark:

Flight			Price (\$)
Denmark	→	Canada	2440
Denmark	→	Frankfurt	890
Denmark	→	Finland	640
Denmark	→	Hong Kong	2700
Denmark	→	Iceland	680

From Frankfurt:

Flight		Price (\$)
Frankfurt	→ Egypt	1690
Frankfurt	→ Hong Kong	3600
Frankfurt	→ Las Vegas	2110
Frankfurt	→ London	690
Frankfurt	→ Shanghai	4700
Frankfurt	→ Sydney	1880

From Hong Kong:

Flight		Price (\$)
Hong Kong	→ Florence	1340
Hong Kong	→ Greece	1210
Hong Kong	→ Los Angeles	4700
Hong Kong	→ London	2500
Hong Kong	→ Lyon	890
Hong Kong	→ New York	3500
Hong Kong	→ Shanghai	1500
Hong Kong	→ Singapore	1600

From London:

Flight		Price (\$)
London	→ Amsterdam	1060
London	→ Berlin	760
London	→ Frankfurt	580
London	→ Hong Kong	4200
London	→ New York	1635
London	→ New Zealand	2440

From Las Vegas:

Flight		Price (\$)
Las Vegas	→ Beijing	5300
Las Vegas	→ Osaka	5200
Las Vegas	→ Paris	3380
Las Vegas	→ Seoul	4890

From Los Angeles:

Flight			Price (\$)
Los Angeles	→	Hong Kong	5080
Los Angeles	→	Iceland	3780
Los Angeles	→	New Orleans	2080
Los Angeles	→	New York	1210
Los Angeles	→	Shanghai	5800
Los Angeles	→	Seoul	5500

From New York:

Flight			Price (\$)
New York	→	Australia	5800
New York	→	Berlin	5600
New York	→	Frankfurt	6000
New York	→	Finland	5900
New York	→	Hong Kong	6000
New York	→	London	1449
New York	→	Los Angeles	5700
New York	→	Nagoya	6450
New York	→	New Orleans	2080
New York	→	Shanghai	6755

From New Orleans:

Flight			Price (\$)
New Orleans	→	Bangkok	5600
New Orleans	→	Chiang Mai	5700
New Orleans	→	Los Angeles	1900
New Orleans	→	New York	1760
New Orleans	→	Norway	2700
New Orleans	→	Osaka	5340

From New Zealand:

Flight			Price (\$)
New Zealand	→	Bali	4410
New Zealand	→	Hawaii	4590
New Zealand	→	Malaysia	5300
New Zealand	→	Nanjing	5700

From Shanghai:

Flight			Price (\$)
Shanghai	→	Barcelona	4600
Shanghai	→	Hong Kong	2200
Shanghai	→	Los Angeles	5600
Shanghai	→	London	2300
Shanghai	→	New York	2700

Shanghai	→	Rome	3890
Shanghai	→	Scotland	4310
Shanghai	→	Shanghai	1200

From Sydney:

Flight			Price (\$)
Sydney	→	Chicago	3580
Sydney	→	Denmark	2320
Sydney	→	Moscow	2890
Sydney	→	Scotland	3400
Sydney	→	Prague	2180
Sydney	→	San Francisco	4200

From Singapore:

Flight			Price (\$)
Singapore	→	Beijing	3130
Singapore	→	Berlin	3990
Singapore	→	Calgary	5800
Singapore	→	Hong Kong	2600
Singapore	→	Egypt	3470
Singapore	→	Florence	2700
Singapore	→	New York	3300
Singapore	→	Osaka	2450
Singapore	→	Shanghai	1445

From Kyiv:

Flight			Price (\$)
Kyiv	→	Moscow	0

~End~