Class Methods as Alternative constructors

In object oriented programming the term “constructor” refers to a specific type if method that is automatically executed when an object is created from a class. The purpose of a constructor is to initialize the object’s attributes, allowing the object to be fully functional and ready to use.

However there are times when you may want to create an object in a different way or with different initial values, then what is provided by the default constructor. This is where class methods can be used as alternative constructors.

A class method belongs to the class to the class rather than to an instance of the class. One common use case for class methods as alternative constructors is when you want to create an object from data that is stored in a different format, such as a string or a dictionary. For example consider a class method “programmer” that has two attribute “name” and “salary”. The default constructor for the class might look like.

```
class programmer:
    def __init__(self,name,salary):
                self.name=name
                self.salary=salary

    @classmethod
    def fromString(cls,string):
        return cls(string.split("-")[0],string.split("-")[1])

p=programmer("Munawar",148)
print(p.name)
print(p.salary)

string="Munawar-2000"
p1=programmer.fromString(string)
print(p1.name)
print(p1.salary)
```

Source Code

```python
class programmer:
    def __init__(self,name,salary):
                self.name=name
                self.salary=salary

    @classmethod
    def fromString(cls,string):
        return cls(string.split("-")[0],string.split("-")[1])

p=programmer("Munawar",148)
print(p.name)
print(p.salary)

string="Munawar-2000"
p1=programmer.fromString(string)
print(p1.name)
print(p1.salary)
```

Thank You