Methods

The functions written inside class are called methods.
Methods define functionality or behavior of object.

The methods defined inside class are 3 types
1. Instance method
2. Class method
3. Static method

Instance method
A method defined inside class with first parameter “self” is called instance method.
Instance method defines the behavior of object or functionality of object.
This method is bind with object name cannot call or invoked without creating object.

Syntax:
```
def <method-name>(self,param,param,param,...):
        statement-1
        statement-2
```

What is “self”?
“self” is a parameter name
“self” parameter refer or hold reference of object.

Example:
```
class Car:
    def start(self):
        print("Car Start...")
    def stop(self):
        print("Car Stop")


audi=Car() # Creating object of Car class
volvo=Car() # Creating object of Car class
```

```
list1=list() # creating object of list class
list2=list() # creating object of list class

list1.append(10)
list2.append(20)

audi.start()
audi.stop()

volvo.start()
volvo.stop()
```

Output

```
Car Start...
Car Stop
Car Start...
Car Stop
```

Example:

```
class Robo:
    def walk(self):
        print("Robo Walk....")
    def talk(self):
        print("Robo Talk")
    def sleep(self):
        print("Robo Sleep")


robo1=Robo() # Creating object
robo1.walk()
robo1.talk()
robo1.sleep()
```

Output

```
Robo Walk....
Robo Talk
Robo Sleep
```

These instance methods can be defined,

1. With parameters
2. Without parameters

Without parameters
Method without parameters does not receive any value.

With parameters
Method with parameters receives values

Example:
```
class Robo:
    def talk(self,text):
        print(text)



robo1=Robo()
robo1.talk("Hello Python")
robo1.talk("Python is OOPL")

robo2=Robo()
robo2.talk("Hello Java")
```

Output
```
Hello Python
Python is OOPL
Hello Java
```

Example:
```
class Robo:
    def talk(self,text):
        print(text)
        print(self)
```

```
robo1=Robo()
robo1.talk("Hello Python")
robo1.talk("Python is OOPL")

robo2=Robo()
robo2.talk("Hello Java")
```

Output

```
Hello Python
<__main__.Robo object at 0x0000021DF39FD700>
Python is OOPL
<__main__.Robo object at 0x0000021DF39FD700>
Hello Java
<__main__.Robo object at 0x0000021DF3B3C340>
```

Example:

```
class Matrix:
    def add_matrix(self):
        print("adding matrix")
    def sub_matrix(self):
        print("sub matrix")

class Calculator:
    def add(self):
        print("Add")
    def sub(self):
        print("Sub")
    def multiply(self):
        print("Multiply")
    def div(self):
        print("Div")


matrix1=Matrix()
matrix2=Matrix()

matrix1.add_matrix()
matrix2.sub_matrix()
```

```
calculator1=Calculator()
calculator1.add()
calculator1.sub()
calculator1.multiply()
calculator1.div()
```

Output

```
adding matrix
sub matrix
Add
Sub
Multiply
Div
```

Variables

The variable created inside the class are two types

1. Instance variables
2. Class variables

Instance Variables

Instance variables are object level variables.
These variables are created within object.
Every object is having its own properties and these properties are defined using instance variables.

How many ways instance variables are created?

1. Within class using instance method
2. Outside the class using object name
3. Outside the class using setattr,getattr function
4. Outside the class using dictionary property of object

Outside the class with object name

Outside the class without object name we create instance variables.

Syntax:

<object-name>.<instance-variable-name>=<value>

If instance-variable not exists, PVM create instance variable
If instance-variable exists, PVM access/modify value.

Example:

```
class Employee:
    pass


emp1=Employee()

emp1.empno=101
emp1.ename="naresh"
emp1.job="HR"
emp1.salary=6000

print(emp1.empno)
print(emp1.ename)
print(emp1.job)
print(emp1.salary)

emp1.salary=9000

print(emp1.empno)
print(emp1.ename)
print(emp1.job)
print(emp1.salary)

emp2=Employee()
```

Output

```
101
naresh
HR
6000
```

```
101
naresh
HR
9000
```

Instance variables can be created outside the class using setattr and getattr predefined function.

Example:
```
class Student:
    pass


stud1=Student()
setattr(stud1,"rollno",1)
setattr(stud1,"name","naresh")
setattr(stud1,"course","python")

print(getattr(stud1,"rollno"))
print(getattr(stud1,"name"))
print(getattr(stud1,"course"))

stud2=Student()
```

Output
```
1
naresh
python
```

Outside the class using __dict__ property of object

Example:
```
class Player:
    pass
```

```
p1=Player()
p1.__dict__={'pname':'rohit','runs':50}
print(p1.pname,p1.runs)
print(p1.__dict__['pname'])
print(p1.__dict__['runs'])
```

Output

```
rohit 50
rohit
50
```