# 159.172 Computational Thinking

# Tutorial 4: Creating Classes and Objects

In this tutorial you will develop class definitions and create objects that are instances of those classes.

Go to Stream and download the files

```
critters.py
my_critters.py
```

Set up a new project and add these files to it.

**critters.py** is a program that is meant to display caterpillars and butterflies (and any other critters that you want to create) against a simple background scene. When you run this program, you will see a screen, 1000 pixels wide by 1000 pixels high, displaying a basic background scene.

On each iteration of the main program loop we check to see if the user wants to quit, or if he has pressed the 'c' key, which initiates creation of a new **Caterpillar** object that is appended to the **critterlist** (initially empty). After that, we update the screen by redrawing the background scene and the critters in the critter list. To begin with, a caterpillar is just depicted as a face, no body parts.

If you look at the top of the program **critter.py** , you will see that two modules are imported, **pygame** and **my_critters**. A little further down, you will see the drawing function for the background, **draw_background()**. If you want a nicer background picture, you can alter this function.

If you now go to the module **my_critters.py** you will see that this module implements a basic implementation for a Caterpillar class.

A caterpillar has the following attributes:

1. **xcoord** and
2. **ycoord**

horizontal and vertical coordinates which are used by the **draw_critter(self, screen)** method to give the location of the top left boundary of the graphic that represents the caterpillar.

We create an instance of a caterpillar object in our main program **critters.py** with the code:

```
newcaterpillar = mycritters.Caterpillar()
```

# Tasks

1. Add a **`size`** attribute to the Caterpillar class, an integer between 0 and 3. Then, improve the **`draw_critter(self, screen)`** method so that it takes account of the size attribute and draws a whole caterpillar, not just a face. The size attribute may determine the number and complexity of the caterpillar parts as well as the overall size of the graphic.

   Documentation for the required Pygame drawing commands can be found at www.pygame.org/docs/ref/draw.html.

   Consider other attributes that could be added to the Caterpillar class to enhance the display of these critters - health, age, colour-scheme, species … and so on.

2. Create a Butterfly class that will allow butterflies to be created and displayed. It is up to you to determine the attributes that a butterfly will require, but you should at least include location attributes and a **`draw_critter(self, screen)`** method to allow Butterfly objects to be displayed.

3. Add code to the **`critters.py`** file so that when the user presses the 'b' key a new **`Butterfly`** object is created and appended to the **`critterlist`**.

4. Set the location attributes for your two classes so that caterpillars are located at ground level or below, and butterflies are located in the sky.

5. Add a **`colour_scheme`** attribute to each of your classes (make this a list of colours) and then update your **`draw_critter`** methods to take account of this **`colour_scheme`** attribute.

6. Add a **`change_colour(self)`** method to each of your classes that can be used to switch between two or more colour schemes.

7. Add code to the **`critters.py`** file so that when the user presses the 's' key the **`change_colour(self)`** method is invoked on each of the critters in the **`critterlist`**.

# Submission

Submit your completed code, contained in the files **`critters.py`** and **`my_critters.py`**, via Stream. Internal students: please note that you must attend a workshop session and show evidence of making progress on this tutorial during the session in order to gain marks for your work.