Dictionaries

Section 3

Chapter 5

Quiz 12

Dictionaries

- Python dictionary is defined as
  - A collection of comma separated pairs
  - Of the form key:value
  - Enclosed in curly braces.

- Value associated with *key1* is given by the expression *dictionaryName*[*key1*].

- Dictionary keys must be immutable objects.
  - Lists and sets cannot serve as keys!

Dictionaries

```
>>> cityu = {"name": "CityU HK", "age": 37, "Colleges": ["College of Business", "College of Science"]}
>>>
>>> cityu["name"]
'CityU HK'
>>> cityu["age"]
37
>>> cityu["Colleges"]
['College of Business', 'College of Science']
```

```
>>> "age" in cityu
True
>>> 37 in cityu
False
```

Dictionaries

```
>>> cityu = {"name": "CityU HK", "age": 37, "Colleges": ["College of Business", "College of Science"]}
>>> len(cityu)
3
>>> list(cityu)
['name', 'age', 'Colleges']
>>> list(cityu.values())
['CityU HK', 37, ['College of Business', 'College of Science']]
>>> list(cityu.items())
[('name', 'CityU HK'), ('age', 37), ('Colleges', ['College of Business', 'College of Science'])]
```

The *dict* Function

- List of 2-item lists or 2-item tuples can be converted to a dictionary with *dict* function

```
>>> cu = [('name', 'CityU HK'), ('age', 37), ('Colleges', ['College of Business', 'College of Science'])]
>>> dict(cu)
{'name': 'CityU HK', 'age': 37, 'Colleges': ['College of Business', 'College of Science']}
```

Dictionary shortens scripts

```
#Avoid long if-elif statements
def schoolDistrict(schoolName):
    schools = {"CityU":"Kowloon Tong", "CUHK":"Ma Liu Shui", "HKUST":"Clear Water Bay"}
    return schools[schoolName]
```

Program with long *if-elif* statement can be simplified with use of a dictionary.

```
def main():
    ## Determine an admission fee based on age group.
    print("Enter the person's age group ", end="")
    ageGroup = input("(child, minor, adult, or senior): ")
    print("The admission fee is", determineAdmissionFee(ageGroup), "dollars." )

def determineAdmissionFee(ageGroup):
    if ageGroup == "child":      # age < 6
        return 0                 # free
    elif ageGroup == "minor":    # age 6 to 17
        return 5                 # $5
    elif ageGroup == "adult":    # age 18 to 64
        return 10
    elif ageGroup == "senior":   # age >= 65
        return 8

    ...
```

```
def determineAdmissionFee(ageGroup):
    dict = {"child":0, "minor":5, "adult":10, "senior":8}
    return dict[ageGroup]
```

d[*key1*]	returns the value associated with *key1*. Raises an error if *key1* is not a key of d.

Dictionary Comprehension

- Dictionaries can be created with dictionary comprehension.

```
>>> {x: x**2 for x in range(4)}
{0: 0, 1: 1, 2: 4, 3: 9}
```

Installing pip

Install pip

- pip is the standard Python package management system.
- https://pip.pypa.io/en/stable/installation/
- Newer version Python should come with pip.

Add Python to Path (Windows)

- If you installed Python in Windows using the default installation options, the path to the Python executable wasn't added to the Windows **Path variable**.

- Refer to the **PIP_Windows.pdf** on Canvas.

- https://geek-university.com/python/add-python-to-the-windows-path/

Install pip for windows

- If pip is not installed.

- Put the get-pip.py (from Canvas) in the same directory as Python.
  - Normally under C:\Users\...\AppData\Local\Programs\Python\Python###

- In the Command Prompt, type in
  - python get-pip.py

Install pip for mac

- Open the mac terminal.

