Advantages of OOP

1. Modularity
2. Reusability
3. Readability
4. Extensibility
5. Security
6. Efficiency

These are achieved using,

1. Encapsulation
2. Polymorphism
3. Inheritance

Instance method or object level method

A function defined inside class is called method or member function.
A method defined inside class with first argument as “self” is called instance method or object level method.
This method defines behavior of object or operations of object.
This method is bind with object and cannot call or invoked without creating object.

```
list1=list()
list1.append(10)

str1=”python”
str1.upper()
```

Q: What is “self”?

“self” is an argument name or argument, which hold reference of current object to which method is bind (OR) “self” argument hold address of object.

Syntax:

```
def  <method-name>(self,arg1,arg2,…):
        statement-1
        statement-2
        statement-3
```

Example:

```
class Car:
    def start(self):
        print("Car Starts....")
    def stop(self):
        print("Car Stops....")


def main():
    car1=Car()
    car1.start()
    car1.stop()
    car2=Car()
    car2.start()
    car2.stop()

main()
```

Output:

```
========= RESTART: F:/python6pmaug/ooptest1.py ========
Car Starts....
Car Stops....
Car Starts....
Car Stops....
```

When object level method is called or invoked PVM send reference or address of current object, this object address is hold by “self” argument.

Example:

```
class A:
    def m1(self): # object level method
        print(self)



def main():
    obj1=A()
    obj1.m1()
    obj2=A()
    obj2.m1()
    print(obj1)
    print(obj2)

main()
```

Output

```
<__main__.A object at 0x000000C46E463340>
<__main__.A object at 0x000000C46E463D30>
<__main__.A object at 0x000000C46E463340>
<__main__.A object at 0x000000C46E463D30>
```

Properties of object are defined by creating instance variable, these instance variables are created or properties are created in difference ways.

1. After creating object
```
class Employee:
        pass

emp1=Employee()
emp1.empno=101
emp1.ename="naresh"
print(emp1.empno,emp1.ename)
```
2. Using instance methods or object level methods
```
class Employee:
        def set_emp(self):
           self.empno=101
           self.ename="naresh"
        def print_emp(self):
           print(self.empno,self.ename)
emp1=Employee()
emp1.set_emp()
emp1.print-emp()
```

inside the class, object level variables or instance variables are bind with “self”. Object level variables or instance variables define the properties of object.

Example:
```
class Student:
    def set_student(self):
        self.rollno=101
        self.name="naresh"
    def print_student(self):
        print(self.rollno)
        print(self.name)
```

```
def main():
    stud1=Student()
    stud1.set_student()
    stud1.print_student()
    stud2=Student()
    stud2.set_student()
    stud2.print_student()
main()
```

Output:

```
101
naresh
101
Naresh
```

Example:

```
class Employee:
    def set_empno(self,e):
        self.empno=e
    def set_ename(self,en):
        self.ename=en
    def print_emp(self):
        print(self.empno,self.ename)

def main():
    emp1=Employee()
    emp1.set_empno(101)
    emp1.set_ename("naresh")
    emp1.print_emp()
    emp2=Employee()
    emp2.set_empno(102)
    emp2.set_ename("suresh")
    emp2.print_emp()

main()
```

Output:

```
101 naresh
102 suresh
```

3. **Using Constructor method or constructor**

What is constructor?
Constructor is a special method or special instance method. This method is used to initialize object (OR) this method is used to define properties/instance variables/attributes/object level variables of object.
Constructor is a instance method.
The method is executed automatically whenever object of class is created.
Constructor is a magic method.

Syntax:

```
def __init__(self,arg1,arg2,..):
        statement-1
        statement-2
```

Constructor can be defined,

1. With arguments
2. Without arguments

Example:

```
class A:
    def __init__(self):
        print("object is created....")



def main():
    obj1=A()
    obj2=A()

main()
```

Output:

```
========= RESTART: F:/python6pmaug/ooptest5.py ========
object is created....
object is created....
```