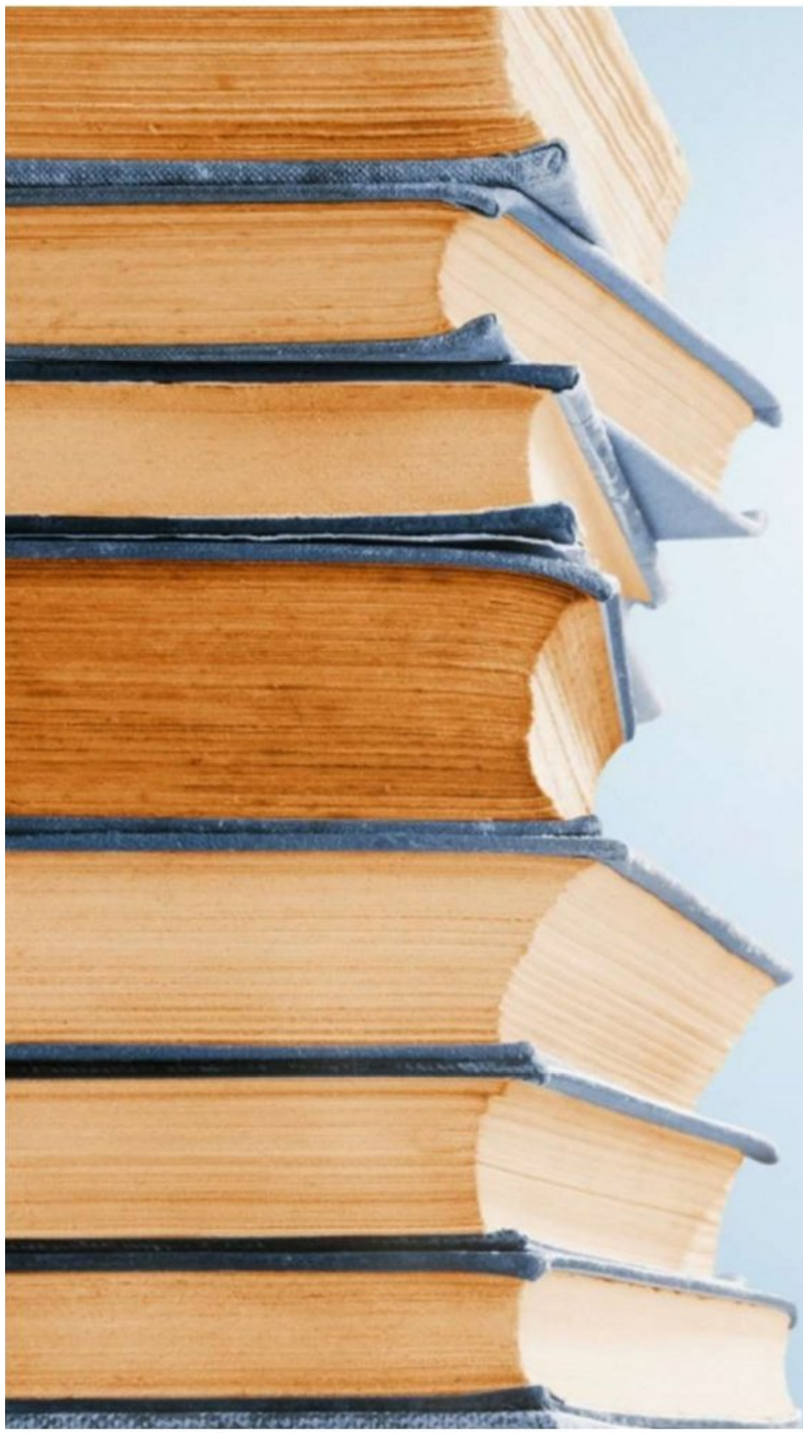
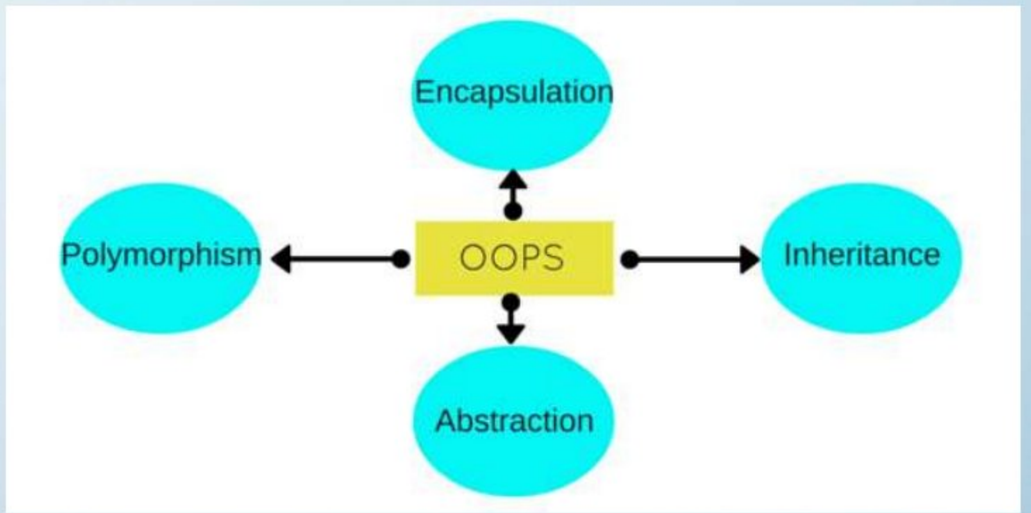