- Copy and paste the following to the terminal window.
  1. curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  2. python3 get-pip.py

```
tj@TJs-MacBook-Pro ~ % curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2108k  100 2108k    0     0  4903k      0 --:--:-- --:--:-- --:--:-- 4985k
tj@TJs-MacBook-Pro ~ % python3 get-pip.py
Collecting pip
  Downloading pip-21.3.1-py3-none-any.whl (1.7 MB)
     |████████████████████████████████| 1.7 MB 5.8 MB/s
Collecting wheel
  Downloading wheel-0.37.0-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel, pip
  Attempting uninstall: pip
    Found existing installation: pip 21.2.3
    Uninstalling pip-21.2.3:
      Successfully uninstalled pip-21.2.3
Successfully installed pip-21.3.1 wheel-0.37.0
```

Install packages from pip

- `pip install some-package-name`
  - `pip install numpy`
  - `pip install pandas`
  - `pip install matplotlib`

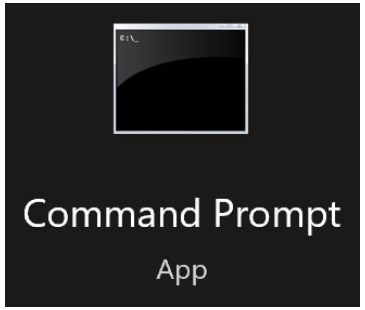
pandas

pandas

- pandas is a library designed for data analysis and manipulation.
- The name originates from the term panel data.
- The library was developed by Wes McKinney, while working at AQR Capital Management.
  - Online book: Python for data analysis.

Install pandas on Windows

- Search for ‘cmd’ in the windows finder.



- Open the Command Prompt.

- Enter **`pip install pandas`**

```
>pip install pandas
```

Example 1: accessing CSV files through pandas

```python
import pandas as pd

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

diff = male - female
```

wages.txt - Notepad

File Edit Format View Help

```
Year,Male,Female
2011,14200,11000
2012,15000,11700
2013,15800,12200
2014,16500,12700
2015,17600,13600
2016,18400,14100
2017,19100,14700
2018,19900,15300
2019,20700,15900
2020,20900,16200
```

```
>>> data
   Year   Male  Female
0  2011  14200   11000
1  2012  15000   11700
2  2013  15800   12200
3  2014  16500   12700
4  2015  17600   13600
5  2016  18400   14100
6  2017  19100   14700
7  2018  19900   15300
8  2019  20700   15900
9  2020  20900   16200
>>>
>>> year
0    2011
1    2012
2    2013
3    2014
4    2015
5    2016
6    2017
7    2018
8    2019
9    2020
Name: Year, dtype: int64
>>>
>>> year[2]
2013
```

```
>>> diff
0    3200
1    3300
2    3600
3    3800
4    4000
5    4300
6    4400
7    4600
8    4800
9    4700
dtype: int64
>>>
>>> diff[5]
4300
```

Data Visualization

Motivation

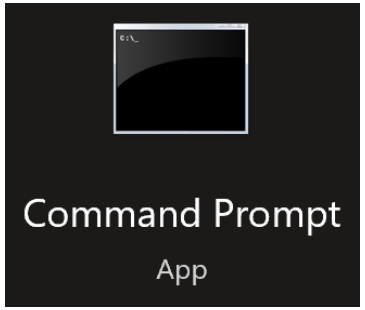
- Data visualization is the presentation of data in a graphical format.

- Data visualization allows us to quickly interpret the data and get some business insight on their effect.

- As data volumes grow, visualization becomes a necessity rather than a luxury.

Install matplotlib on Windows

- Search for ‘cmd’ in the windows finder.



- Open the Command Prompt.

- Enter **`pip install matplotlib`**

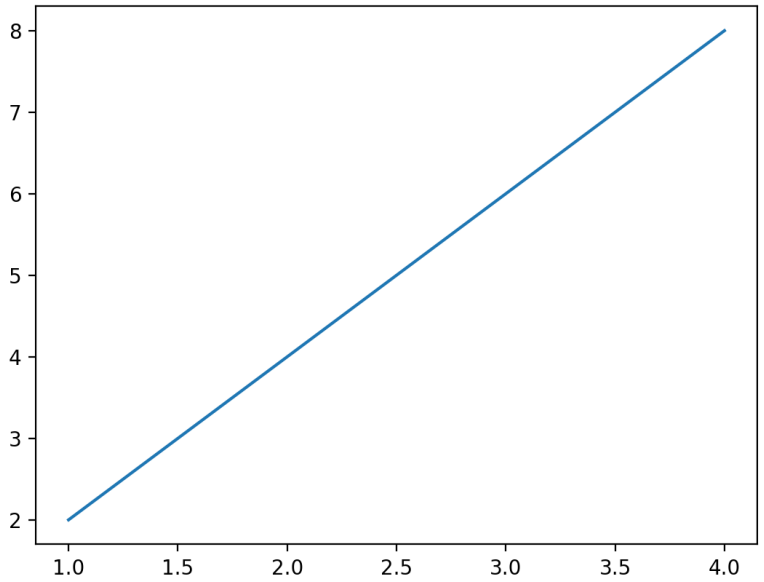
```
>pip install matplotlib
```

plt.plot(x,y)

