PYTHON PROGRAMMING AND MACHINE LEARNING

OBJECT ORIENTED IN PYTHON

Yunghans Irawan (yirawan@nus.edu.sg)

Objectives

- Understand the basic syntax of OO in Python

Define a Class

```
class Dog:
    pass
```

Instance Attributes

```
class Dog:

    # Initializer / Instance Attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

__init__ is the constructor method. We don't need to call it directly. self is always the first argument of the method and refer to the instance itself

We create an instance of Dog by calling

```
a_dog = Dog('Blackie',1)
```

Class Attributes

```
class Dog:

    # Class Attribute
    category = 'mammal'

    # Initializer / Instance Attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

A dog is always a mammal regardless of the instance

Using the class

```
philo = Dog("Philo", 5)
mikey = Dog("Mikey", 6)

print("{} is {} and {} is {}.".format(
    philo.name, philo.age, mikey.name, mikey.age))

if philo.category == "mammal":
    print("{0} is a {1}!".format(philo.name, philo.category))
```

```
Philo is 5 and Mikey is 6.
Philo is a mammal!
```

A dog is always a mammal regardless of the instance

Instance Method

```
class Dog:

    # Class Attribute
    category = 'mammal'

    # Initializer / Instance Attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def description(self):
        return "{} is {} years old".format(self.name, self.age)

    # instance method
    def speak(self, sound):
        return "{} says {}".format(self.name, sound)
```

The first argument of all the instance method is the self argument

Inheritance

```
# Child class (inherits from Dog class)
class RussellTerrier(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)


# Child class (inherits from Dog class)
class Bulldog(Dog):
    def run(self, speed):
        return "{} runs {}".format(self.name, speed)
```