methods

February 11, 2024

[1]:
```
# Let's talk about classes and methods you can define in them.
# We cover three types (yes, there is more):
# instance methods
# class methods
# static methods
```

[2]:
```
# Let's first create a simple class akin to what was used in class:

class User:
    is_admin = False

    def __init__(self):
        self._name = name

# A class with a class attribute is_admin and an instance attribute _name.
```

[3]:
```
# Expand it with an instance method (regular method):

class User:
    is_admin = False

    def __init__(self, name):
        self._name = name

    def get_name(self):
        return self._name

user = User("Simon")
print(user.is_admin)
print(user.get_name())

# Instance methods have access to properties of a particular instance,
# and access to class attributes.
```

```
False
Simon
```

[4]:
```
# Class methods:
class User:
    is_admin = False

    def __init__(self, name):
        self._name = name

    def get_name(self):
        return self._name

    @classmethod
    def change_status(cls):
        if cls.is_admin == False:
            cls.is_admin = True
        else:
            cls.is_admin = False

user1 = User("Simon")
print(user1.is_admin)

User.change_status()

user2 = User("John")

print(user1.is_admin, user2.is_admin)

user1.change_status()
print(user1.is_admin, user2.is_admin)

# Another neat trick: you can define a class method that returns an instance of␣
 ↪that class!
```

```
False
True True
False False
```

[5]:
```
# Static methods: static methods are not easy to motivate.
# they have access to neither instance nor class attributes, so why bother?
# They are functionality you want to attach to your class that does not depend
# on its attributes. Basically, keep it there to organize instead of defining it
# somewhere.

# Let's say we want to be able to create some sort of access engine (class) for␣
 ↪users...

class AccessEngine:
    def __init__(self, credentials):
```

```
        pass

class User:
    is_admin = False

    def __init__(self, name):
        self._name = name

    def get_name(self):
        return self._name

    @classmethod
    def change_status(cls):
        if cls.is_admin == False:
            cls.is_admin = True
        else:
            cls.is_admin = False

    @staticmethod
    def create_engine(credentials):
        engine = AccessEngine(credentials)
        return engine

user1 = User("Simon")
cred = {"username": "simon", "password": "you_will_not_guess"}
acc_eng = user1.create_engine(cred)
```

[ ]: