[Hindi] Class Methods As Alternative Constructor | Object Oriented Programming In Python Tutorial #5

Class Methods As Alternative Constructor

In the previous article, we discussed how to work with class methods. In this article, we are going to discuss the class method as an alternative constructor.

```
class Employee:
    def __init__(self,fname,lname,salary):
        self.fname=fname
        self.lname=lname
        self.salary=salary

# Initializing the object
harry = Employee("harry","jackson",4400)
rohan = Employee("rohan","das",4400)


print(harry.fname,rohan.fname)
```

From the above code, it is clear that when we want to add a new object to the class Employee, we simply initialize the new object variable with the conditions such as fname,lname, and salary. Using the object name, we can call any function. Example :

```
harry.increase()
```

In this above example, we have called the **increase** function using the object **harry.** This function, by default, takes the argument "self" which is our object. But in certain conditions, we do not require any object in the function. Thus, we started using the *class method* to avoid code inefficiency. Now the question is how we can use the class method as an alternative constructor.

Starting with the solution:

Creating an object that holds a string. Initializing the object. Let the object name be **lovish.**

```
lovish = Employee.from_str("lovish-jackson-76000")
```

The **employee** is the class, and **from_str** is the method. Now, we will create a class method that will be treated as an alternative class method. The alternative class method will take a string as input and sets all the variables. Here, **from_str** is a new method in the Class.

```
@classmethod
    def from_str(cls,emp_string):
        fname,lname,salary= emp_string.split("-")
        return cls(fname,lname,salary)
```

Let's understand the above code line by line :

a. Here, the method from_str takes the argument *cls,* which is our class *Employee,* and *emp_string* is the string provided.`

b. We are using a split function that returns a list and will be separated followed by the condition "-" and will be stored in the respective variables, i.e., fname, name, and salary. This is parsing the string.

c. After the split, the variables will be populated in the respective variables,i.e.,

```
fname=lovish
lname=jackson
salary=76000k
```

d. At last, we are calling the class constructor or returning cls (cls is our class Employee) with fname, name, and salary.

e. **from_str** function will return an object that will be allocated in **"lovish"**

Getting the output :

```
print(lovish.fname)
print(lovish.lname)
print(lovish.salary)
```

Output :

```
Lovish
Jackson
7600
```

So, this article ends here. If you've enjoyed this article, then please share this article with your friends. In the next article, we are will discuss the static method. Till then, keep coding!

Previous Next