Type of Methods

- Instance Methods
  - Accessor Methods
  - Mutator Methods
- Class Methods
- Static Methods

Instance Method

Instance methods are the methods which act upon the instance variables of the class.

Instance method need to know the memory address of the instance which is provided through *self* variable by default as first parameter for the instance method.

Syntax:-

```
def method_name(self):
        function body
```

Instance Method without Parameter/Formal Arguments

```
def method_name(self, f1, f2):
        function body
```

Instance Method with Parameter/Formal Arguments

Instance Method without Parameter

```
class Mobile:

    def show_model(self):
        print("RealMe X")

realme = Mobile( )
```

Instance Method

```
class Mobile:
    def __init__(self):
        self.model = 'RealMe X'

    def show_model(self):
        print(self.model)

realme = Mobile( )
```

Instance variable

Instance Method

Accessing Instance variable
Inside Instance Method

Calling Instance Method w/o Argument

Instance methods are bound to object of the class so we call instance method with object name.

Syntax:- object_name.method_name()

Ex:- realme.show_model()

```
class Mobile:
    def show_model(self):
        print("RealMe X")

realme = Mobile( )
realme.show_model()
```

Calling Instance Method w/o Argument

Instance Method with Parameter

```
class Mobile:
    def __init__(self):
        self.model = 'RealMe X'

    def show_model(self, p):
        self.price = p
        print(self.model, self.price)

realme = Mobile( )
```

Instance variable

Instance Method with parameter

Instance Variable

Parameter

Calling Instance Method with Argument

Syntax:- object_name.method_name(Actual_argument)

Ex:- realme.show_model(1000)

```
class Mobile:
    def __init__(self):
        self.model = 'RealMe X'
    def show_model(self, p):
        self.price = p
        print(self.model, self.price)

realme = Mobile( )
realme.show_model(1000)
```

Calling Method with argument

Accessor Method

This method is used to access or read data of the variables. This method do not modify the data in the variable. This is also called as getter method.

Ex:-

def get_value(self):

def get_result(self):

def get_name(self):

def get_id(self):

```
class Mobile:
    def __init__(self):
        self.model = ‘RealMe X’

    def get_model(self):
        return self.model

realme = Mobile( )
m = realme.get_model()
print(m)
```

Mutator Method

This method is used to access or read and modify data of the variables. This method modify the data in the variable. This is also called as setter method.

Ex:-

```
def set_value(self):
def set_result(self):
def set_name(self):
def set_id(self):
```

```
class Mobile:
    def __init__(self):
        self.model = ‘RealMe X’

    def set_model(self):
        self.model = ‘RealMe 2’

realme = Mobile( )
realme.set_model()
```

```
class Mobile:

    def set_model(self, m):
        self.model = m

realme = Mobile( )
realme.set_model(‘RealMe X’)
```

Class Methods

Class methods are the methods which act upon the class variables or static variable of the class.

Decorator @classmethod need to write above the class method.

By default, the first parameter of class method is cls which refers to the class itself.

Syntax:-

```
@classmethod
def method_name(cls):
        method body

@classmethod
def method_name(cls, f1, f2):
        method body
```

Decorator

Class Method without Parameter/Formal Arguments

Decorator

Class Method with Parameter/Formal Arguments

Class Method without Parameter

```
class Mobile:
    @classmethod
    def show_model(cls):
        print("RealMe X")

realme = Mobile( )
```

Decorator

Class Method

```
class Mobile:
    fp = 'Yes'
    @classmethod
    def show_model(cls):
        print(cls.fp)

realme = Mobile( )
```

Class Variable

Decorator

Class Method

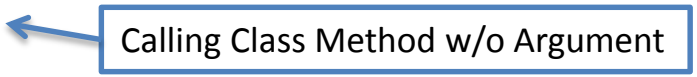
Accessing Class variable
Inside Class Method

