Name_____

Lesson 10: Worksheet 10.1 – Vampire robot

In this challenge-based activity, you will apply everything you have learned so far to create a program with lots of interesting behaviours for your Edison robot to perform while transformed into a vampire robot.

To turn your Edison into a vampire robot, we will use object-oriented code.

Objects and classes in Python

Python is an object-oriented programming language. Object-oriented code is a powerful way of reducing complexity inside a program. That's why many programmers use object-oriented code in large, complicated programs.

In an object-oriented programming language, rather than just using standalone variables and functions, programs can also use 'objects'.

An object is a collection of functions and variables which are related to each other. Objects are defined by a 'class'.

A class (also called a class definition) is like a blueprint that describes how to make an object.

In a program, you can create lots of different objects from a class and then manipulate these objects individually in your program for different purposes. All of the objects created from the class will have the same 'blueprint' when they are created, but you can then do different things to different objects throughout your program.

Example: class 'Shape'

The most common example of a class is a Shape.

All shapes, like squares, triangles and hexagons, have common attributes. For example, all shapes have some number of sides and some number of vertices. There are also common things you do with shapes, like find the area or the perimeter of the shape.

We can use these common elements to create a class for Shape in Python. By creating the class, we are setting up a blueprint with just the known, common elements. We will then be able to create many different objects from that class, such as 'square' and 'triangle'.

The syntax for defining a class in Python is 'class' and then the name you want to give that class, followed by a colon (:). For example:

```
class Shape:
```

Anything you include as indented code in the lines following this is inside the class definition.

Name_____

Look at the following example of a class for Shape in Python:

```
class Shape:
     def __init__(self,x,y):
         self.x = x
         self.y = y

     def area(self):
         return self.x * self.y

     def perimeter(self):
          return 2 * self.x + 2 * self.y
```

This class says that the blueprint for any object created from class Shape will have two input parameters (x and y). The class also defines three functions: 'init', 'area' and 'perimeter'.

The __init__ function and 'self'

Each class must always have an __init__ function.

The __init__ function is start-up code. The term 'init' stands for 'initialisation'.

When the program needs to create an object, the program goes into the class definition and runs the __init__ function. This function contains code which sets up the object, which typically means setting the start values of the object variables.

Functions in a class must always have 'self' as their first parameter. This tells the created object to use its own functions and variables.

Example: object 'rectangle'

Once we have created a class definition, we can use it to create objects in our program.

Let's say we want to create an object 'rectangle' in class Shape.

In Python, when you want to create an object from a class, you use the syntax object_name = class_name(parameters).

Look at the example of creating object 'rectangle':

```
rectangle = Shape(100,45)
```

In a program, this calls the __init__ function, which runs, creating an object 'rectangle' using the blueprint from the Shape class definition. The object 'rectangle' is set to have 100 be the 'x' input value and 45 as its 'y' input value. (You don't need to put in 'self' – that's only a reference for the program to know where to look.)

Once the program creates the object, it can access that object's functions as defined by the class.

Name_____

To call one of these functions in Python, you use the syntax object_name.function_name(parameters).

For example:

```
rectangle.area()
```

In this example, the rectangle.area() function will return the value of the rectangle object's 'x' input parameter multiplied by its 'y' input parameter.

One way to use this particular function would be to store this value in a variable in your program. For example:

```
AreaOfRectangle = rectangle.area()
```

Setting up a class in EdPy

In EdPy, we can use a class to create different objects in a program for Edison.

Here is a simple class for a Vampire robot with an __init__ function and two other functions: one for daytime behaviour for the robot and one for night time behaviour. The Vampire class also contains a variable for the age of the robot:

```
class Vampire:

    def __init__(self,age):
        self.age = age

    def doDayTimeBehaviour(self):
        Ed.LeftLed(Ed.OFF)
        Ed.RightLed(Ed.OFF)
        Ed.PlayTone(Ed.NOTE_A_7, self.age)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
        Ed.TimeWait(1, Ed.TIME_SECONDS) # for debugging purposes and as a placeholder

    def doNightTimeBehaviour(self):
        Ed.RightLed(Ed.ON)
        Ed.LeftLed(Ed.ON) # for debugging purposes and as a placeholder
```

This class definition will allow you to create different Vampire objects.

You can then use these Vampire object in a program for your Edison robot.

Name_____

Write the following code and check that you understand what is happening in the program:

```
#-------------Setup----------------

import Ed

Ed.EdisonVersion = Ed.V2

Ed.DistanceUnits = Ed.CM
Ed.Tempo = Ed.TEMPO_MEDIUM

#--------Your code below-----------

myEdisonVampire = Vampire(21)

while True:
    # if it is the daytime
    if Ed.ReadLeftLightLevel() > 100:
        myEdisonVampire.doDayTimeBehaviour()
    #it is night time
    else:
        myEdisonVampire.doNightTimeBehaviour()

class Vampire:

    def __init__(self,age):
        self.age = age

    def doDayTimeBehaviour(self):
        Ed.LeftLed(Ed.OFF)
        Ed.RightLed(Ed.OFF)
        Ed.PlayTone(Ed.NOTE_A_7, self.age)
        while Ed.ReadMusicEnd() == Ed.MUSIC_NOT_FINISHED:
            pass
        Ed.TimeWait(1, Ed.TIME_SECONDS) # for debugging purposes and as a placeholder

    def doNightTimeBehaviour(self):
        Ed.RightLed(Ed.ON)
        Ed.LeftLed(Ed.ON) # for debugging purposes and as a placeholder

```

Your turn:

Task 1: Design the Vampire behaviours

Now it is time to design some more interesting behaviours for your Vampire robot. Be creative! You can make your robot react to button pushes, claps and obstacles. Consider using event handling in your program. You can make multiple vampire objects with whatever ages you want. You can also modify the base program so that your objects have multiple input parameters.

Think about what you want to include in your program, then work through your program design by first creating flowcharts and then pseudocode based off of the flowcharts.

Name_____

Step 1: Flowcharts

Draw flowcharts for the daytime and nighttime behaviours, either by hand or using a program like Google Slides or a similar drawing program. Include your flowcharts here. Use extra paper if you need.

Name_____

Step 2: Pseudocode

Translate your flowcharts into pseudocode. Write your pseudocode here. Use extra paper if you need.

Name_____

Task 2: Write your program

Translate your pseudocode into Python code using the EdPy app. Download it to your Edison robot and run your program to test it out.

Task 3: Problem-solving

In programming, it is very common not to have everything work out the way you think it will the first time. No sweat! Remember this: Thomas Edison famously failed 10,000 times before he successfully invented the light bulb. So, *keep trying!*

We learn from the problems we encounter, so it is a good habit to take time to think about what went wrong. Describe any problems you had, what ideas you came up with to fix them, and the solutions you found in later attempts. Use extra paper, if you need.

Name_____

Task 4: Describe some of the programming structures in your program

Choose three different programming structures you used in your program. Describe what each one does below.

1. Name of the programming structure:
   _____

   What does it do?

_____

_____

_____

_____

2. Name of the programming structure:
   _____

   What does it do?

_____

_____

_____

_____

3. Name of the programming structure:
   _____

   What does it do?

_____

_____

_____

_____

Task 5: Present your program

Demonstrate your Vampire robot to your partner, group or class. Talk about your ideas, the program, the problems that you encountered and how you solved them.