- **`plot(xvalues, yvalues)`**

- Example 2: plotting a line.

```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4]
y = [2, 4, 6, 8]

plt.plot(x,y)
plt.show()
```



Example 3: Median monthly wages

- Visualize the median monthly wages of males and females in Hong Kong from 2011 to 2020.

wages.txt - Notepad

File Edit Format View Help

```
Year,Male,Female
2011,14200,11000
2012,15000,11700
2013,15800,12200
2014,16500,12700
2015,17600,13600
2016,18400,14100
2017,19100,14700
2018,19900,15300
2019,20700,15900
2020,20900,16200
```

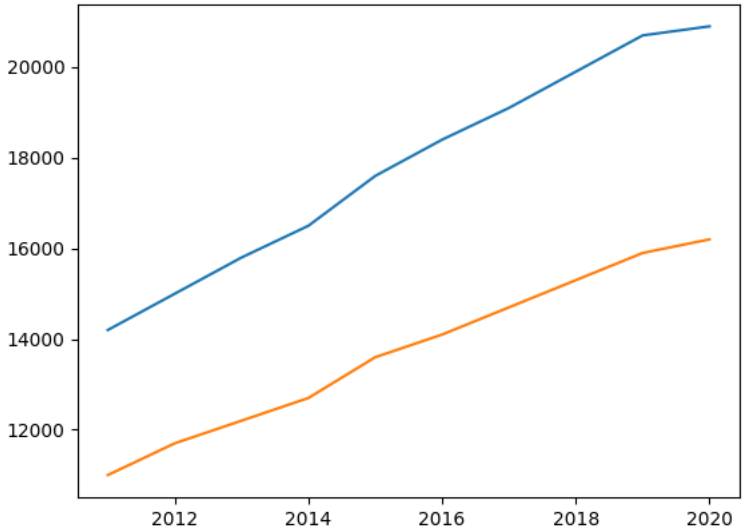
Example 3A: Median monthly wages

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

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

plt.plot(year, male)
plt.plot(year, female)
plt.savefig('wages1.png')  #save to the same directory as the .py file.
plt.show()
```

wages.txt - Notepad

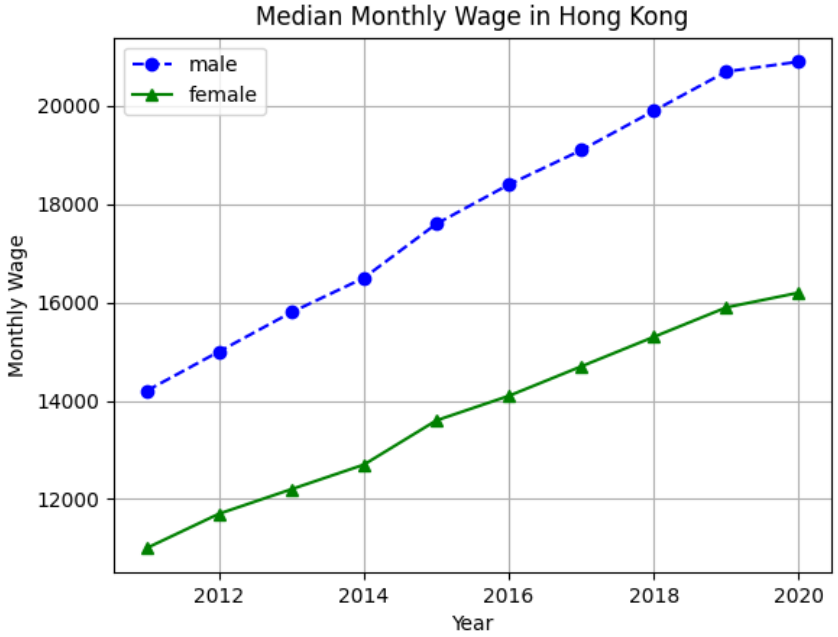
File Edit Format View Help

