Computing

Lesson 2: Classes and Objects

KS4 - Object-oriented programming

Mac Bowley

Materials from the Teach Computing Curriculum created by the National Centre for Computing Education

Task 1 - What is an animal?

What do all animals have in common, give some examples of both properties and actions.

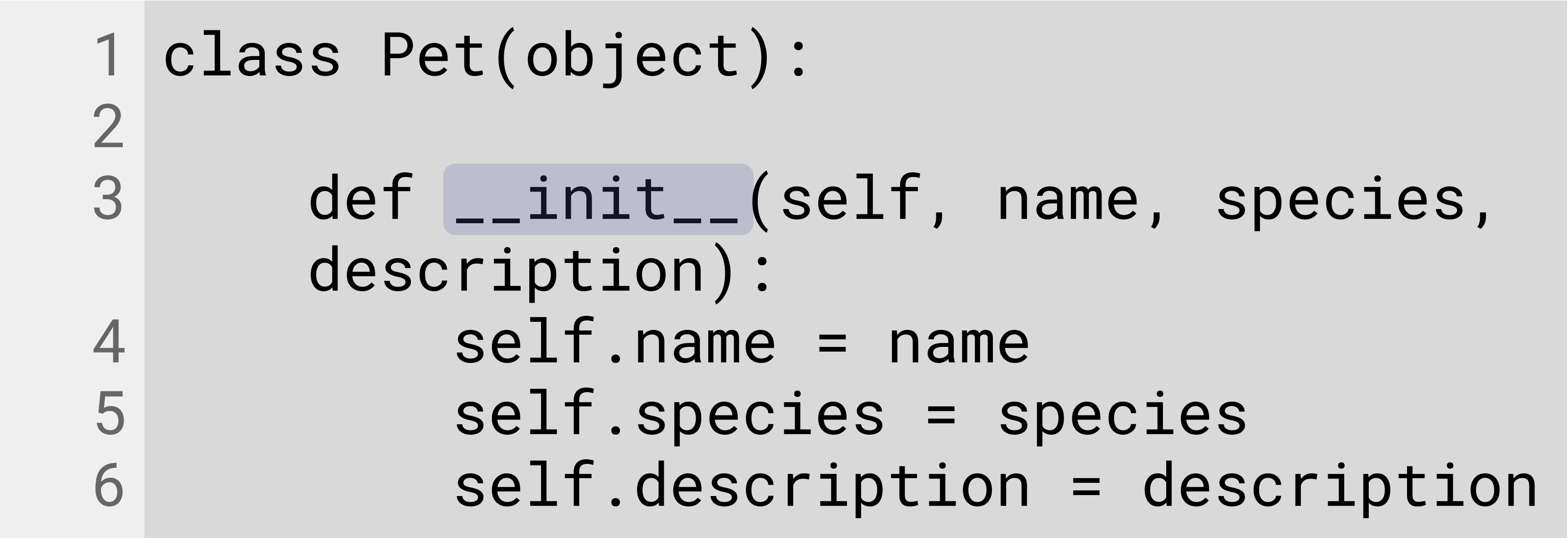
Task 2.1 - Creating a template for a class

```python
class Pet(object):

```

Create a new file called **`pets.py`**

Inside the file start by creating an identifier for your class with the object you want to create, in this example we are going to create an object for a pet.

Task 2.2 - Creating a template for a class



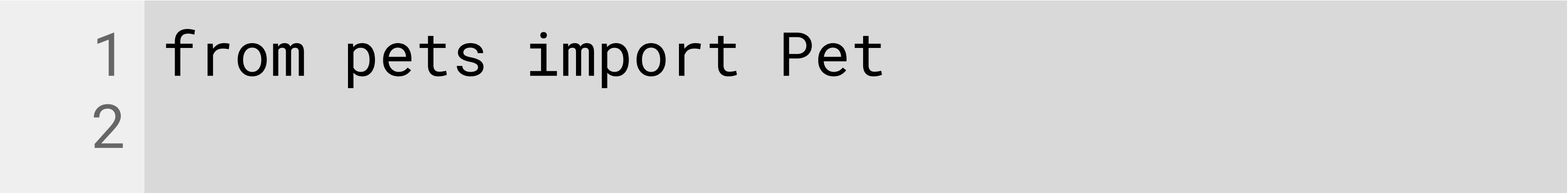
```
class Pet(object):

    def __init__(self, name, species,
    description):
        self.name = name
        self.species = species
        self.description = description
```

Write the constructor, a special method to instruct Python how to create on object of this class.

`init` here stands for ‘initialise’.

In this example you can see the pet will have a name, species, and description. These are the **attributes** that your pet will have.

Task 3.1 - Creating a template for a class



```
from pets import Pet

```

Now you have a class file for our object, you can go ahead and create a new instance of a pet.

