OBJECT ORIENTED PROGRAMMING

Object Oriented Approach in Python

Zuhaib A. Shaikh,
Asst. Prof., CSE Deptt.,QUEST
Web: zuhaib-shaikh.neocities.org

Classes and Objects

- Class is defined globally in program, must be inherited from a parent class
- Classes are arranged in Hierarchical form, from abstract parent to child classes
- Highest Parent class in python is called “object”, considered by default
- While its objects can be created locally or globally.
- Syntax

```
class class_name(Parent_class_optional)
        members
```

- All members are accessed with bind object keyword, called self
- The access specifiers in the class provides data hiding
  - Public: All members are considered as public, e.g. `def display(self):`
  - Private: The members that starts with double underscore, e.g. `def __display(self):`
  - Protected: The members that starts with single underscore, e.g. `def _display(self):`
- All methods have at-least one argument i.e. self, a method can also have other attributes too.

Reserved functions

- Many special functions of a class have reserved keywords
- Constructor:

```
def __init__(self, other_arguments_optional):
```

- Destructor:

```
def __del__(self):
```

- String: To display contents/msg when object is printed

```
def __str__(self):
```

- Representation: To display information about object

```
def __repr__(self):
```

- Next: To work with next function

```
def __next__(self):
```

- Iteration: To use object in iterative structure

```
def __iter__(self):
```

- Iteration can be defined after next.
- Iteration variable can be declared in next function

Polymorphism

- Overloading:
  - Method overloading is not possible in python, the same can be achieved by using default arguments
  - e.g.

```
def __init__(self, val=None):
```

  - Operator overloading can be achieved by using pre-defined function names:

+	__add__(self, other)	Addition
*	__mul__(self, other)	Multiplication
-	__sub__(self, other)	Subtraction
//	__floordiv__(self, other)	Floor Division
**	__pow__(self, other)	Power
%	__mod__(self, other)	Remainder
/	__truediv__(self, other)	Division
<	__lt__(self, other)	Less than
<=	__le__(self, other)	Less than or equal to
==	__eq__(self, other)	Equal to
!=	__ne__(self, other)	Not equal to

Polymorphism

- Operator overloading predefined functions

>	__gt__(self, other)	Greater than
>=	__ge__(self, other)	Greater than or equal to
<<	__lshift__(self, other)	Bit-wise left shift
>>	__rshift__(self, other)	Bit-wise right shift
&	__and__(self, other)	Bit-wise AND
|	__or__(self, other)	Bit-wise OR
^	__xor__(self, other)	Bit-wise XOR
~	__invert__(self)	Bit-wise Not
[index]	__getitem__(self, index)	Index operator
in	__contains__(self, value)	Check membership
len	__len__(self)	The number of elements

- Overriding:
  - Overriding can be achieved in same manner

Inheritance

- Inheritance can be achieved by mentioning parent class name

```
class Parent_class_name(object_optional):
        members
class Child_class_name(Parent_class_name):
        members
```

- Multiple and Multilevel inheritance can be achieved similarly by separating comma
- Parent class methods can be called from child class using name, e.g.

```
Parent_class_name.method()
```

- Static Members:
  - All data members defined without self are considered as static
    - Referred as Class_name.Static_variable
    - But can be can modified separately with object (a local variable instread of static is created)
    - It is better to access it with static member.
  - A method of a class can be made static by writing function property with "*decorator*" before defining function

```
@staticmethod
def function(arguments_optional):
```

Abstract Classes

- No virtual functions in Python
- Abstract class can be created using:
  - abc (abstract base class) package
  - Adding abstract method in class (method is declared without implementation)

```
from abc import ABC, abstractmethod
class Abstract_class_name(ABC):
        members
    @abstractmethod
        def method_name(self):
            pass                    # or any other set of instructions
class Child_class_name(Abstract_class_name):
        members
```

  - Child class should overload abstract method of parent class
  - Besides, child class can have its own methods

Questions