Lecture

@property

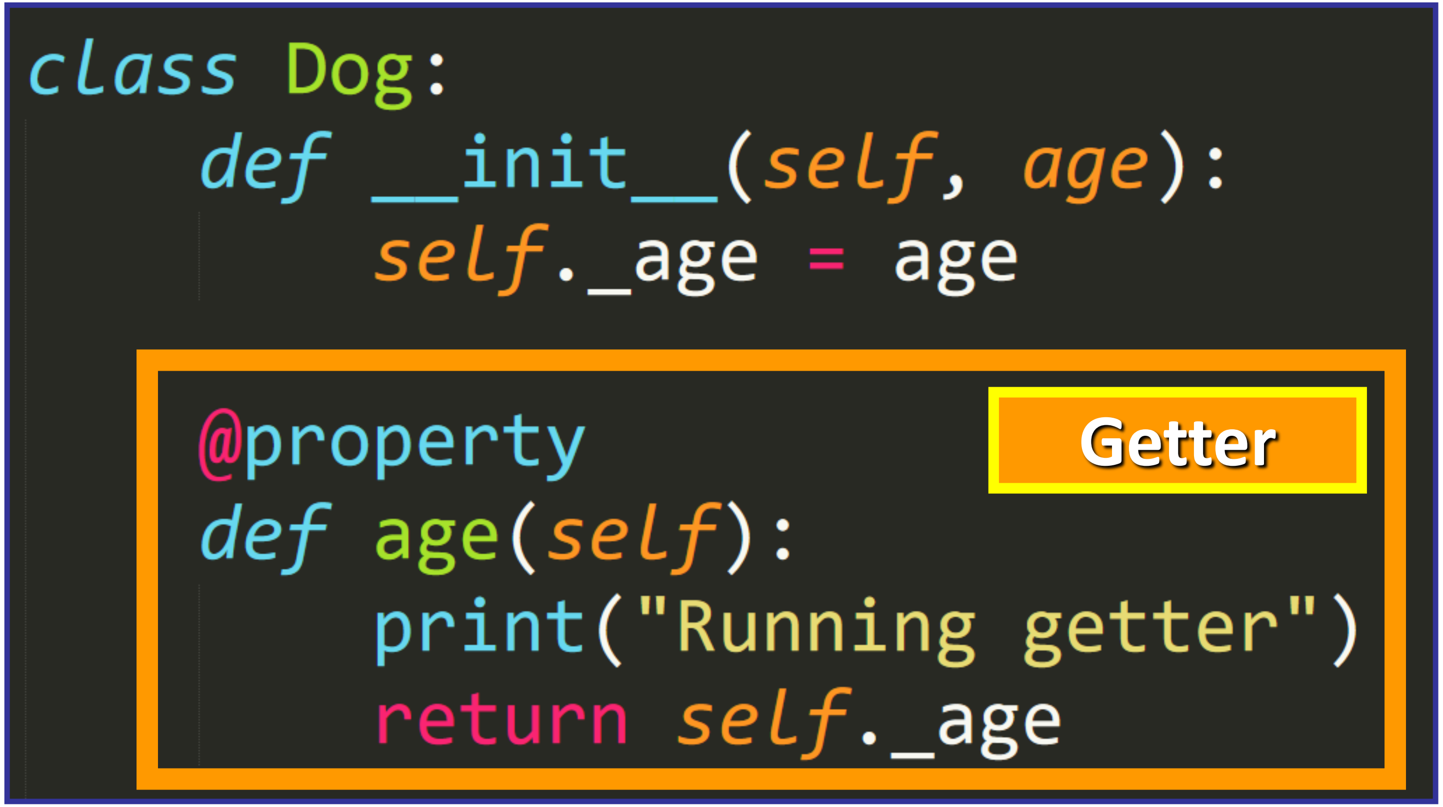
The @property decorator

- More compact
- More readable
- Avoid calling property() directly
- Avoid namespace pollution:
    - No get_<attr>
    - No set_<attr>
    - Reuse the name of the property

The @property decorator

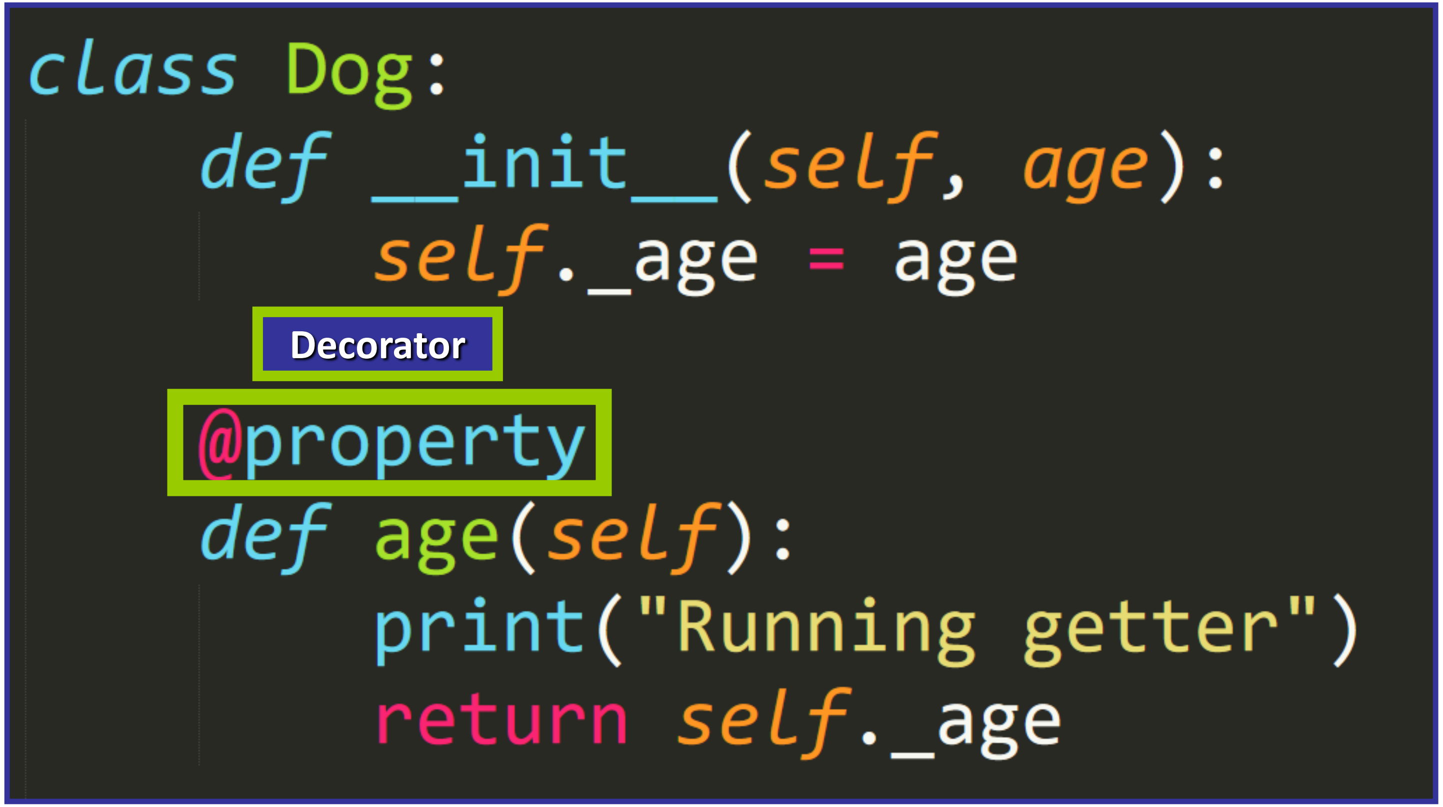
```python
class Dog:
    def __init__(self, age):
        self._age = age
```