In the **`main.py`** file, you are going to import the Pet object from our custom class file pets.

Task 3.2 - Creating a template for a class

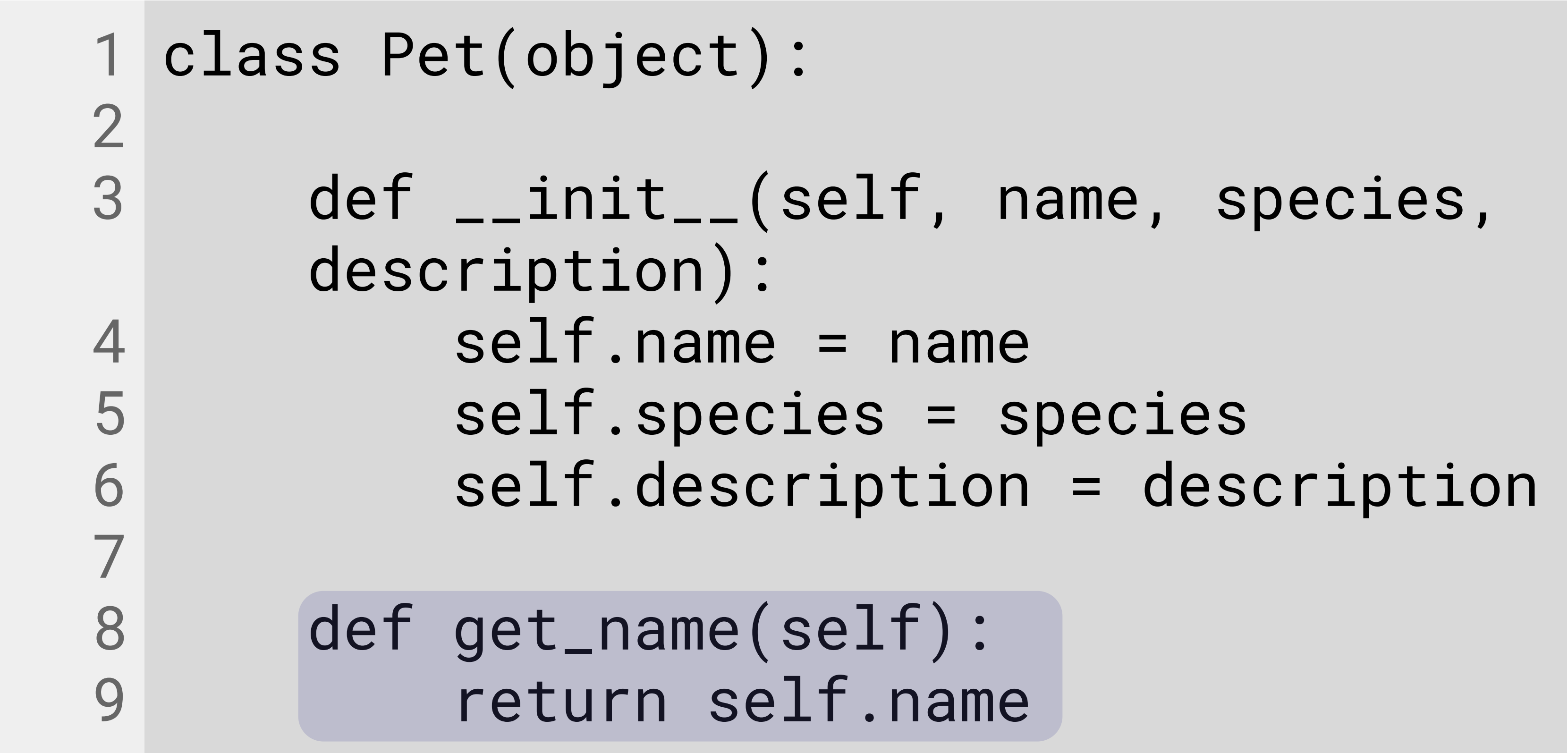
```
from pets import Pet
my_cat = Pet("Fluffy", "Cat", “Black and white long haired.”)
```

Next, you need to define the attributes of our pet.

Here we have created a variable called `my_cat` that is of type Pet.

The pet’s name has been assigned the value “`Fluffy`”, the species has been assigned the value “`Cat`”, and a description has been included.

Task 4.1 - Creating getters for an object



```
class Pet(object):

    def __init__(self, name, species, description):
        self.name = name
        self.species = species
        self.description = description

    def get_name(self):
        return self.name
```

A method has been added opposite to get the name of the pet.

Can you add this, as well as two more methods to get the species, and the description?

Task 5.1 - Accessing attributes on an object

```
from pets import Pet
my_cat = Pet("Fluffy", "Cat", “Black and white long haired.”)

print(my_cat.get_name())
```

Go back and open up **`main.py`**.