```
Year,Male,Female
2011,14200,11000
2012,15000,11700
2013,15800,12200
2014,16500,12700
2015,17600,13600
2016,18400,14100
2017,19100,14700
2018,19900,15300
2019,20700,15900
2020,20900,16200
```

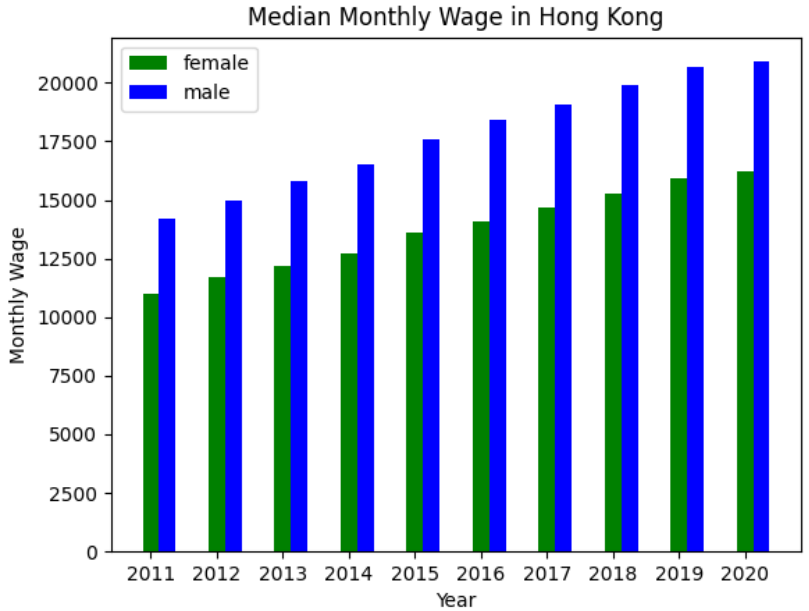


Example 3B: Median monthly wages

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

#print(plt.style.available)
#plt.style.use('seaborn-darkgrid')

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

plt.xlabel('Year')
plt.ylabel('Monthly Wage')
plt.title("Median Monthly Wage in Hong Kong")
plt.plot(year, male, color='b', linestyle='--', marker='o', label='male')
plt.plot(year, female, color='g', marker='^', label='female')
plt.legend()
plt.grid(True)
plt.savefig('wages2.png')
plt.show()
```



