object refers to its class via the `self.__class__` attribute.

The `how_many` is actually a method that belongs to the class and not to the object. This means we can define it as either a `classmethod` or a `staticmethod` depending on whether we need to know which class we are part of. Since we refer to a class variable, let's use `classmethod`.

We have marked the `how_many` method as a class method using a decorator.

Decorators can be imagined to be a shortcut to calling a wrapper function (i.e. a function that "wraps" around another function so that it can do something before or after the inner function), so applying the `@classmethod` decorator is the same as calling:

```
how_many = classmethod(how_many)
```

Observe that the `__init__` method is used to initialize the `Robot` instance with a name. In this method, we increase the `population` count by 1 since we have one more robot being added. Also observe that the values of `self.name` is specific to each object which indicates the nature of object variables.

Remember, that you must refer to the variables and methods of the same object using the `self` *only*. This is called an *attribute reference*.

In this program, we also see the use of *docstrings* for classes as well as methods. We can access the class docstring at runtime using `Robot.__doc__` and the method docstring as `Robot.say_hi.__doc__`

In the `die` method, we simply decrease the `Robot.population` count by 1.