OOPs in Python

Object Oriented programming is a programming style that is associated with the concept of **Class**, **Objects** and various other concepts revolving around these two, like **Inheritance**, **Polymorphism**, **Abstraction**, **Encapsulation** etc.



Procedural vs OOPs

POPs	OOPs
In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects**.
In POP, Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**.
POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
POP does not have any proper way for hiding data so it is **less secure**.	OOP provides Data Hiding so provides **more security**.
In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.

Creating a Simple Class

creating_Class.py - D:/Documents/Python/creating_Class.py (3.6.4)

File Edit Format Run Options Window Help

```
class employee:
    def ___init___  ( self, name, mobile)  :
        self.name=name
        self.mobile=mobile
    def display ( self)  :
        print ( "Name: ",self.name," Contact: ",self.mobile)

obj=employee ( "Rajat",9761757762)
obj.display ()
```

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.4  ( v3.6.4:d48eceb, Dec 19 20
Type "copyright", "credits" or "license ()  " f
>>>
=============== RESTART: D:/Documer
Name:  Rajat  Contact:  9761757762
>>>
```

Built-In Class Attribute

creating_Class.py - D:\Documents\Python\creating_Class.py (3.6.4)

File Edit Format Run Options Window Help

```
# Built-In class attributes
class employee:
   'Base class'
   nam=""
   mob=""
   def ___init___  ( self, name, mobile)  :
      self.nam=name
      self.mob=mobile
   def display ( self)  :
      print ( "Name: ",self.nam," Contact: ",self.mob)
#___doc___ : Class documentation string or none, if undefined.
#___dict___ : Dictionary containing the class's namespace.
#___name___ : Class name.
#___module___: Module name in which the class is defined.
print ( "Documentation: ", employee.___doc___)
print ( "Dictonary: ", employee.___dict___)
print ( "Class Name: ", employee.___name___)
print ( "Module: ", employee.___module___)
```

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.4  ( v3.6.4:d48eceb, Dec 19 2017, 06:04:45)  [MSC v.1900 32 bit
Type "copyright", "credits" or "license ()  " for more information.
>>>
============== RESTART: D:\Documents\Python\creating__Class.py ===
Documentation:  Base class
Dictonary:  {'___module___': '___main___', '___doc___': 'Base class', 'na
___': <function employee.___init___ at 0x03E3B858>, 'display': <function er
2BE468>, '___dict___': <attribute '___dict___' of 'employee' objects>, '___v
'___weakref___' of 'employee' objects>}
Class Name:  employee
Module: ___main___
>>>
```

Variable hiding in Class

OOPs_variable_hidding.py - D:/Documents/Python/OOPs_variable_hidding.py (3.6.4)

File Edit Format Run Options Window Help

```
# use double underscore before variable (___varb)
class Employee:
    ___name=""
    def showname ( self,nam) :
        self.___name=nam
        print ( self.___name)
    def display ( self) :
        print ( self.___name)

obj=Employee ()
obj.showname ( 'Rajat')
print ( obj.___name)
```

