NUS AI SUMMER EXPERIENCE 2019

Object Oriented Programming in Python

PAN BINBIN

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

Introduction

- This lecture discusses Object Oriented Programming
  - Better program design
  - Better modularization

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

What is an Object?

- A software item that contains variables and methods
- Object Oriented Design focuses on
  - **Encapsulation**:
    - dividing the code into a public interface, and a private implementation of that interface
  - **Polymorphism**:
    - the ability to overload standard operators so that they have appropriate behavior based on their context
  - **Inheritance**:
    - the ability to create subclasses that contain specializations of their parents

Definition

- Class - a template - Dog
- Method or Message - A defined capability of a class - bark()
- Field or attribute- A bit of data in a class - length
- Object or Instance - A particular instance of a class - Lassie

Namespaces

- At the simplest level, classes are simply namespaces

- It can sometimes be useful to put groups of functions in their own namespace to differentiate these functions from other similarly named ones.

```
import math
class myclass:
    def exp(x):
        return 0
math.exp(1)
```

```
2.718281828459045
```

```
myclass.exp(1)
```

```
0
```


Python Classes

- Python contains classes that define objects
  - Objects are instances of classes

__init__ is the default constructor

```
class atom:
    def __init__(self,atno,x,y,z):
        self.atno = atno
        self.position = (x,y,z)
```

self refers to the object itself, like *this* in Java.

Example

- Person class
  - Overloaded the default constructor
  - Defined class variables (name, age) that are persistent and local to the person object
- Encapsulation
  - instead of passing long lists of arguments, encapsulate some of this data into an object, and pass the object.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```


Class Method

- Objects can also contain methods.
- self must be the first parameter to any object method
- The **self** parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.
- must access the object's fields through the self reference

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

```
Hello my name is John
```

Object Properties

- You can modify properties on objects

```
p1.age = 40
```

- You can delete properties on objects

```
del p1.age
```

- You can delete objects

```
del p1
```


Inheritance

- Inheritance allows us to define a class that inherits all the methods and properties from another class.
- **Parent class** is the class being inherited from, also called base class.
- **Child class** is the class that inherits from another class, also called derived class.

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def welcome(self):
        print("Welcome", self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:

x = Person("John", "Doe")
x.welcome()

class Student(Person):
    def __init__(self, fname, lname, year):
        Person.__init__(self, fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

Overriding of method


```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def welcome(self):
        print("Welcome", self.firstname, self.lastname)
```

```
class Student(Person):
    def __init__(self, fname, lname, year):
        Person.__init__(self, fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)
```

PAN BINBIN

BINBIN.PAN@NUS.EDU.SG

NUS AI SUMMER EXPERIENCE 2019

Object Oriented Programming in Python

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM