Python Classes and Objects

You may know that python is an object-oriented programming language, but so far we have not talked about that much explicitly. What are objects? Most variables that we use are objects. Objects are specific instances of classes, which let us label things that contain data, and allow specific functions on them. A class is basically the same thing as a type, and you can create new ones yourself. You are now very familiar with writing your own functions, and soon you will also know how to write your own classes. Objects may have **data attributes** and **method attributes**, both of which are defined inside the class description. As an example, we create an object in the `numpy.array` class as follows:

```
import numpy as np
a=np.array([3,1,4,2])
```

This looks like a function, and you may have been thinking of this as a function that turns one type of object into another, but `numpy.array` is not a function; it is a class. The line of code above instantiates an object inside that class. Now we can ask for data attributes (no parentheses), or method attributes (parentheses):

```
a.shape
   Out: (4,)
a.dtype
   Out: dtype('int64')
a.sort()
a
   Out: array([1, 2, 3, 4])
a.resize([2,2])
a
   Out: array([[1, 2], [3, 4]])
```

Let's create our own class. This class will create objects which have a numerical value and a name as their data, and will have methods that can double the numerical data or change the name. We also keep track of how often the name of an object has been changed. Note how we can manipulate different parts of the data independently! The `self` word comes up a lot, and we need this to specify the local data variables, which live only inside the object. The first line of the class definition, with the `__init__` part, is almost always there, since it sets the initial values of the data. They do not need to be user specified.

```
class dblclass(object):
    def __init__(self,val,name):
        self.val=val
        self.name=name
        self.chng=0

    def double(self):
        self.val=2*self.val

    def rename(self,newname):
        self.name=newname
        self.chng+=1
```

We can then use this class as follows:

```
b=dblclass(4,'Andrew')
b.val
   Out: 4
b.double()
b.val
   Out: 8
b.name
   Out: 'Hao'
b.rename('Andrew')
b.name
   Out: 'Andrew'
b.chng
   Out: 1
```

Tkinter

Classes will be useful both in Tkinter and in Qt Designer, which we will use to design GUIs, but we start with a few basics of Tkinter. This is a module that lets us open a window which may contain canvasses, labels, buttons, which the user is allowed to interact with, without using python code. We do this by binding "events" such as mouse clicks and keyboard presses to functions. Different types of events have different attributes. For example, a left mouse-click on a canvas has attributes `x` and `y`, corresponding to the coordinates of the cursor at the time of click. In the following code, the `Tk()` class creates the basic GUI object. Then we create a `Canvas` object and add it to the GUI with the `pack()` command. Finally, we `bind` the mouseclick to the creation of a rectangle, of which the upper left corner is at the position of the cursor. The `mainloop()` command launches the GUI.

```
import Tkinter as Tk #or tkinter in v.3
root = Tk.Tk()
w = Tk.Canvas(root, width=500, height=500)
w.pack()
def rectangle(event):
    w.create_rectangle(event.x,event.y, event.x+20, event.y+20, fill="blue")
w.bind("<Button-1>", rectangle)
root.mainloop()
```

For many more examples of events and actions, have a look through the Tkinter documentations and tutorials[1].

Exercises

- Create a list-class yourself: one that has an `append()` function and a length data attribute. (Standard python lists can only give you the length through the `len()` function, which is not an attribute.)
- Create a class of objects that act like integers, and have a `plus(k)` method attribute, which increases the integer by k, and a `parity` data attribute, which is either `'odd'` or `'even'`, and is correctly updated each time.
- Create a game that asks the user to click inside a rectangle. You can decide what happens when the user clicks correctly.
- Look up how to create radiobuttons, and create a GUI that takes the user through a short questionnaire, and stores the answers in an appropriate output variable.

[1]For example, here: `http://www.python-course.eu/python_tkinter.php`