The @property decorator



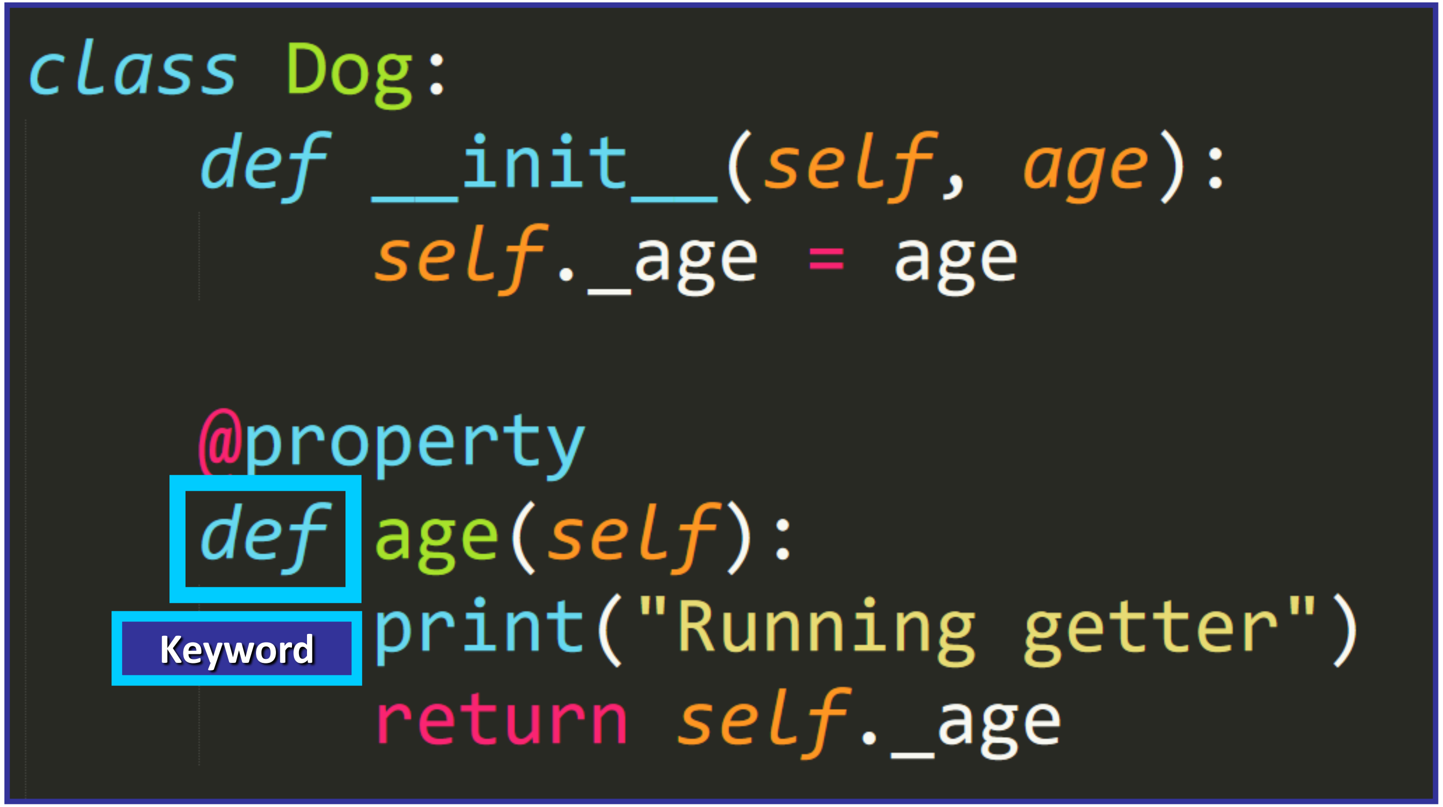
```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



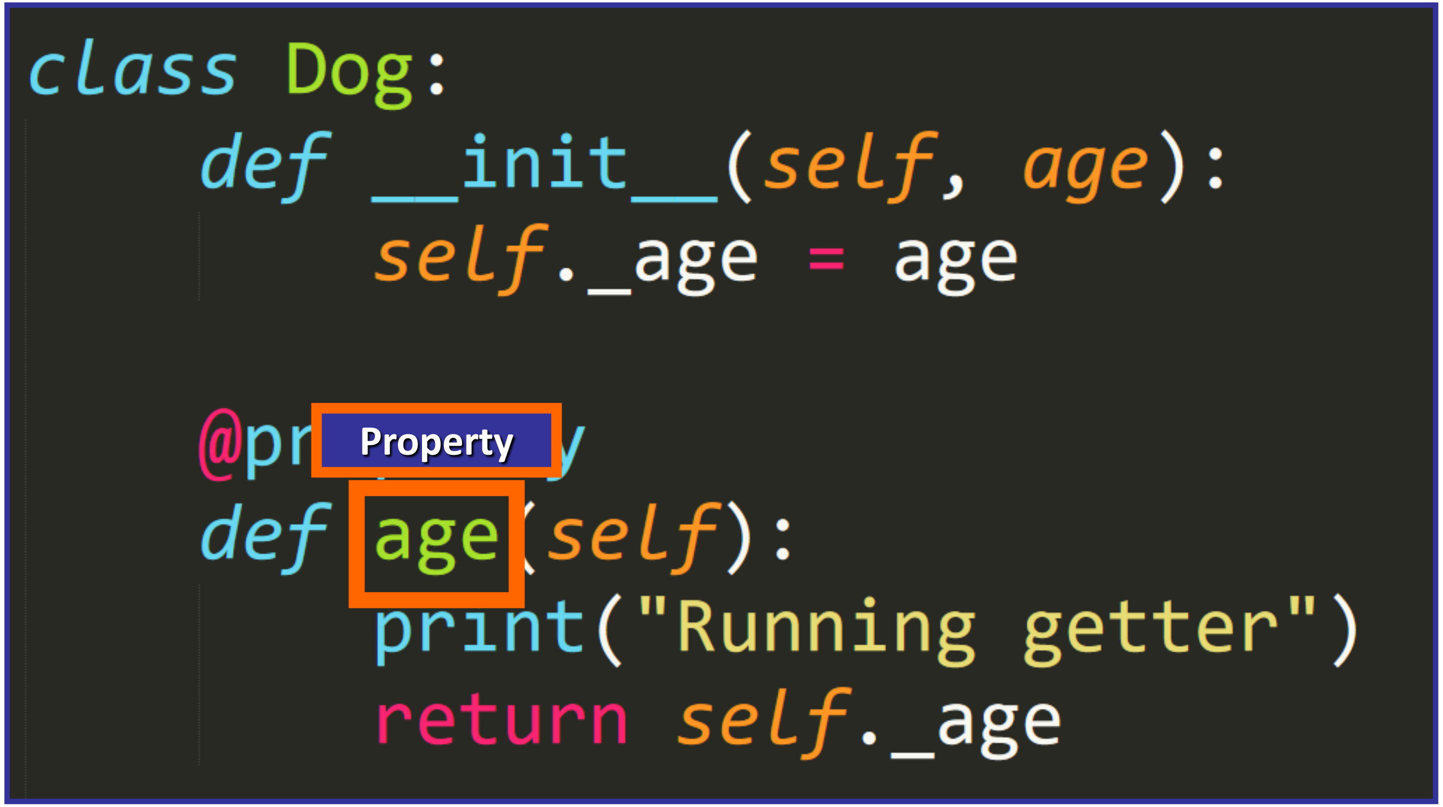
```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



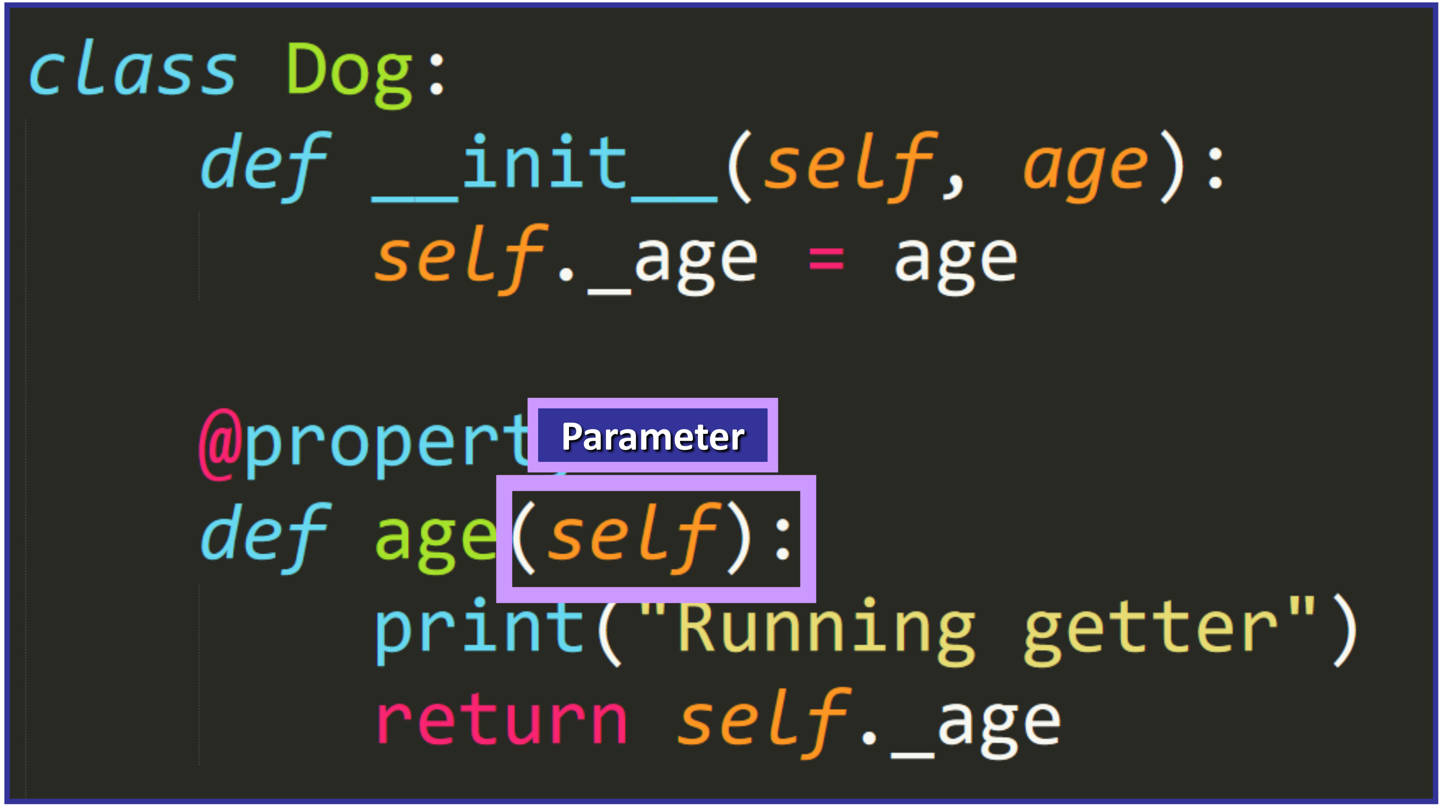
```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



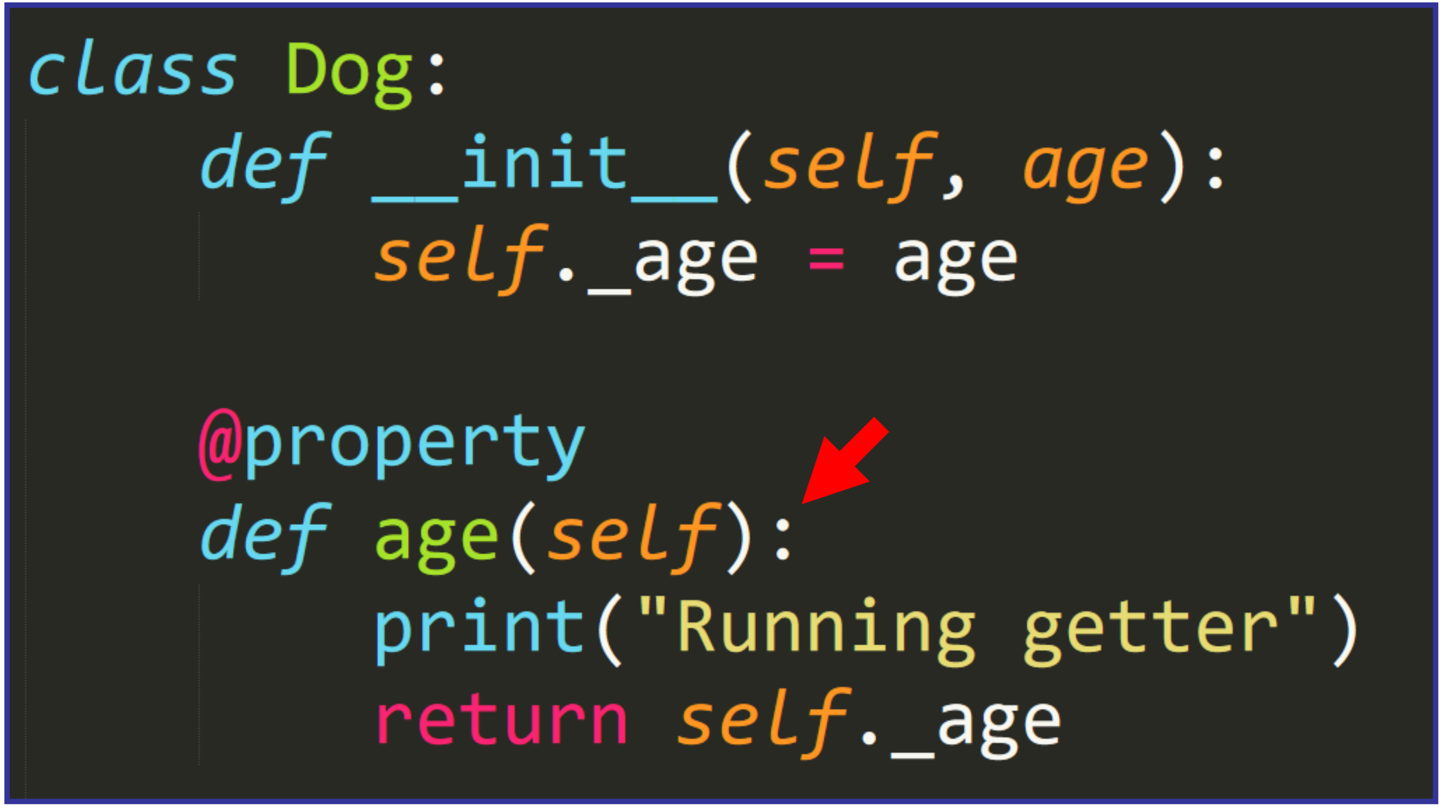
```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



