**Classes Part III: Arrays of Objects**

The pain of creating a class of objects becomes worthwhile if we create many objects. If we create a Player class and only create one player, we will have wasted our efforts. The benefit of classes is most apparent when we create an array of objects.

In our previous example, we had three objects:

player1

player2

player3

These were objects of the Player class. Note that class names are always capitalized and object names are always lower case. This is a convention.

Thing – class

thing – object

For example:

**Furniture** is a class. A **chair** is an object of the Furniture class.

**Player** is a class. A **player** is an object of the Player class. And so on.

**Creating a List of players**

The following code assumes that you already have a full Player class. The code snippets below only add to the code you already have.

In your main program we have created two players:

player1 = Player(4, "Charlie", 6) # create a new player

player2 = Player(3, "Joe", 9) # create another new player

Let’s change this to make a list of players:

player = [] # start with an empty list

player.append( Player(4, "Charlie", 6)) # add a new player

player.append( Player(3, "Joe", 9)) # add another new player

Now we have created an array of players, called playerList. In the list, we have added (appended) two new players. What are those players called?

player = [] # start with an empty list

player.append( Player(4, "Charlie", 6)) # add a new player

player.append( Player(3, "Joe", 9)) # add another new player

print (player[0]) # what does this print?

print (player[1]) # what does this print?

This prints the following:

<\_\_main\_\_.Player object at 0x0209ABD0>

<\_\_main\_\_.Player object at 0x0209ABF0>

What is this? A player is an object. It has many parts to it: attributes such as a name, a speed and a health, plus it has methods (things it can do). How do you print a player? You can’t print a player anymore than you can print a human.

The computer can’t print the player, so instead it prints the memory address of where the player object can be found. The address is written in hexadecimal. The addresses are very similar, meaning that both player objects were stored near each other in the computer’s memory. The memory would look something like this:

| 0209ABD0 | 0209ABD1 | 0209ABD2 | 0209ABD3 | 0209ABD4 | 0209ABD5 | 0209ABD6 |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | C | h | A | r | l | I |

And so on. Each memory cell would contain information on the player.

How can we access each individual attribute? It is simple. Let’s change the last two lines a tiny bit:

print (playerList[0].speed) # what does this print?

print (playerList[1].name) # what does this print?

**Creating Multiple Objects using a For Loop**

Let’s not be satisfied with 3 players; why not make 5 players, or 10 players? Once you have set up a list, it’s easy to do using a FOR loop.

For i in range(10):

playerList.append(Player(4, "Charlie", 6)) # add a new player

Now we have 10 new players. The problem is - they are all the same! To make them different, we have three options:

1. **Create each new player by hand**

playerList.append(Player(7, "Bill", 4)) # add a new player

playerList.append(Player(5, "Jad", 9)) # add a new player

etc.

1. **Ask the user to create each new player**

For i in range(10):

n= input("Enter the player’s name")

s = int(input ("Enter the player’s speed"))

h = int(input("Enter the player’s health"))

playerList.append(Player(h,n, s)) # add a new player

1. **Create each player with random attributes**

import random

name = "Bob"

For i in range(10):

s = random.radint(1,10) # computer picks a number

h = random.randint(1,10) # from 1 to 10

playerList.append(Player(h,name, s)) # add a new player

In this last example, since names can’t be random, they are all named “Bob”. This is OK, we will just change each name by hand:

player[0].name = "Jill"

player[1].name = "Lisa"

player[2].name = "Steve"

etc.

Using the random function is most convenient when dealing strictly with numbers and not with strings.

If we wanted to get fancy, we could use a list of possible names, and have the players randomly pick a name from the list.

Exercise

1. Create an array of ten players with individual values created by hand. Have a FOR loop that loops 5 times. Each time it picks two random players and has them hug or punch each other. Don’t allow them to punch or hug themselves!