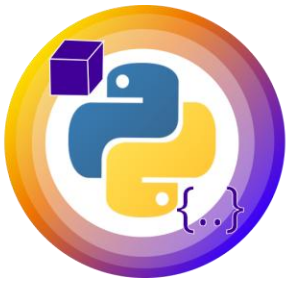
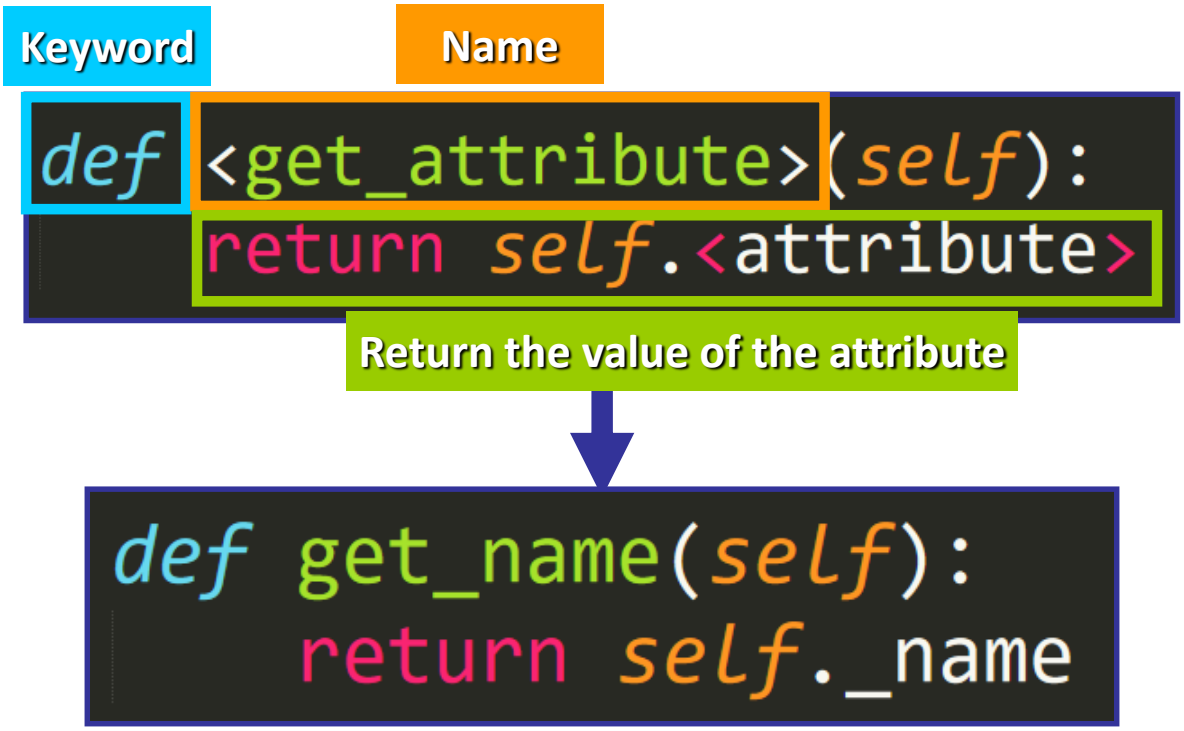


Python OOP: Getters, Setters, & Properties

Key Takeaways

- **Getters:**
  - Methods that instances can call to "**get**" the value of a protected instance attribute.
  - They serve as intermediaries to avoid accessing the data directly.
  - Naming Rules:
    - **get + _ + <attribute>**
    - Examples: get_age, get_name, get_code



Setters

Key Takeaways

- **Setters:**
  - Methods that instances can call to "**set**" the value of a protected instance attribute.
  - They serve as intermediaries to avoid accessing the data directly.
  - You can check if the value is valid before assigning it and you can react appropriately if the value is not valid.
  - They take one argument: the new value for the attribute.
  - Naming Rules:
    - **set + _ + <attribute>**
    - Examples: set_age, set_name, set_code

Keyword Name Parameter

```
def set_name(self, name):
    if isinstance(name, str):
        self._name = name
    else:
        print("Please enter a valid name")
```

Update the value

Key Takeaways

- **Properties:**
  - They are the “pythonic” way of working with getters and setters.
  - The property can be accessed with the same syntax used to access public instance attributes.
  - No need to call getters and setters explicitly, but they do act as intermediaries “behind the scenes”.
  - Two alternatives:
    - Using the built-in function **property()**.
    - Using the **@property** decorator.
  - @property is the recommended syntax to work with properties in Python.
    - Advantages:
      - ✓ More compact.
      - ✓ Improved readability.
      - ✓ No namespace pollution.

Properties

Key Takeaways

- **Using property():**

```
class Patient:

    def __init__(self, name, age, id_num, num_children=0):
        self.name = name
        self.age = age
        self._id_num = id_num
        self._num_children = num_children

    def get_id_num(self):
        print("Getter")
        return self._id_num

    def set_id_num(self, new_id):
        print("Setter")
        if isinstance(new_id, str):
            self._id_num = new_id
        else:
            print("Please enter a valid id")

    id_num = property(get_id_num, set_id_num)
```

Getter **Setter**

```
patient = Patient("Gino", 15, "4535")

patient.id_num # Calls getter

patient.id_num = "545435" # Calls setter
```

Properties

Key Takeaways

- **Using @property:**

```
class House:

    def __init__(self, price):
        self._price = price

    @property
    def price(self):
        return self._price

    @price.setter
    def price(self, new_price):
        if price > 0:
            self._price = _price
        else:
            print("Please enter a valid price")
```

Getter

Setter

```
house = House(50000)

house.price # Calls getter

house.price = 60000 # Call setter
```