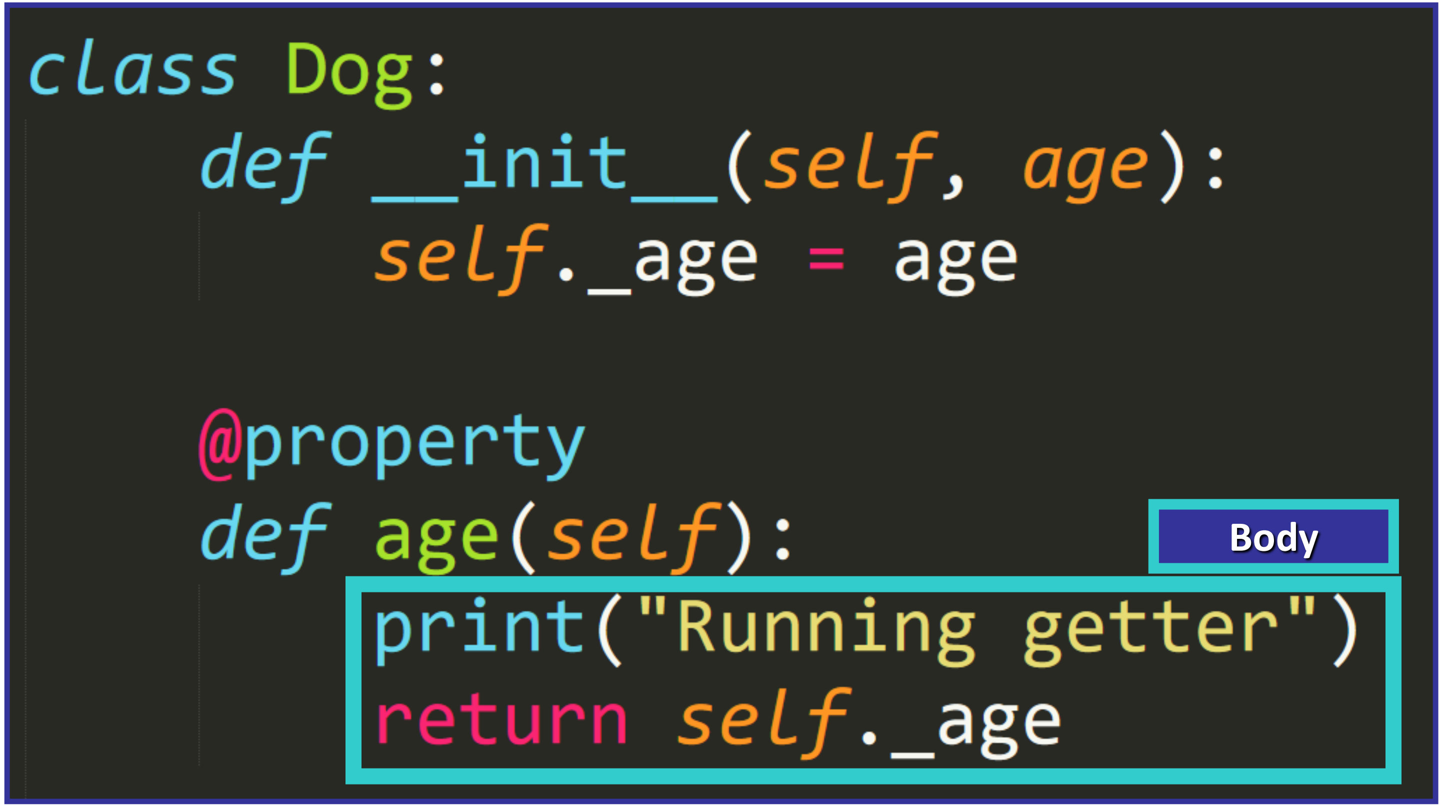
```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



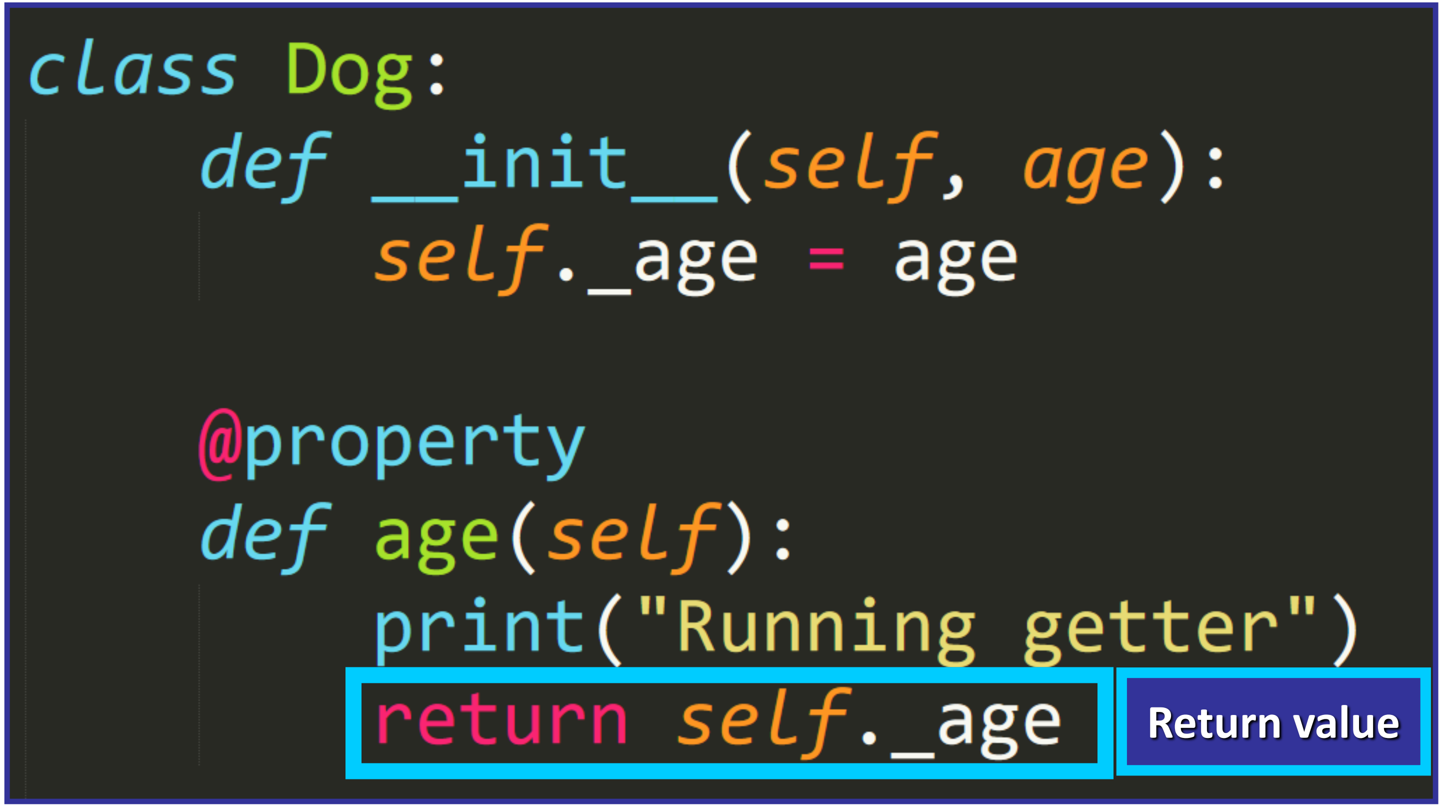
```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator



```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

>>> dog1 = Dog(15)
>>> dog1.age
Running getter
15
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

>>> dog1 = Dog(15)
>>> dog1.age
Running getter
15
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age
```

Mention that you copy/paste to idle, indentation

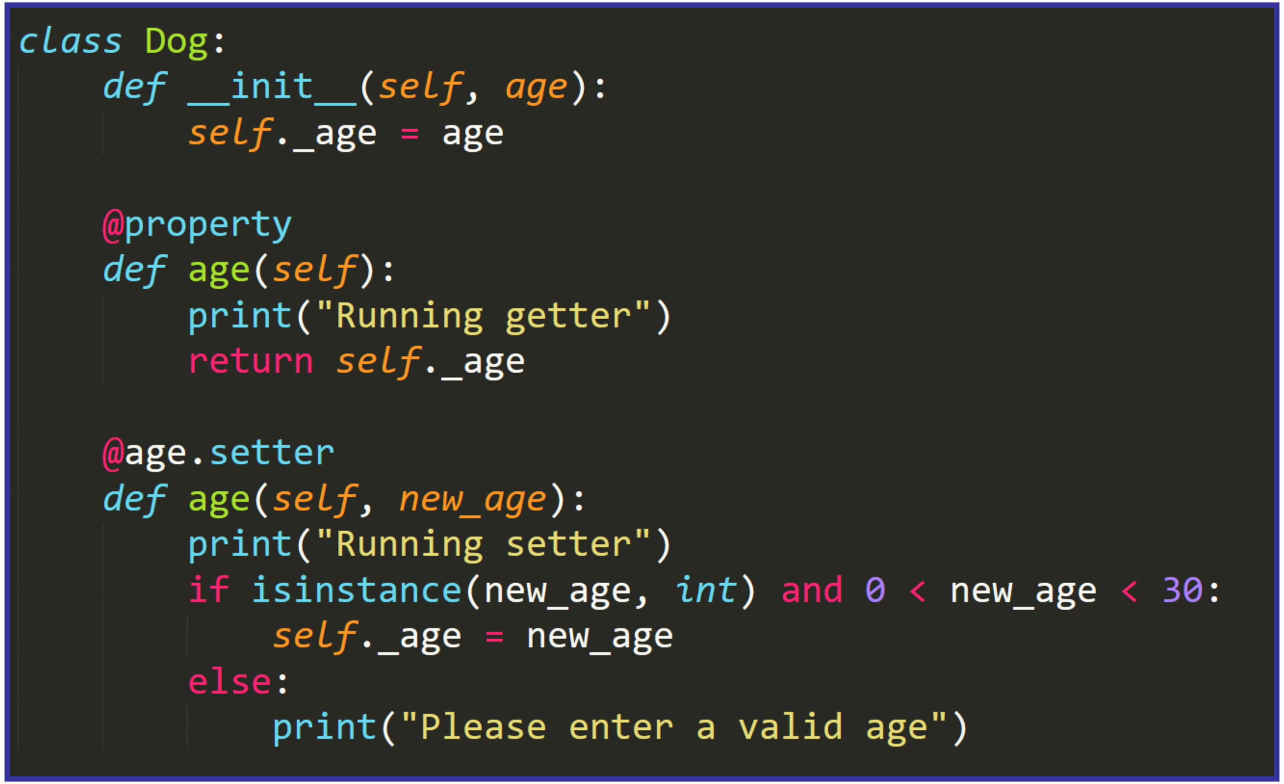
```
    @property
    def age(self):
        print("Running getter")
        return self._age

>>> dog1 = Dog(15)
>>> dog1.age
Running getter
15
```

