Computing

Lesson 3: Creating a Class

KS4 Object-Oriented Programming

Mac Bowley

Recapping OOP terms

Method	A special method used to create a new object from a class
Instance	The associated actions of an object.
Constructor	The individual properties of an object.
Attribute	Every new object created from the same blueprint.

Task 1 - Planning a new class

Requirements:

1. Each monster must have a name, health points and a line of dialogue.
2. Each monster must be able to take damage and speak to the player.

Credit: Pixabay

Task 1 - Planning a new class

Requirement 1:

Each monster must have a name, health points and a line of dialogue.

1. **What attributes do you need to add to meet these criteria?**
2. **What type of data will each of these attributes be**

Credit: Pixabay

Task 1 - Planning a new class

Requirement 2:

Each monster must be able to take damage and speak to the player.

1. **What actions (methods) does your monster class need to be able to use?**
2. **How will they work, what data do they need as parameters?**

Credit: Pixabay

Task 3 - Create getters and setters

1. Add getters and setters to your class.
   a. Make sure you have one for each of the attributes in the Monster class.

Use the Pet example to help you remember how they should look.

```
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

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

Task 3 - Create getters and setters

1. Create a new file called **main.py**
2. Import your class on the first line.
3. Create a Monster object using the constructor.
4. Give it a name, a health point total and a line of dialogue as arguments for the constructor.

```
from pets import Pet

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

Task 4 - Adding the methods

Now add the two methods we planned earlier, use the Pet class as an example again and the next two slides to remind yourself what the methods should do.

```
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

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

    def describe(self):
        print("%s is a %s. %s" %
             (self.name, self.species,
                    self.description))
```

Task 4 - Adding the methods

`take_damage(self, damage):`

- Take the damage parameter away from health attribute.
- Print the new health total.

```python
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

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

    def describe(self):
        print("%s is a %s. %s" %
              (self.name, self.species,
                     self.description))
```

Task 4 - Adding the methods

`speak(self):`

- Print who is speaking (name).
- Print the line of speech.

```
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

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

    def describe(self):
        print("%s is a %s. %s" %
             (self.name, self.species,
                    self.description))
```

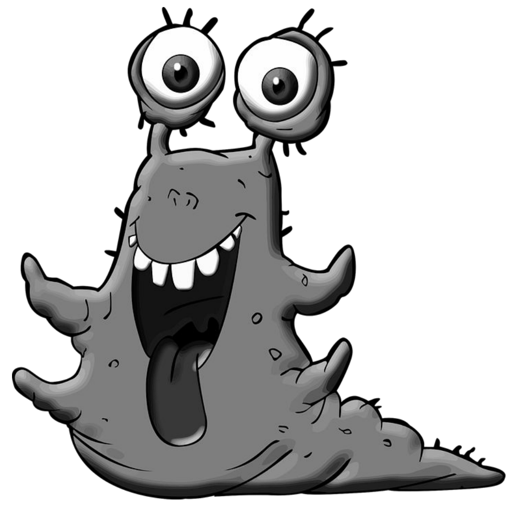
Task 5 - Testing your Monsters

Explorer Tasks:

- Create 3 Monster objects.
- Do damage to 2 of them.
- Make the third monster speak.

Use the new methods to do this.

Credit: Pixabay


Credit: Pixabay

Credit: Pixabay