Using a print statement, test that your getter methods are working.

One has been done for you opposite and you saw it in the video.

Task 6.1 - Accessing attributes on an object

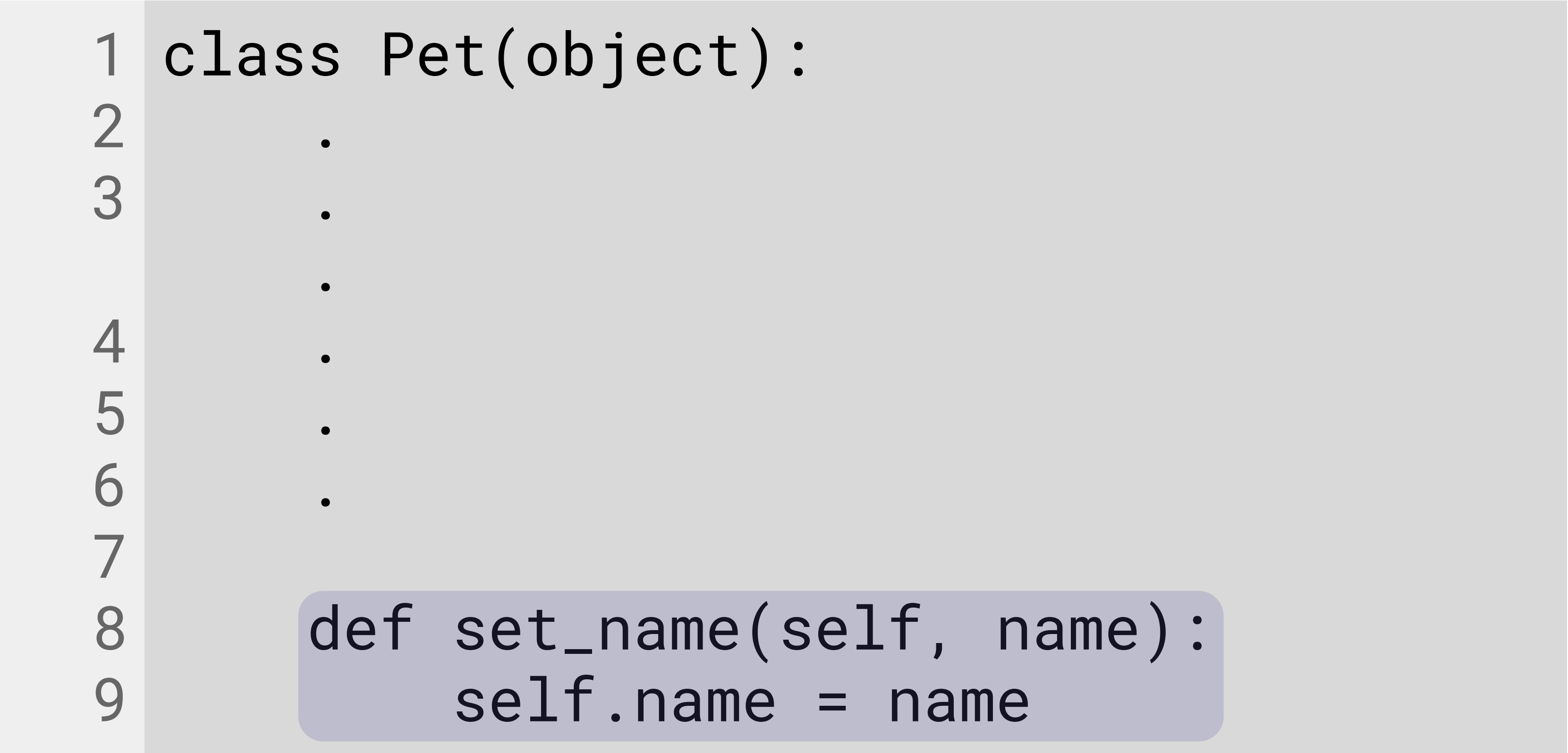
```
from pets import Pet
my_cat = Pet("Fluffy", "Cat", “Black and white long haired.”)

my_cat.describe
```

```
Fluffy is a Cat. Black and white long
haired.
>>>
```

Go back to **`pets.py`**.

Add an additional method called `describe()` which will print out the object’s attribute details when called, as shown opposite.

Task 7.1 - Creating setters for our object



```
class Pet(object):
        .
        .
        .
        .
        .
        .

        def set_name(self, name):
            self.name = name
```

Modify your **`pets.py`** file to include the `set_name()` method, and

Check it works by calling it in your **`main.py`** file using your `describe()` method.

Create additional methods in your pets file to set the species and the description.

Task 8 - Create some pets

1. Create three more different types of pets, giving them a name, species, and a description.
2. Check that your pets are being **initialised**, by calling your `describe()` method on them.
3. Use get and set methods for each pet.