The @property decorator

Getter ✓

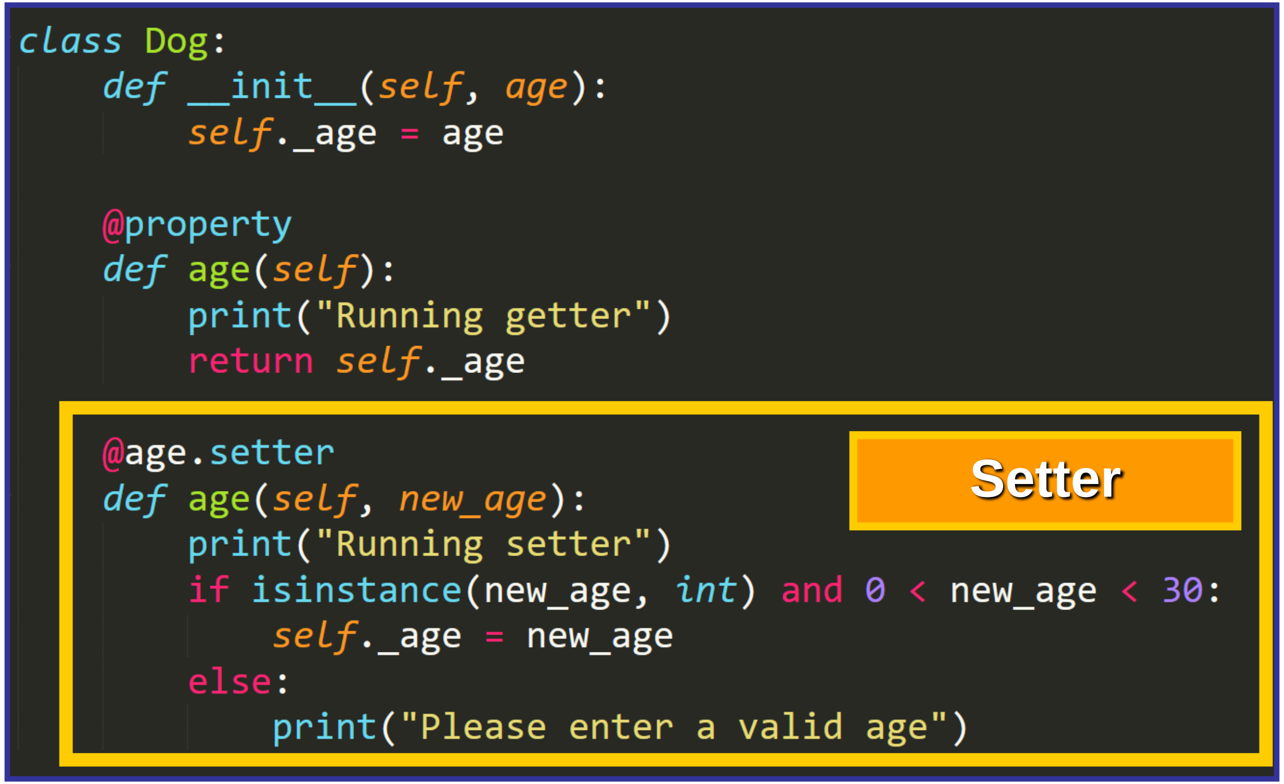
Setter ?

The @property decorator



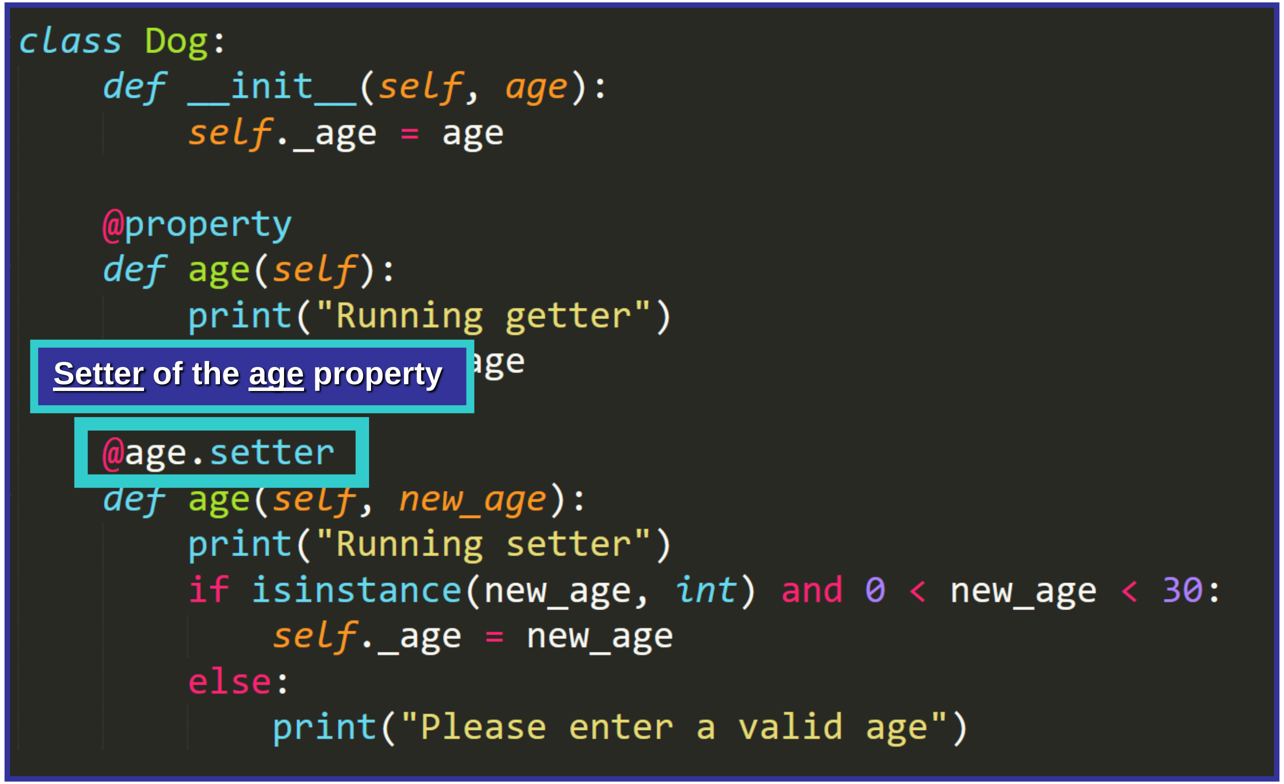
```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator