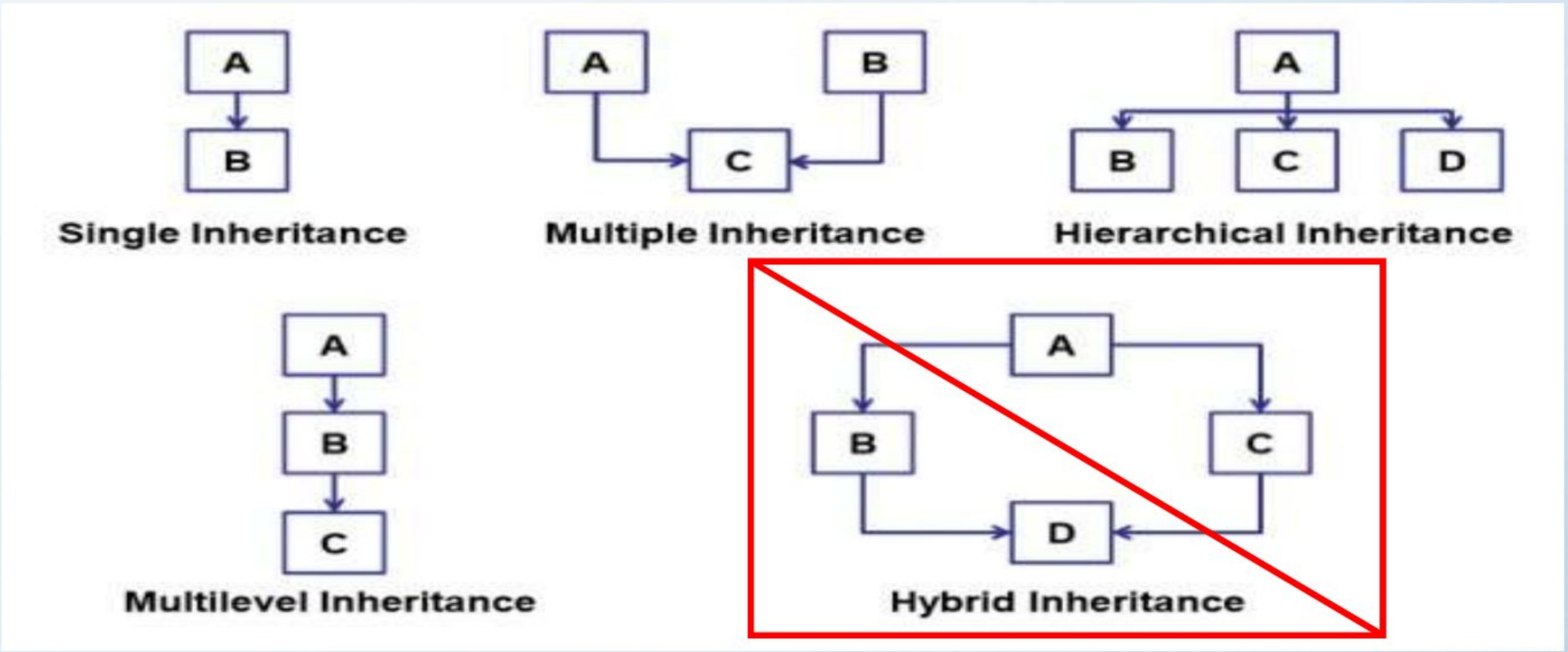
Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.4 ( v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.19(
Type "copyright", "credits" or "license () " for more information.
>>>
========== RESTART: D:/Documents/Python/OOPs__variable__hi
Rajat
Traceback ( most recent call last) :
  File "D:/Documents/Python/OOPs__variable__hidding.py", line 12, in
    print ( obj.___name)
AttributeError: 'Employee' object has no attribute '___name'
>>>
```

What is inheritance?

Inheritance is used to extend of parent class to its child class. Python can support single level, multi level, hierarchal and multiple inheritance.



Inheritance in Python

OOPs_inheritance.py - D:/Documents/Python/OOPs_inheritance.py (3.6.4)

File Edit Format Run Options Window Help

```
# Inheritance
# By default every member is Public.
# We can use single underscore (__var)  for protected
# We can use double underscore (____var)  for private
class employee:
    nam=""
    adrs=""
    def input ( self, n,a)  :
        self.nam=n
        self.adrs=a
class showemployee ( employee)  :
    def display ( self)  :
        print ( "Name: ",self.nam,"\nAddress: ",self.adrs)
obj=showemployee ()
obj.input ( "Rajat","Roorkee")
obj.display ()
# To check subclass
print ( issubclass ( showemployee, employee)  )
print ( issubclass ( employee, showemployee)  )
```

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.4  ( v3.6.4:d48eceb, Dec 19 201
Type "copyright", "credits" or "license ()  " fo
>>>
============== RESTART: D:/Documents
Name:  Rajat
Address:  Roorkee
True
False
>>>
```