PYTHON PROGRAMMING AND MACHINE LEARNING

PYTHON LANGUAGE FOR C# AND JAVA DEVELOPERS

Yunghans Irawan (yirawan@nus.edu.sg)

Objectives

- Learn Python language
- At the end of the course, students will
  - Able to read and write programs in Python

Prerequisites

- Programming Knowledge
  - You are assumed to master the basic programming concepts and object oriented programming in C#, Java or other languages

Agenda

- Python vs C#/Java
- Python Ecosystem
  - Machine Setup for Python Development
- Variables
- Functions
- Classes and Methods
- Conditionals
- Loops
- List
- Map
- Tuple

Python vs. C#/Java

Python	C# and Java
High Level Language	High Level Language
Interpreted Source code is run by an interpreter in an interactive mode (line by line) or script mode (a file/project as a whole) There are other variant which compile Python into executable	**Compiled** Source code is converted into binary (bytecode in Java and CIL in C#) There are variants of Java/C# which is interpreted
Dynamic Type The type of a variable can change depending on the type of value stored in the variable	**Static Type** Variable type is fixed and will be checked by the compiler
Strict Indentation Indentation is part of the code. Wrong indentation leads to errors.	**Ignore Indentation**

Running Python Code

- Interactively
  - Run "Python" from command prompt
  - Type your code
- Script mode
  - Write your code in a .py file
  - Run "python file.py"
- Jupyter Notebook
  - Notebook is interactive document which contains texts and codes
  - Allow you to mix your notes, codes and the result of your code execution and share it with others

Environment

- Local machine
  - To run python on local machine, you need to install Python and the required packages
  - A popular Python distribution is Anaconda which makes installation of Python packages a lot easier
- Cloud
  - There are a number of online services which allow us to run a Jupyter notebook in the cloud
  - One of my favorite is Google Collaboratory. Just put the Notebook in your Google Drive and open it from Google Collaboratory.

Variables

- Python is a dynamically-typed language which means that we do not know the type of a variable until we run it, unlike C# and Java that we have learned before.
- As the implication, we do not need to specifically declare the type of a variable. A variable will start to exist the moment you use it. A variable can contain values of any type and you can query the type by using type() method.
- Naming convention:
  - CAPITALIZED_WITH_UNDERSCORES for constants
  - lowercase_separated_by_underscores for others

Variables

Code

```
x = 6
print(x)
print(type(x))
```

Output

```
6
<class 'int'>
```

Variables

Code

```
x = "Hello world"
print(x)
print(type(x))
```

Output

```
Hello world
<class 'str'>
```

Functions

- Because Python is a dynamically typed language, we don't have to put the types on function declaration

- When a **function** is a member of a class, it's called a **method**

Functions

Code

```
def say_hello():
    print("Hello world!")

say_hello()
```

Output

```
Hello world!
```

Classes and Methods

- Classes and methods are declared in a similar way as C# and Java
  - Dynamically typed
- All members are public
  - Members starts with double underscore are not meant to be used by other classes
  - Example: _ _ init _ _
- __init__ is a special method that acts like a constructor

Classes and Methods

Code

```
def __init__(self):
    self.data = []
```

Other sample (interactive mode)

```
>>> class Complex:
...     def __init__(self,
realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

Conditionals

Code

```
hour = int(input ("Enter hour in
24-hour format:"))

if (hour < 12):
    print("Good morning")
elif (hour < 18):
    print("Good afternoon")
else:
    print("Good evening")
```

Output

```
Enter hour in 24-hour format:19
Good evening
```

Loops

Code

```
x=1
while x < 10:
    print(x)
    x = x + 1
```

Output

```
1
2
3
...
8
9
```

Loops

Code

```
for x in range(1,10):
    print(x)
```

Output

```
1
2
3
...
8
9
```

List

- List is a data structure like an array that is dynamically resized
- Python has no array, so list usage is common.

List

Code

```
array_1 = [1,2,3,4,5]
print("Array 1 : " , array_1)
print("The length of array 1 is " ,
len(array_1))
array_2 = ["duck", "chicken", 3, 4, True]
#array is untyped
for val in array_2:
    print(val, type(val))
```

Output

```
Array 1 :  [1, 2, 3, 4, 5]
The length of array 1 is  5
duck <class 'str'>
chicken <class 'str'>
3 <class 'int'>
4 <class 'int'>
True <class 'bool'>
```

Tuple

Code

```
tuple_1 = (1,2)
print(tuple_1, type(tuple_1))

# first element of tuple_1
print(tuple_1[0])
```

Output

```
(1, 2) <class 'tuple'>
1
1
```

Map

- Map or dictionary is a list of key-value pair with unique keys
- Values can be retrieved given a key
- Known as Map in Java and Dictionary in C#

Map

Code

```
# Let's say we use our dictionary to represent room booking
map_1 = {'r1': 'alice', 'r2':'bob', 'r3': 'charlie', 'r4': None}
print(map_1, type(map_1))

# checking who books room 2
print(map_1['r2'])

# Denise books room 4
map_1['r4'] = 'Denise'
print(map_1)
```

Output

```
{'r1': 'alice', 'r2': 'bob', 'r3': 'charlie', 'r4': None} <class
'dict'>
bob
{'r1': 'alice', 'r2': 'bob', 'r3': 'charlie', 'r4': 'Denise'}
```