```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

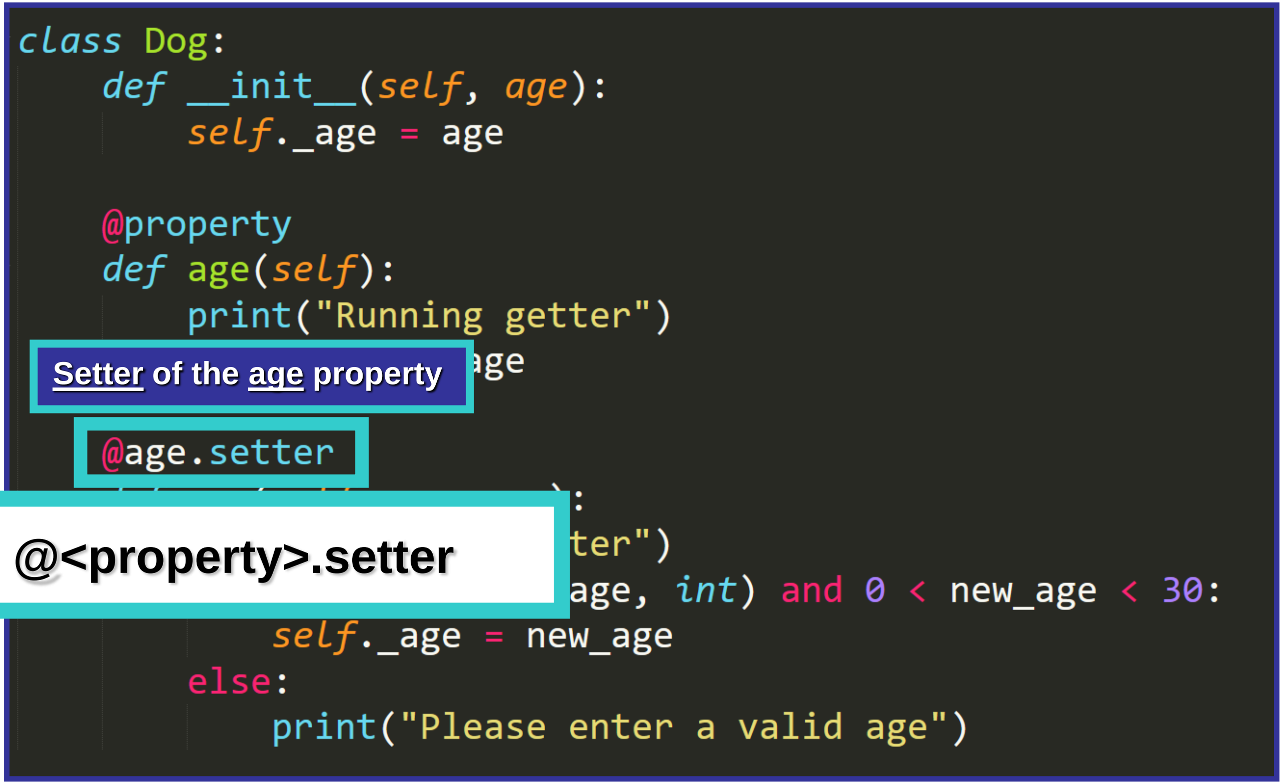
The @property decorator



```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
                     age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

Setter of the age property

The @property decorator

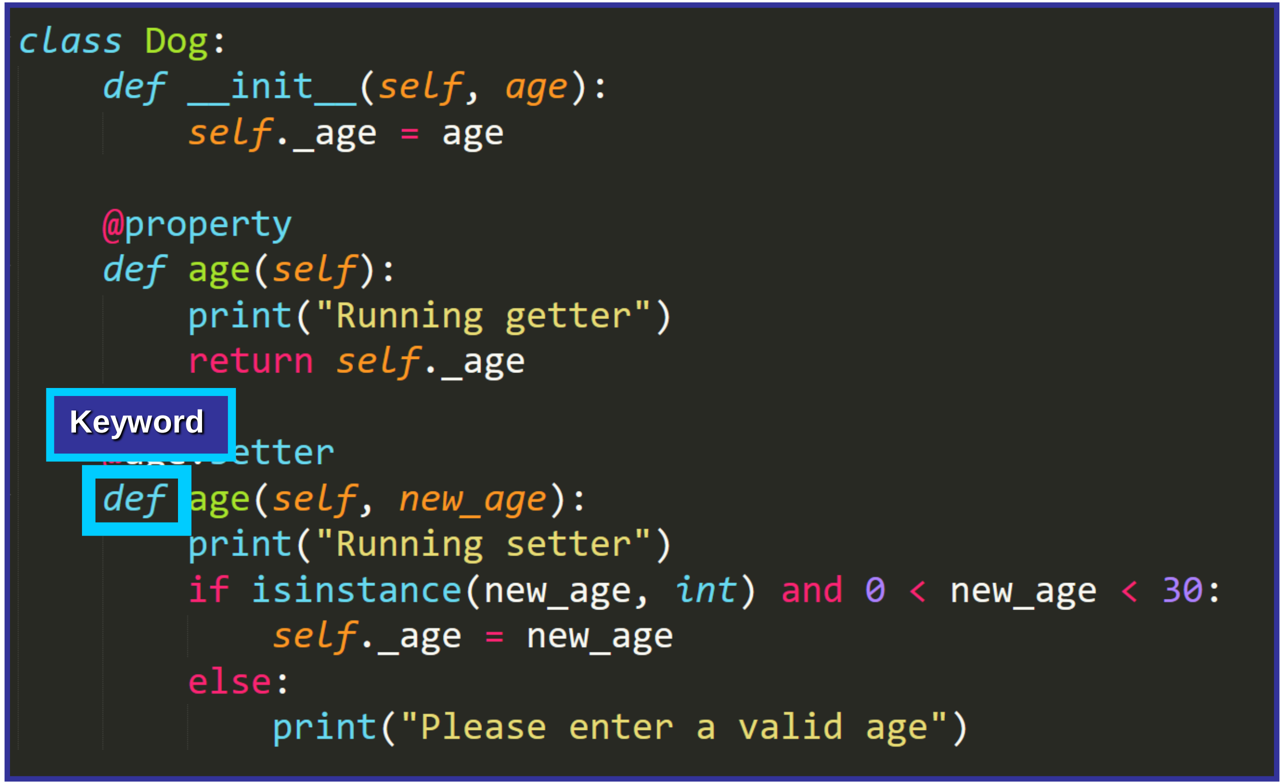


```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
                     age

    @age.setter
                          ):
                          ter")
                          age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

