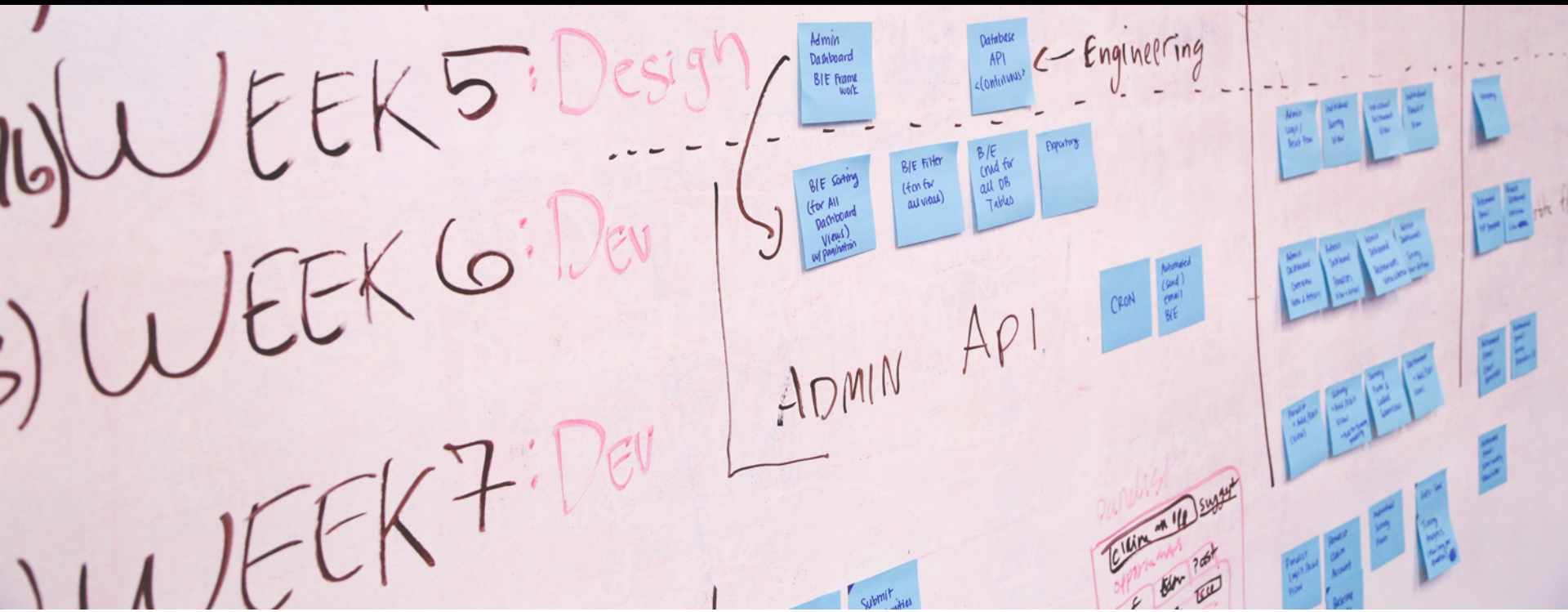


COM5961 DATA DRIVEN PRODUCTS & SERVICES DESIGN:

LESSON 1 - BASIC PYTHON PROGRAMMING II

Bernard Suen
Center for Entrepreneurship
Chinese University of Hong Kong

Today’s agenda.

1. Python data collection structure: list, tuple, dictionary, etc.
2. Use of loop in handling data collection.
3. What is a ‘function’? Why is it so important to programming?
4. Use of module in building program libraries for reusability.
5. Introducing the Pandas program library for data analysis and visualisation.

LISTS AND DICTIONARIES

Basic Data Structures in Python

- List and Tuple
- Dictionary and Set

Tuple and List

- Tuples are arrays enclosed with round brackets for storing multiple variables
- Variables with data types can be stored into a Tuple
- A Tuple operates like a string and therefore can be indexed from the beginning (positive) and the end (negative)
- Tuples are immutable
- Inorder to manipulate a tuple, a new one has to be created
- Tuples can be nested e.g. Tuple2 = (1,2,(3,4),5)
- Due to its rigidity, List is more commonly used than Tuple.
- Lists are like Tubles but are mutable and enclosed with square brackets
- List, similar to a Tuple, operates like a string when it comes to access individual element within the data structure

Looping Operations

In [50]:
```
for i in range(0,10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

Using List to Store Data

In [52]:
```
name_list = ['John',"Mike","Mary","Jane"]
hrs_list = [30.0,40.0,50.0,60.0]
rate_list = [65.0,75.0,65.0,75.0]
index  = 0
for n in name_list:
    hrs = hrs_list[index]
    rate = rate_list[index]
    fee = hrs * rate
    money_made = name_list[index] + " makes $" + str(fee) + "."
    print(hrs,rate,fee, money_made)
    index = index + 1
```

```
30.0 65.0 1950.0 John makes $1950.0.
40.0 75.0 3000.0 Mike makes $3000.0.
50.0 65.0 3250.0 Mary makes $3250.0.
60.0 75.0 4500.0 Jane makes $4500.0.
```

Using Dictionary to Store Data

Set and Dictionary

- Similar to lists and tuples, sets support different Python types
- Sets use {} (braces) to embed values/elements
- Sets do not allow duplicates
- Lists can be converted into sets with the set function
- Dictionary store data in an array of key-value pairs in braces
- For instance, here is a dictionary instance: dict = {“key1”:1,”key2”:2,”key3”:3}
- 1st column representing the key and 2nd column representing the value
- .keys() returns all the keys
- .values.() returns all the values

In [176]:

```
dict1 = {'Peter':80,"David":90,"Mary":100}
print(dict1)
print(dict1.keys())
print(dict1.values())
print(dict1["David"])
```

```
{'Peter': 80, 'David': 90, 'Mary': 100}
dict_keys(['Peter', 'David', 'Mary'])
dict_values([80, 90, 100])
90
```

In [29]:

```
dict1 = {'Peter':80,"David":90,"Mary":100}
for key in dict1:
  print(key, dict1[key])
```

```
Peter 80
David 90
Mary 100
```

Combining List and Dictionary

Try it out

In [5]:
```
student1 = {'id':'1111','name':'David Chan','email':'david@cuhk.edu.hk'} # Python dictionary is similar to JavaScr
student2 = {'id':'1112','name':'Peter Lee','email':'peter@cuhk.edu.hk'}
student3 = {'id':'1113','name':'Mary Fung','email':'mary@cuhk.edu.hk'}
print(student1,student2) # similar to JavaScript document.write
```

```
{'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'} {'id': '1112', 'name': 'Peter Lee', 'email': 'pete
r@cuhk.edu.hk'}
```

In [6]:
```
students = []               # Python list is similar to JavaScript array
students.append(student1)   # Python list append is similar to JavaScript array push
students.append(student2)
students.append(student3)
print(students) # print out a list of dictionaries
```

```
[{'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'}, {'id': '1112', 'name': 'Peter Lee', 'email': 'pe
ter@cuhk.edu.hk'}, {'id': '1113', 'name': 'Mary Fung', 'email': 'mary@cuhk.edu.hk'}]
```

In [205]:
```
for i in students: # Python for loop is similar to JavaScript for (i=0;i<students.length;i++) {document.write...}
    print(i)       # print out a list of dictionaries
```

```
{'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'}
{'id': '1112', 'name': 'Peter Lee', 'email': 'peter@cuhk.edu.hk'}
{'id': '1113', 'name': 'Mary Fung', 'email': 'mary@cuhk.edu.hk'}
```

In [206]:
```
key = 0
for i in students:
    print(i)              # first loop prints out individual dictionary
    for key in i:           # second loop within first loop prints out individual element within each dictionary
        print(key, ':', i[key])
```

