Question 1:

1.1

i) Define a simple class called `Individual`.
ii) Add an initialisation method which initialises the `self.character_name` instance attribute.
iii) Add an access method to the class that returns `self.character_name`. Call this method `get_character_name()`.
iv) Create an instance of the character class and assign it to the variable `individual1`. This class instance should be assigned the `character_name` 'Buster' on initialisation.
v) Create another instance, which should be assigned to the variable `individual2`. Set the name to 'Tobias'.
vi) Print the character name of `individual1` and `individual2` to the screen using the appropriate method.
vii) Save this to a script called `oop1.py`.

1.2

Let's build on our individual class a little more to make it more interesting.

i) On initialisation, set the instance attribute `self.happy` to `True`. This should be done by default (i.e. no parameter needs to be passed on instantiation in order to do this.)
ii) Create a predicate method `is_happy` to return the status of `self.happy`.
iii) Create a modification method named `switch_mood()` that changes `self.happy` from `True` to `False` (and vice versa).
iv) Create a method called `speak()` that returns "Hello, I am [self.name]" or 'Go away!', depending on whether `self.happy` is set to `True` or `False` respectively.
v) Create `individual3` with character name initialised to 'Lucille'
vi) Write some code to try out these methods/attributes of Buster and Tobias.
vii) Save all this code to a script called `oop2.py`.

Question 2:

Write the definition of a Point class. Objects from this class should have a

- a method **shown** to display the coordinates of the point
- a method **move** to change these coordinates.
- a method **dist** that computes the distance between 2 points.

The following provides an example of the expected behavior of objects belonging to this class:

```
>>> p1 = Point(2, 3)
>>> p2 = Point(3, 3)
>>> p1.show()
(2, 3)
>>> p2.show()
(3, 3)
>>> p1.move(10, -10)
>>> p1.show()
(12, -7)
>>> p2.show()
(3, 3)
>>> p1.dist(p2)
1.0
```