Setter of the age property

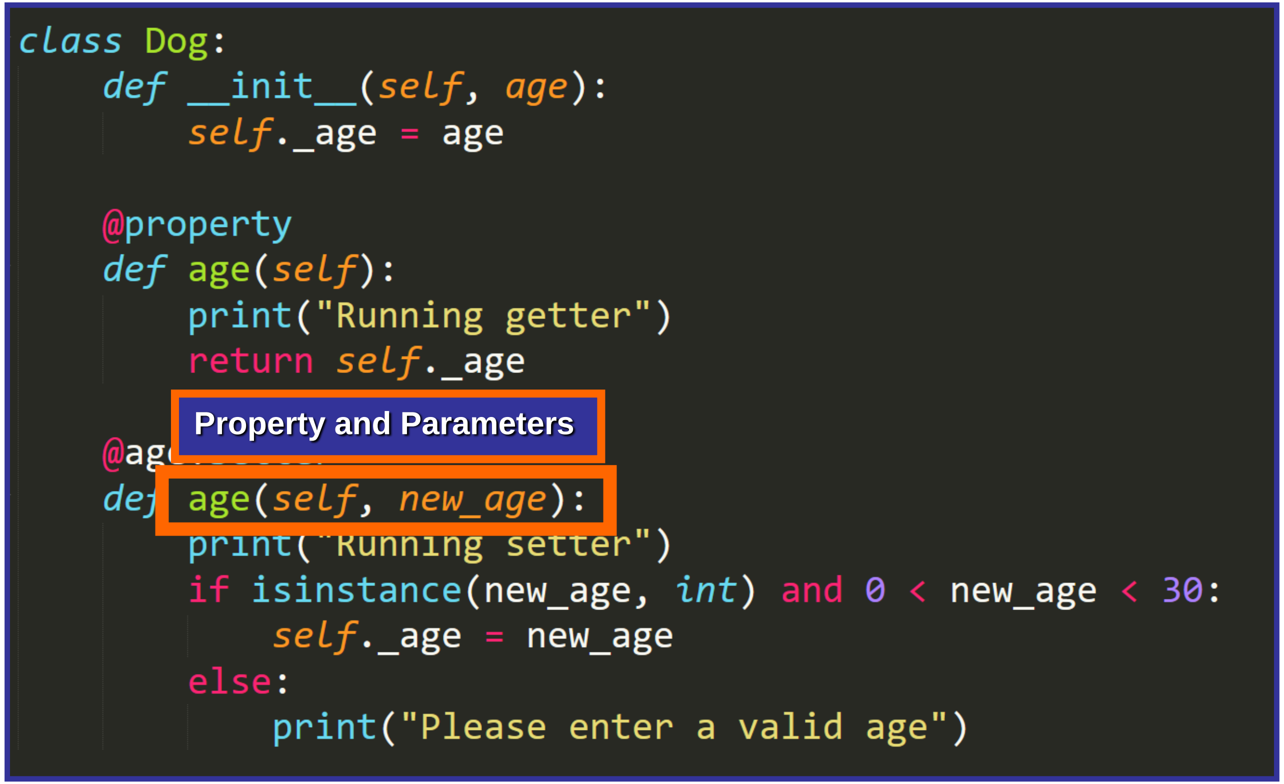
@<property>.setter

The @property decorator



```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

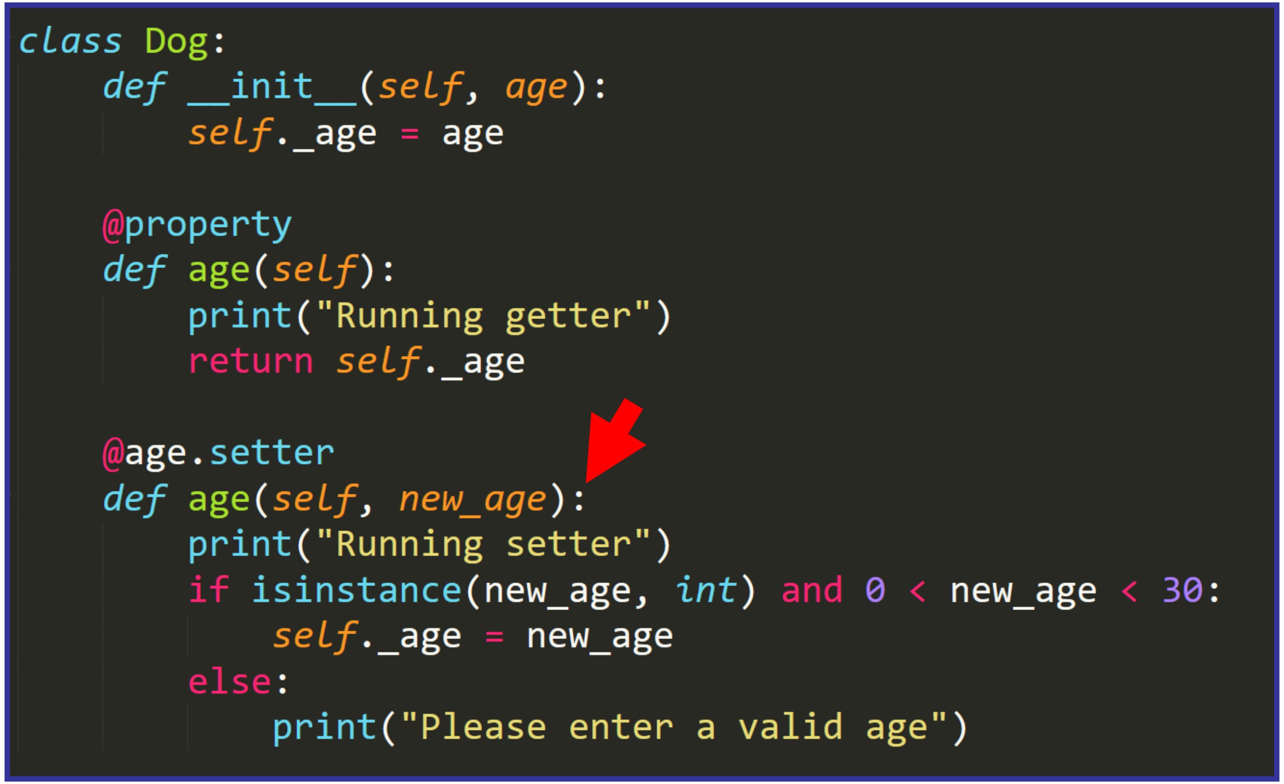
The @property decorator



```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @ag
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