```
{'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'}
id : 1111
name : David Chan
email : david@cuhk.edu.hk
{'id': '1112', 'name': 'Peter Lee', 'email': 'peter@cuhk.edu.hk'}
id : 1112
name : Peter Lee
email : peter@cuhk.edu.hk
{'id': '1113', 'name': 'Mary Fung', 'email': 'mary@cuhk.edu.hk'}
id : 1113
name : Mary Fung
email : mary@cuhk.edu.hk
```

Python Function

Functional Decomposition and Abstraction

- A Python function takes input parameters and returns output results
- A function starts with the "def" keyword
- Calling the function will pass control from the calling command to the called function
- Make sure the number of parameters called match with the number of parameters and type
- Enclose programming statements in a function to transform inputs into outputs
- Built-in functions can come with Python (e.g. y = len(list)) and other imported external modules
- A function can take it multiple input parameters (e.g. def Multi(a,b):)
- Global and local variables can be initialized and used inside a Python program but only local variables exist inside a function

```
def name_of_function(parameter1, parameter2…):
    [ Python variable assignment statements,
      mathematical and logical operations ]
    return result
```

Example

```
def calc_avg(x,y):
    result = x + y
    return result
print("Result = ", calc_avg(10,20)
```

Output

Result=30

Python Module

What is a module?

A module is an external Python file which contains the functions that can be imported into the current Python program to be used.

In [7]:
```
1  from convert_temp import *
2  # Move the following functions into convert_temp module file.
3
4  total_temp = 0
5  total_temp = get_daily_celcius()
6  display_avg(total_temp)
```

```
Enter Sun's temperature in celcius:23
Sun's temperature in Fahrenheit:73.4
Enter Mon's temperature in celcius:24
Mon's temperature in Fahrenheit:75.2
Enter Tue's temperature in celcius:28
Tue's temperature in Fahrenheit:82.4
Enter Wed's temperature in celcius:19
Wed's temperature in Fahrenheit:66.2
Enter Thu's temperature in celcius:17
Thu's temperature in Fahrenheit:62.6
Enter Fri's temperature in celcius:21
Fri's temperature in Fahrenheit:69.8
Enter Sat's temperature in celcius:25
Sat's temperature in Fahrenheit:77.0

Average temperature for the week:72.37
```

The Pandas Library

Creating Pandas Data Frames from Lists and Dictionaries

In [18]:

```
import pandas as pd
# Define a list of dictionaries
scores = [{'Student': 'David Chan', 'Jan': 90, 'Feb': 85, 'Mar': 88},
          {'Student': 'Peter Lee',  'Jan': 72, 'Feb': 75, 'Mar': 68},
          {'Student': 'John Lui',   'Jan': 60,  'Feb': 80,  'Mar': 100 }]

df = pd.DataFrame(scores) # Assign the list of dictionaries to a dataframe
df # display dataframe
```

Out[18]:

	Feb	Jan	Mar	Student
0	85	90	88	David Chan
1	75	72	68	Peter Lee
2	80	60	100	John Lui

In [17]:

```
import pandas as pd
students = []  # Python list is similar to JavaScript array
student1 = {'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'}
student2 = {'id': '1112', 'name': 'Peter Lee', 'email': 'peter@cuhk.edu.hk'}
student3 = {'id': '1113', 'name': 'Mary Fung', 'email': 'mary@cuhk.edu.hk'}
students.append(student1)    # Python list append is similar to JavaScript array push
students.append(student2)
students.append(student3)

print(students) # print out a list of dictionaries
df = pd.DataFrame(students) # Assign the list of dictionaries to a dataframe
df # display dataframe
```

