Access Modifiers, Constructors, and Class Methods in Python

Key Terms

Access Modifier

> In Python, access modifiers are not enforced as strictly as in Java. The convention is to prefix a name with an underscore (_) for protected-like behavior, and use double underscores (__) for private-like behavior.

Constructor

> A constructor in Python is the `__init__` method. It is called automatically when a new instance of a class is created.

Access Modifiers in Python

Python does not have the same access modifiers as Java. Instead, it relies on naming conventions:

- Public: No underscores, e.g., `self.name`
- Protected: One underscore, e.g., `self._name`
- Private: Two underscores, e.g., `self.__name`

Constructors in Python

Default Constructor

A default constructor in Python is simply the `__init__` method without parameters, except for `self`.

```
class Student:
    def __init__(self):
        self.name = None
        self.email = None
        # Other default values
```

Parameterized Constructor

A parameterized constructor in Python is an **init** method with parameters.

```
class Student:
    def __init__(self, name, email):
        self.name = name
        self.email = email
```

Class Methods and Static Methods

In Python, class methods are methods that are bound to the class rather than its object. They can access and modify class state that applies across all instances of the class.

Why use a class method or static method?

- Class methods can access and modify class state.
- Static methods do not access instance or class state. They are utility-type methods that take some parameters and work upon those parameters.

How to create a class method or static method?

```
class Person:
    def __init__(self, name, email):
        self.name = name
        self.email = email

    @classmethod
    def from_string(cls, name_email_str):
        name, email = map(str.strip, name_email_str.split(','))
        return cls(name, email)

    @staticmethod
    def get_person_info(person):
        return f"{person.name} {person.email}"
```