Calling Class Method without Argument

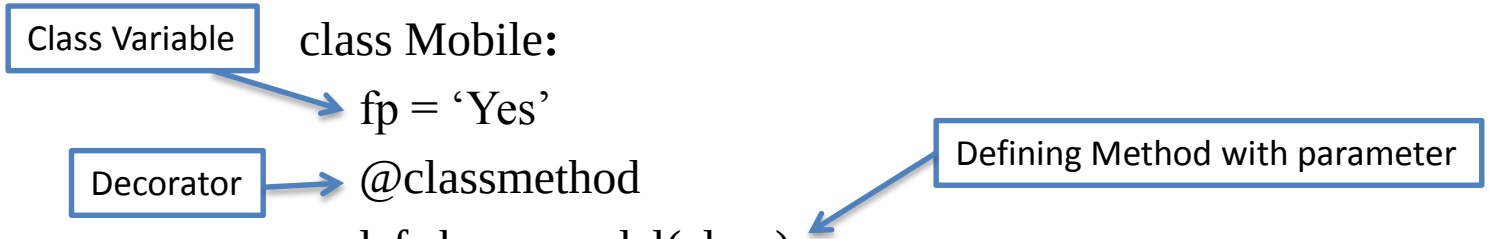
Syntax:- Classname.method_name()

```
class Mobile:
    @classmethod
    def show_model(cls):
            print("RealMe X")

realme = Mobile( )
Mobile.show_model()
```



Class Method with Parameter




```
class Mobile:
    fp = 'Yes'
    @classmethod
    def show_model(cls, r):
        cls.ram = r
        print(cls.fp, cls.ram)

realme = Mobile( )
```

Calling Class Method with Argument

Syntax:- Classname.method_name(Actual_argument)

Ex:- Mobile.show_model(‘4GB’)

```
class Mobile:
    fp = ‘Yes’
    @classmethod
    def show_model(cls, r):
        cls.ram = r
        print(cls.fp, cls.ram)

realme = Mobile( )
Mobile.show_model(101)
```

Calling Method with argument

Static Methods

Static Methods are used when some processing is related to the class but does not need the class or its instances to perform any work.

We use static method when we want to pass some values from outside and perform some action in the method.

Decorator @staticmethod need to write above the static method.

Syntax:-

```
@staticmethod
def method_name():
        method body
```

Decorator

Static Method without Parameter/Formal Arguments

```
@staticmethod
def method_name(f1, f2):
        method body
```

Decorator

Static Method with Parameter/Formal Arguments

Static Method without Parameter

```
class Mobile:
    @staticmethod
    def show_model():
        print("RealMe X")

realme = Mobile( )
```

Decorator

Static Method

```
class Mobile:
    fp = 'Yes'
    @staticmethod
    def show_model():
        print(Mobile.fp)

realme = Mobile( )
```

Static Method

Calling Static Method without Argument

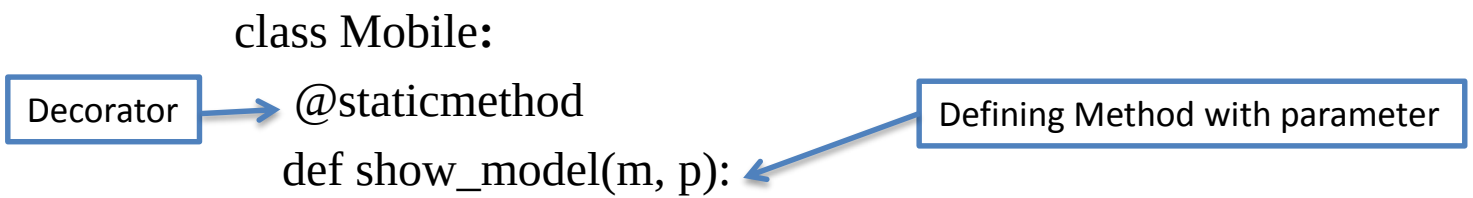
Syntax:- Classname.method_name()

```
class Mobile:
    @staticmethod
    def show_model():
        print("RealMe X")

realme = Mobile( )
Mobile.show_model()
```

Calling Static Method w/o Argument

Static Method with Parameter




```
class Mobile:
    @staticmethod
    def show_model(m, p):
        model = m
        price = p
        print(model, price)
realme = Mobile( )
```

Calling Static Method with Argument

Syntax:- Classname.method_name(Actual_argument)

Ex:- Mobile.show_model(1000)

```
class Mobile:
    @staticmethod
    def show_model(m, p):
        model = m
        price = p
        print(model, price)
realme = Mobile( )
Mobile.show_model('RealMe X', 1000)
```

Calling Method with argument