## 🐍Python OOP: Access Modifiers, Getter & Setter (Colorful Notes)

---

### 🐓Access Modifiers in Python

| Type | Syntax | Behavior |
|---|---|---|
| **Public** | `self.name` | Accessible from anywhere. Default in Python. |
| **Protected** | `self._age` | Convention: internal use. Accessible but discouraged outside class. |
| **Private** | `self.__age` | Name mangling: `_ClassName__age`. Harder to access directly. |

**Example:**

```
class Person:
    def __init__(self, name, age):
        self.name = name       # public
        self._age = age       # protected
        self.__salary = 5000  # private

p = Person('Zahid', 24)
print(p.name)        # ✅ works
print(p._age)        # ⚠️ works, not recommended
print(p._Person__salary)  # ✅ access private via name mangling
```

---

### 🐕Classic Getter & Setter (Java-like)

```
class Person:
    def __init__(self, name, age):
        self.__age = age  # private

    def get_age(self):
        return self.__age

    def set_age(self, age):
        if age > 0:
            self.__age = age
        else:
            print('Invalid age')

p = Person('Zahid', 24)
print(p.get_age())  # 24
```

```
p.set_age(30)
print(p.get_age())  # 30
```

---

### 🐗 Pythonic Getter & Setter (`@property`)

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

    # Getter
    @property
    def age(self):
        return self.__age

    # Setter
    @age.setter
    def age(self, value):
        if value > 0:
            self.__age = value
        else:
            print('Invalid age')

p = Person('Zahid', 24)
print(p.age)   # ✅ getter (no ())
p.age = 30     # ✅ setter (no ())
print(p.age)
p.age = -5     # Invalid age
```

**Key Difference:** - **No parentheses** when using `@property`. - Looks like normal attribute access, but method logic runs internally.

---

### 🐚 Quick Comparison: Java vs Pythonic

| Feature | Java | Python (`@property`) |
|---|---|---|
| Access | `getAge()`, `setAge()` | `person.age`, `person.age = 30` |
| Private | `private int age;` | `self.__age` |
| Method call | Required `()` | Not needed |
| Readability | Less Pythonic | More Pythonic & clean |

---

⬜ **Tip:** Python trusts developers ("consenting adults" principle). Private/protected are **conventions**, not strict enforcement.