Classes and Objects

→ Class is a user defined data type that binds data & functions together into a single entity.
→ It is a 'blue print' that doesn't consume any memory & doesn't have existence
→ class has its own variables / attributes / member variables and member functions / methods

→ Object is an instance of class with physical existence.
→ One can create any number of objects for a particular class

General OOP concepts

1. C
   - → successfully implemented structural / procedural methodology
   - → Solution for this is top down approach
   - → process oriented model, code acting on the data
2. Python, C++, Java
   - → object oriented programming
   - → bottom up approach
   - → Has many advantages like reusability of code, security
   - → Its data - centered programming methodology thought as data controlling access to code.

→ member function is the only means of accessing an object's data.

Elements of OOP :

1. class : Its a user defined data type that binds data and functions together into a single entity
   : It has properties or data members and behaviours or member functions

2. Object : An object is an instance of class
   : Object is a physical entity consumes energy
   : Every object has unique existence and is different from other objects of the same class

3. Encapsulation : The process of binding code and data together into a single entity is called encapsulation
   : The member variables and functions of the class can be private or public.
   - → private → only other members of the same class can access it
   - → public → It can be accessed by other classes also.

   : This security is provided by encapsulation

4. Data Abstraction : Hiding the implementation details from the end user.
   : Abstraction is a way of designing the program in such a way that the end user neednot know the internal details of the function

5. Inheritance : Making use of existing code
   : the concept is of hierarchical / top down approach
   : Its a process by which one object can acquire the properties of another object.

: Is - a relationship.
: top - to - Down ⇒ Generalization to Specialization
: bottom - to - top ⇒ Specialization to Generalization

6. Polymorphism : This can be thought of as one interface multiple methods.
: Using this method function overloading and operator overloading can be done
: It's a more economical way of expressing different type of functions

1/5/21 : 1. Programmer defined types

→ class keyword is used to create a class of a particular object type

→ An object is an instance of a class

→ The process of creating a new object is called instantiation

→ When object is printed python tells which class it belongs to and where it is stored in the memory.

→ The address of object is a hexadecimal number. i.e object has physical existence Therefore location but class doesn't

a. Attributes

→ objects contains named elements called attributes

→ attributes are assigned using dot operators with the object.

ex: p.x ⇒ mean go to object P and get the value of x

→ There won't be name conflict between a variable of a normal program and an object's attribute.

a) docstring

→ described using """ . . . """, 3 consecutive double quotes

→ It tells the user what the class actually is.

→ print (Point. - - doc - -) →> gives docstring of the class
→ print (P1. - - doc - -) →> gives docstring of the class of the object

b) Types of attributes for classes in python

1) Class attribute → defined in the class usually immediately after class header
→ Common to all objects of the class i.e they are shared by all objects created from that class

ii) Instance attribute → defined for individual object and are available only for that object
→ it can't be used by other objects

3. Rectangles

→ to illustrate it is possible to make an object of one class as an attribute to another class.

→ objects are mutable

21/5/21

4) Copying

→ An object is aliased when an object is assigned to another object of the same class.

i) direct assignment
ii) when object is passed as argument to function
iii) when object is returned from the function.

Shallow copy { module → copy, method → copy } used to duplicate / copy objects and contents without aliasing.

deep copy
→ Copy ensures that contents of object is object is copied but the memory space / location will be different
→ It behaves strangely if an object is copied onto another object
⇓
attributes of objects will be aliased
to avoid this we use
⇓
method deepcopy → from module copy

5) Debugging

i) attribute Error → If we try to access an object that is not there
↳ to avoid this we can use try and Except

ii) type() → used to check which class the object belongs to

iii) isinstance() → used to check if an instance is of a particular class

iv) hasattr() → can be used to check if an object has a particular attribute or not.

6) Classes and Functions

Programming → Prototype and patch
→ developing a complex program starts with prototype and

incrementally dealing with complication i.e patch or

Prototyping vs Planning

i) Write a prototype & then patch it with new code as and when new info becomes available.
↳ Unreliable & creates unnecessary code

ii) Designed development → have full insight into the programme before coding it.

Functional programming → Functions

i) Pure functions : Takes objects as arguments and work with them without modifying original arguments

ii) Modifiers : Modify the original arguments

a) debugging
→ Write error - checking code to detect wrong inputs

→ 'assert' can be used
↳ if assert is true, then following statements will be evaluated

else Assertion error is raised.

7) Classes and Method
→ classes contain class level attributes, instance level attributes and methods.

a) Object oriented features
→ method is a function associated with a particular class i.e function as member of a class.
→ syntax for invoking a method is different from that of calling a function.
→ Methods are defined inside class definition to make their relationship explicit.

i) Program have classes and method
ii) computation is operation on objects
iii) Objects exist in real world & method is their interaction

b) the __init__() method (Special method)
→ it indicates initialization
→ Its invoked automatically when object of a class is created.

→ Every method of any argument has the first argument as self.
→ The argument self is a reference to the current class

→ Object that invokes the method is a subject

c) __str__() method (special method)

→ its a special method used for string representation of user defined objects.

→ Using __str__() is a polymorphic method i.e when print is used on user defined objects __str__() is called automatically

→ __str__() will return the string format of what we have given inside it and that string is printed by print() method.

e) Operator overloading → Polymorphic nature of Object oriented programming

→ Ability of an existing operator to work on user defined data type (class) is known as operator overloading

operator	special method	operator	special method
+	__add__()	<=	__le__()
-	__sub__()	>=	__ge__()
*	__mul__()	==	__eq__()
/	__truediv__()	!=	__ne__()
%	__mod__()	in	__contain__()
<	__lt__()	len	__len__()
>	__gt__()	str	__str__()

type based dispatch : Based on type of argument received, appropriate action is taken

8) Debugging

Vars() → map the attribute names and values as dictonary

print_attributes() transverses this dictonary

getattr(obj, attr) — takes obj & attr as string & returns attr value.