Python Fundamentals day 49

Today's Agenda

- Types of methods within class
- Instance method
- Static method
- Class method

Types of methods within class

The functions present within the class are called as methods. We can have 3 types of methods within the class

- Instance method
- Static method
- Class method

So far the methods we have come across are instance methods.

Let us take the example of BMW car and modify it according to the necessity and explore the different classes.

Instance methods

This is a very basic and easy method that we use regularly when we create classes in python. If we want to print an instance variable or instance method we must create an object of that required class.

If we are using self as a function parameter or in front of a variable, that is nothing but the calling instance itself.

As we are working with instance variables we use self keyword.

Note: Instance variables are used with instance methods.

```
class BmwCar:

    def __init__(self,name,cc,color):
        self.name=name
        self.cc=cc
        self.color=color

    def start_engine(self):
        print(self.name,'engine is starting')

def main():
    c=BmwCar('bmw',2100,'blue')
    c.start_engine()
    #BmwCar.start_engine(c)

if __name__ == '__main__':
    main()
```

Output:

```
In [16]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
bmw engine is starting
```

Now let us modify the above code by adding function that can convert kilometres to miles.

```python
class BmwCar:

    def __init__(self,name,cc,color):
        self.name=name
        self.cc=cc
        self.color=color

    def start_engine(self):
        print(self.name,'engine is starting')

    def kms_to_miles(kms):
        print(kms*1.6)

def main():
    c=BmwCar('bmw',2100,'blue')
    c.start_engine()
    #BmwCar.start_engine(c)
    BmwCar.kms_to_miles(2)

if __name__ == '__main__':
    main()
```

Output:

```
In [17]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
bmw engine is starting
3.2
```

Great! Using the class it successfully works. But can we do the same using the object? Let us see

```python
class BmwCar:

    def __init__(self,name,cc,color):
        self.name=name
        self.cc=cc
        self.color=color

    def start_engine(self):
        print(self.name,'engine is starting')

    def kms_to_miles(kms):
        print(kms*1.6)

def main():
    c=BmwCar('bmw',2100,'blue')
    c.start_engine()
    #BmwCar.start_engine(c)
    BmwCar.kms_to_miles(2)
    c.kms_to_miles(2)
    #c.kms_tomiles(c,2)

if __name__ == '__main__':
    main()
```

Output:

```
  File "C:/Users/rooman/OneDrive/Desktop/python/test.py",
line 19, in main
    c.kms_to_miles(2)

TypeError: kms_to_miles() takes 1 positional argument but
2 were given
```

We certainly cannot, because internally it is taking only one parameter but two were given. So let us see how to overcome this

Static method

A static method can be called without an object for that class, using the class name directly. If you want to do something extra with a class we use static methods.

```
class BmwCar:

    def __init__(self,name,cc,color):
        self.name=name
        self.cc=cc
        self.color=color

    def start_engine(self):
        print(self.name,'engine is starting')

    @staticmethod
    def kms_to_miles(kms):
        print(kms*1.6)

def main():
    c=BmwCar('bmw',2100,'blue')
    c.start_engine()
    #BmwCar.start_engine(c)
    BmwCar.kms_to_miles(2)
    c.kms_to_miles(2)
    #c.kms_tomiles(2)

if __name__ == '__main__':
    main()
```

Output:

```
In [19]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
bmw engine is starting
3.2
3.2
```

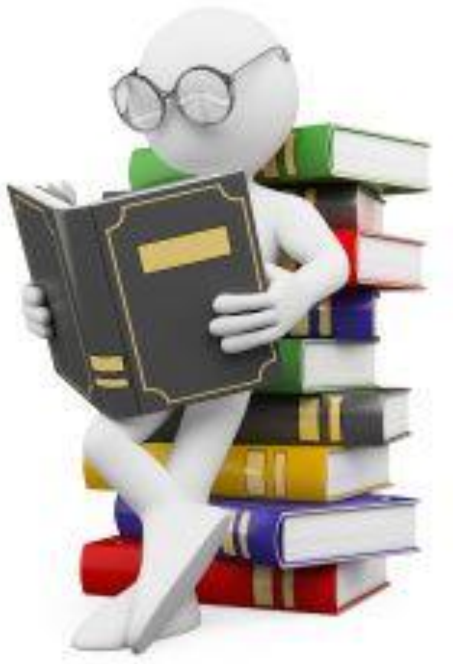
Class method

There are two ways to create class methods in python:

1. Using classmethod(function)
2. Using @classmethod annotation

As we are working with ClassMethod we use the cls keyword. Class variables are used with class methods.

```
class BmwCar:

    def __init__(self,name,cc,color):
        self.name=name
        self.cc=cc
        self.color=color

    @classmethod
    def x1(cls):
        return cls('x1',1998,'blue')

    @classmethod
    def series5(cls):
        return cls('5series',2993,'blue')

    @classmethod
    def i8(cls):
        return cls('i8',1499,'blue')

    def display(self):
        print(self.name)
        print(self.cc)
        print(self.color)

def main():
    c1=BmwCar.x1()
    c2=BmwCar.series5()
    c3=BmwCar.i8()
    c1.display()
    c2.display()
    c3.display()

if __name__ == '__main__':
    main()
```



Output:

```
In [20]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
x1
1998
blue
5series
2993
blue
i8
1499
blue
```