

e-Yantra Robotics Competition (eYRC-2016)

Task 1 – Model a Terrain

We will now see the next sub module in BPY which is `bpy.data`. This module is used for all blender/python access. Python accesses Blender's data in the same way as the animation system and user interface; this implies that any setting that can be changed via a button can also be changed from Python.

Accessing data from the currently loaded blend file is done with the module `bpy.data`. This gives access to library data.

For example:

1. **`list(bpy.data.objects)`**: Lists the number of objects in 3D window. In our case its Camera, a Cube and a Lamp which is available by default.

```
>>> list(bpy.data.objects)
[bpy.data.objects['Camera'], bpy.data.objects['Cube'], bpy.data.objects['Lamp']]
```

2. **`bpy.data.objects`**: Gives you the count of objects in the 3D window.

```
>>> bpy.data.objects
<bpy_collection[3], BlendDataObjects>
>>> |
```

Similarly, **`bpy.data.scenes`** and **`bpy.data.materials`** will give you the count of scenes and materials used respectively.

To access members of the collection, you can use an index as well as a string of the collection.

For e.g. When **`list(bpy.data.objects)`** was entered it listed the objects in a sequence: Camera, Cube and then Lamp.

If you want to access the object by its index, you can simply write

`bpy.data.objects[1]` will give Cube.

```
>>> list(bpy.data.objects)
[bpy.data.objects['Camera'], bpy.data.objects['Cube'], bpy.data.objects['Lamp']]

>>> bpy.data.objects[0]
bpy.data.objects['Camera']

>>> bpy.data.objects[1]
bpy.data.objects['Cube']

>>> bpy.data.objects[2]
bpy.data.objects['Lamp']

>>> |
```

And if you want to access the object by its name, then

`bpy.data.objects['Cube']`

```
>>> bpy.data.objects['Cube']
bpy.data.objects['Cube']

>>> bpy.data.objects['Camera']
bpy.data.objects['Camera']

>>> |
```

3. `bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.x`: Give the location of the object in x axis

```
>>> bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.x
1.0

>>> |
```

Similarly,

```
>>> bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.x
1.0

>>> bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.y
0.9999999403953552

>>> bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.z
-1.0

>>> bpy.data.scenes[0].objects["Cube"].data.vertices[0].co.xyz
Vector((1.0, 0.9999999403953552, -1.0))

>>> |
```

#Code to make trapezium out of default cube

```
import bpy
bpy.data.objects["Cube"].data.vertices[0].co.x += 1.0
bpy.data.objects["Cube"].data.vertices[0].co.y += 1.0
bpy.data.objects["Cube"].data.vertices[1].co.x += 1.0
bpy.data.objects["Cube"].data.vertices[1].co.y += 1.0
bpy.data.objects["Cube"].data.vertices[2].co.x -= 1.0
bpy.data.objects["Cube"].data.vertices[2].co.y -= 1.0
bpy.data.objects["Cube"].data.vertices[3].co.x -= 1.0
bpy.data.objects["Cube"].data.vertices[3].co.y -= 1.0
```