```
[{'id': '1111', 'name': 'David Chan', 'email': 'david@cuhk.edu.hk'}, {'id': '1112', 'name': 'Peter Lee', 'email': 'pe
ter@cuhk.edu.hk'}, {'id': '1113', 'name': 'Mary Fung', 'email': 'mary@cuhk.edu.hk'}]
```

Out[17]:

	email	id	name
0	david@cuhk.edu.hk	1111	David Chan
1	peter@cuhk.edu.hk	1112	Peter Lee
2	mary@cuhk.edu.hk	1113	Mary Fung

Saving To and Retrieving From CSV Files

In [24]:
```
import pandas as pd
# Define a list of dictionaries
scores = [{'assignment scores': 'David Chan', 'Jan': 90, 'Feb': 85, 'Mar': 88},
          {'assignment scores': 'Peter Lee',  'Jan': 72, 'Feb': 75, 'Mar': 68},
          {'assignment scores': 'John Wong',  'Jan': 60,  'Feb': 80,  'Mar': 100 }]
df = pd.DataFrame(scores) # Assign the list of dictionaries to a dataframe
df # display dataframe
df.to_csv('students.csv',mode='w',index=False)
df
```

Out[24]:

	Feb	Jan	Mar	assignment scores
0	85	90	88	David Chan
1	75	72	68	Peter Lee
2	80	60	100	John Wong

Reading Data From a CSV File Into a Pandas Data Frame

In [25]:
```
import pandas as pd
df = pd.read_csv("students.csv") # import csv file by using the read_csv functon in Pandas
df # list the dataframe
```

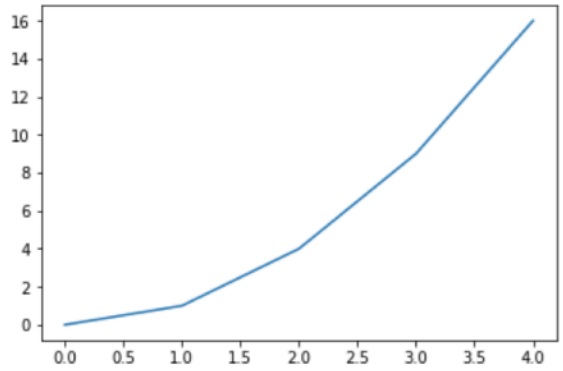
Out[25]:

	Feb	Jan	Mar	assignment scores
0	85	90	88	David Chan
1	75	72	68	Peter Lee
2	80	60	100	John Wong

In [2]:
```
from matplotlib import pyplot as plt
x_values = [0, 1, 2, 3, 4]
y_values = [0, 1, 4, 9, 16]
plt.plot(x_values, y_values)
plt.show()
```

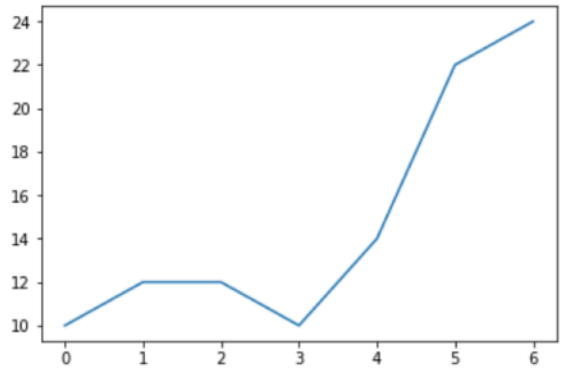
In [2]:

```
from matplotlib import pyplot as plt
x_values = [0, 1, 2, 3, 4]
y_values = [0, 1, 4, 9, 16]
plt.plot(x_values, y_values)
plt.show()
```



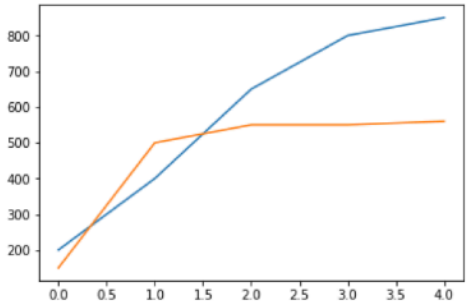
In [3]:

```
from matplotlib import pyplot as plt
days = range(7)
money_spent = [10,12,12,10,14,22,24]
plt.plot(days,money_spent)
plt.show()
```



In [6]:

```
from matplotlib import pyplot as plt
time = [0, 1, 2, 3, 4]
# time = range(5)
salary = [200, 400, 650, 800, 850]
expenses = [150, 500, 550, 550, 560]
plt.plot(time, salary)
plt.plot(time, expenses)
plt.show()
```



In [5]:

```
from matplotlib import pyplot as plt
time = [0, 1, 2, 3, 4]
salary = [200, 400, 650, 800, 850]
expenses = [150, 500, 550, 550, 560]
plt.plot(time, salary, color='purple',linestyle='--')
plt.plot(time, expenses, color='#82edc9',marker='s')
plt.legend(['Salary', 'Expenses'])
plt.xlabel("Time")
plt.ylabel("Dollars $")
plt.title('Salary and Expenses Comparison')
plt.show()
```

Thank you for your time!