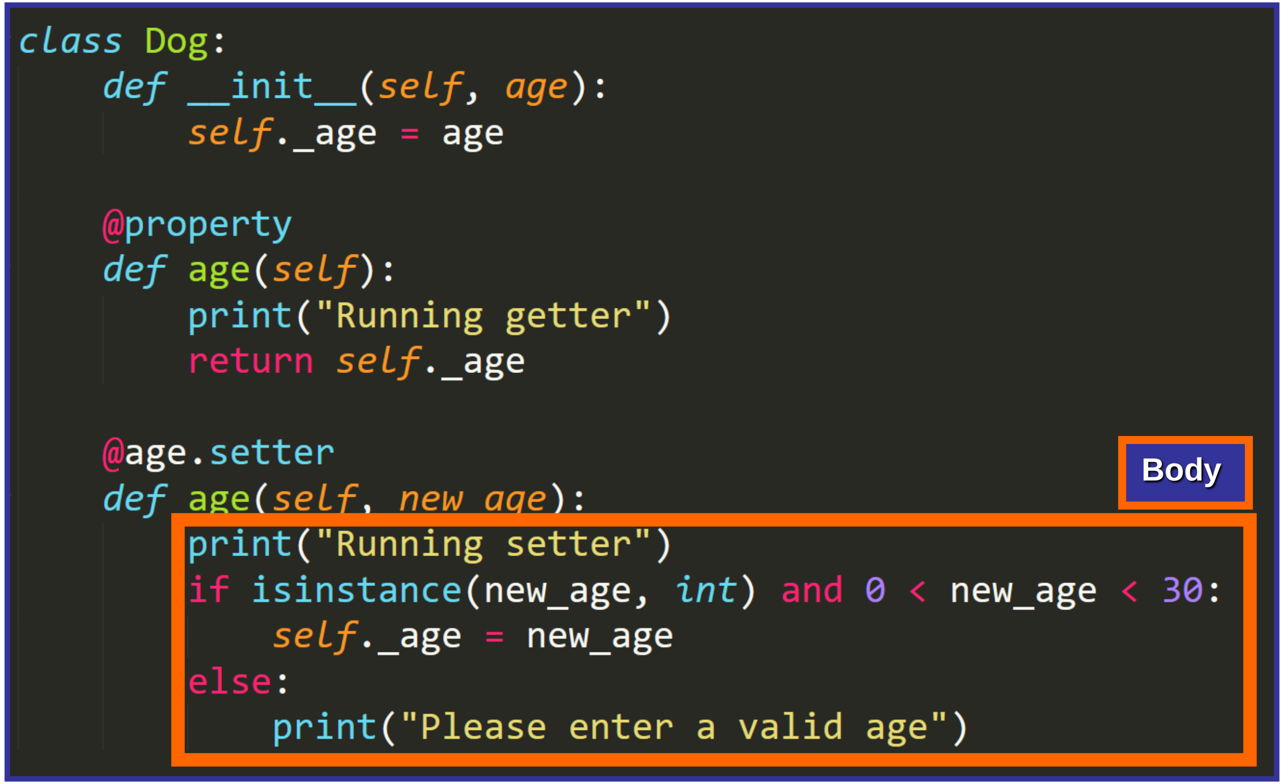
Property and Parameters

The @property decorator



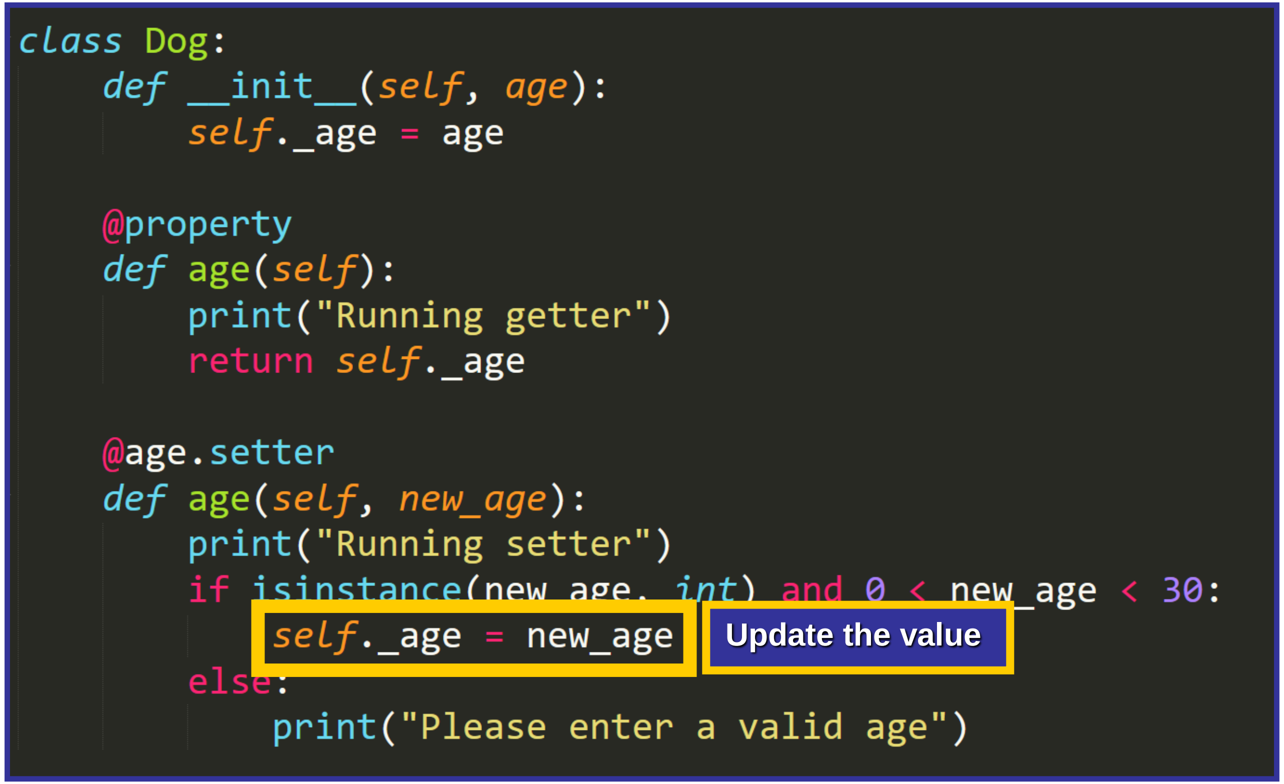
```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator



```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator



```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")


>>> dog1 = Dog(15)
>>> dog1.age = 16
Running setter
>>> dog1.age
Running getter
16
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")


>>> dog1 = Dog(15)
>>> dog1.age = 16
Running setter
>>> dog1.age
Running getter
16
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")

>>> dog1 = Dog(15)
>>> dog1.age = 16
Running setter
>>> dog1.age
Running getter
16
```

The @property decorator

```
>>> class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")


>>> dog1 = Dog(15)
>>> dog1.age = 16
Running setter
>>> dog1.age
Running getter
16
```

The @property decorator

Previously

```python
class Dog:
    def __init__(self, age):
        self._age = age

    def get_age(self):
        print("Running getter")
        return self._age

    def set_age(self, age):
        print("Running setter")
        if isinstance(age, int) and 0 < age < 30:
            self._age = age
        else:
            print("Please enter a valid age")

    age = property(get_age, set_age)
```

Now

```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator

Previously

```python
class Dog:
    def __init__(self, age):
        self._age = age

    def get_age(self):
        print("Running getter")
        return self._age

    def set_age(self, age):
        print("Running setter")
        if isinstance(age, int) and 0 < age < 30:
            self._age = age
        else:
            print("Please enter a valid age")

    age = property(get_age, set_age)
```

Now

```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator

Previously

```python
class Dog:
    def __init__(self, age):
        self._age = age

    def get_age(self):
        print("Running getter")
        return self._age

    def set_age(self, age):
        print("Running setter")
        if isinstance(age, int) and 0 < age < 30:
            self._age = age
        else:
            print("Please enter a valid age")

    age = property(get_age, set_age)
```

Now

```python
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

The @property decorator

Previously

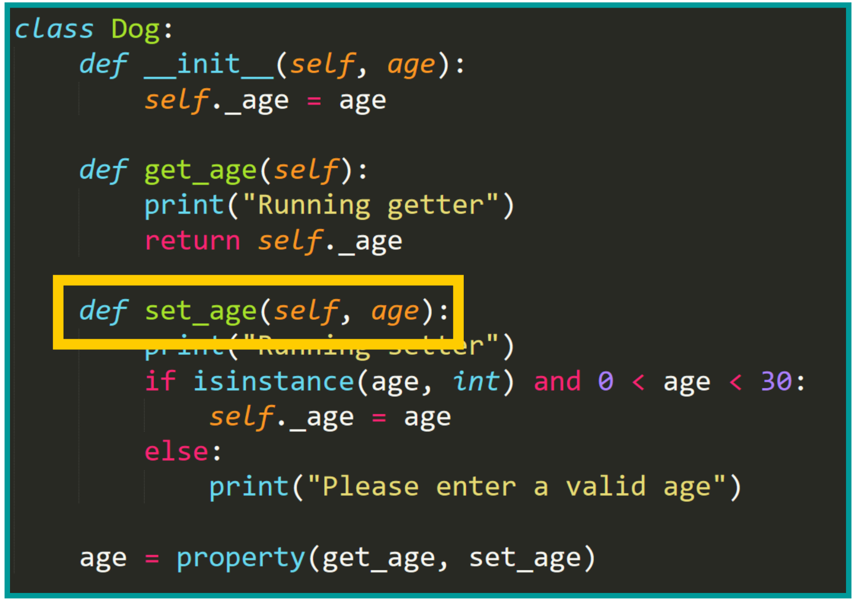
```
class Dog:
    def __init__(self, age):
        self._age = age

    def get_age(self):
        print("Running getter")
        return self._age

    def set_age(self, age):
        print("Running setter")
        if isinstance(age, int) and 0 < age < 30:
            self._age = age
        else:
            print("Please enter a valid age")

    age = property(get_age, set_age)
```

Now

```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

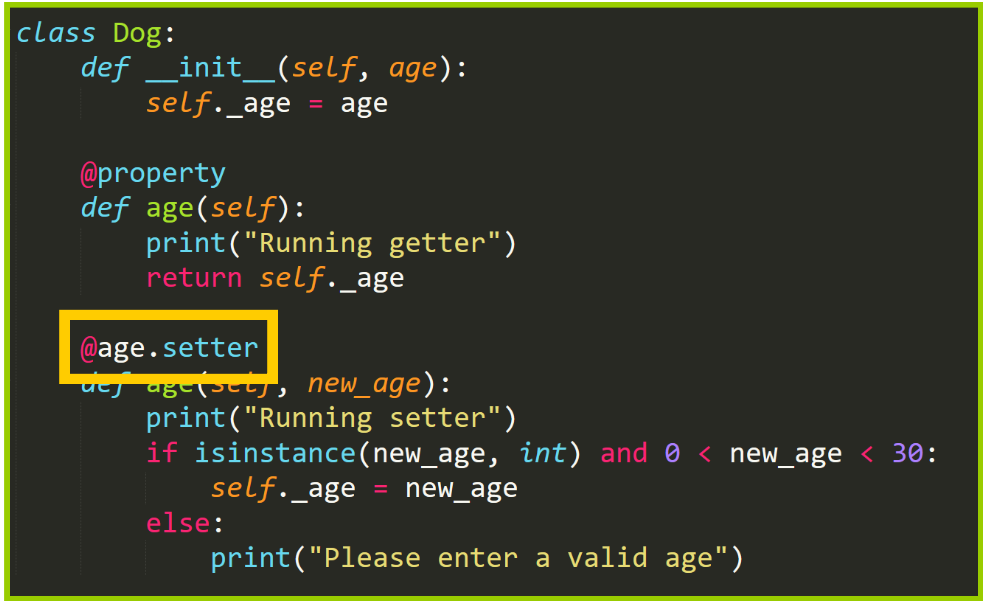
The @property decorator

Previously



```
class Dog:
    def __init__(self, age):
        self._age = age

    def get_age(self):
        print("Running getter")
        return self._age

    def set_age(self, age):
        print("Running setter")
        if isinstance(age, int) and 0 < age < 30:
            self._age = age
        else:
            print("Please enter a valid age")

    age = property(get_age, set_age)
```

Now



```
class Dog:
    def __init__(self, age):
        self._age = age

    @property
    def age(self):
        print("Running getter")
        return self._age

    @age.setter
    def age(self, new_age):
        print("Running setter")
        if isinstance(new_age, int) and 0 < new_age < 30:
            self._age = new_age
        else:
            print("Please enter a valid age")
```

Now... Example