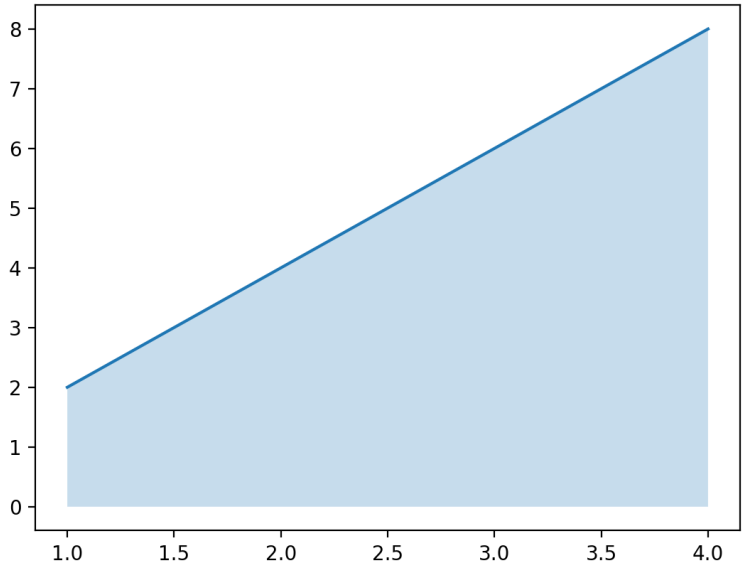
Example 4: Bar chart

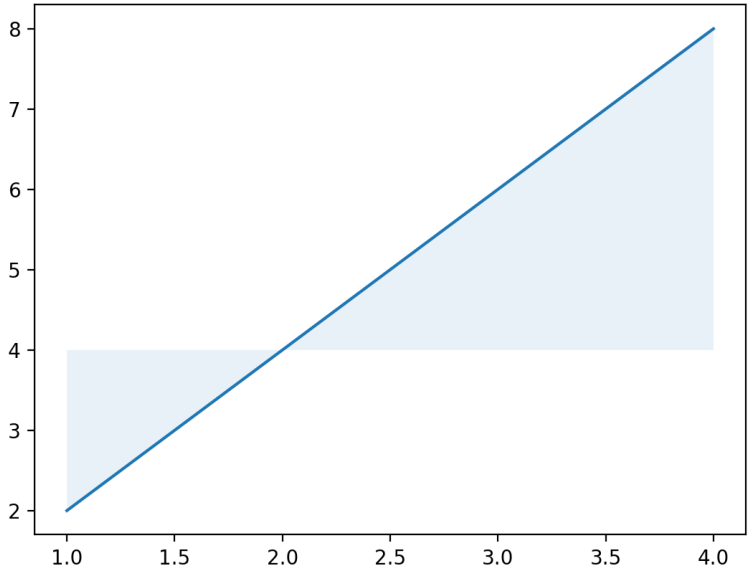



```
#bar charts

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

x_indices = np.arange(len(year))
width = 0.25

plt.xlabel('Year')
plt.ylabel('Monthly Wage')
plt.title("Median Monthly Wage in Hong Kong")
plt.bar(x_indices, female, width = width, color='g', label='female')
plt.bar(x_indices+width, male, width = width, color='b', linestyle='--', label='male')
plt.legend()
plt.xticks(ticks=x_indices, labels = year)
plt.savefig('wages3.png')
plt.show()
```

Example 5: filling areas

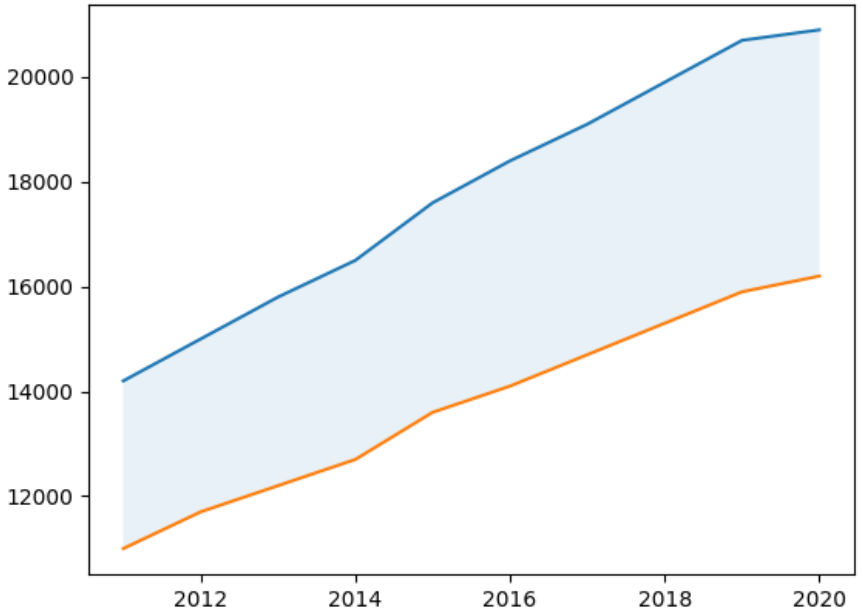
```
from matplotlib import pyplot as plt

x = [1, 2, 3, 4]
y = [2, 4, 6, 8]

plt.plot(x,y)
plt.fill_between(x, y, 4, alpha = 0.1)

#The 3rd arguement is 0 by default.
#plt.fill_between(x, y, alpha = 0.25) generates the top graph.
#alpha controls the see-through level of the fill.

plt.show()
```





Example 6: filling areas between two lines

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

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

plt.plot(year, male)
plt.plot(year, female)

plt.fill_between(year, male, female,
                 where=(male > female),
                 interpolate=True, alpha=0.1)

plt.savefig('wages4.png')
plt.show()
```



Example 7: stack plot

```
#stackplot
import pandas as pd
from matplotlib import pyplot as plt

data = pd.read_csv("wages.txt")
year = data["Year"]
male = data["Male"]
female = data["Female"]

plt.stackplot(year, male, female)

plt.savefig('wages5.png')
plt.show()
```