IGDB api stuff

Client ID: rw59y9anhhaeh0gosyydtkv7fm5a66

Secret: wziq4nlv629nshpnwb0d1hxzjzvsyk

Class Vs. Instance Data

* **Instance Variables**:
  + Exists only when it is explicitly created and added to the object. Initialized in the objects **\_\_init\_\_(self):** method or later at any moment of the **object’s life**
  + Each object (or class instance) contains its own set of these instance variables. They do not interfere with one another.
* **Class Variables:**
  + These are defined in the line after declaring your class
  + They are accessible using: **className.variableName** when you need to modify the variable value
  + Help store data that is common to all instances of a class.

Shallow And Deep Operations:

* Assignment statements in Python **do not create copies of object, they only bind names to an object**.
* A **Shallow Copy**: means constructing a new collection object and then **populating it with references to the child objects found in the original.**
* A **Deep Copy**: means recursively copying by first constructing a new collection object and then **populating it with copies of the child objects found in the original.**
* You can create the copy you’d like by importing the **copy** library and using:
  + copy.copy(target): to make a **shallow** copy of the target
  + copy.deepcopy(target): to make a **deep** copy of the target

Inheritance, Extension, And Polymorphism:

* **Inheritance** allows us to define a new class that inherits all the methods and properties of another class while also allowing us to extend the behavior of the new class. Defined as:
  + **Class childClass(ParentClass):**
* **Extension** allows us to maintain all the constructed elements of the parents class while still allowing to add new attributes. This can be done using the **super()** function:
  + **class childClass(ParentClass):**

**def \_\_init\_\_(self, fname, lname, year):**

**super().\_\_init\_\_(fname, lname):**

**self.graduationyear = year**

* + **class ParentClass():**

**def \_\_init\_\_(self, fname, lname):**

**self.firstName = fname**

**self.lastName = lname**

* **‘Pass’**ing a class through will default construct the prior attributes. Setting a constructor for the child class without using the super() function will override the parent’s attributes:
  + **Class childClass(ParentClass):**

**pass**

**x = childClass(“Mike”, “Olsen”)**

* **Polymorphism:** the word polymorphism means many forms. In programming it refers to **methods/ functions/ operators** with the **same name** that can be **executed on many objects or classes. (e.g. len(), +, student.display(), teacher.display(), etc.)**
* Inheritance rules also apply to polymorphism.
  + Children inherit both attributes and methods from parents.
  + Children also have the ability to customize/ overwrite methods.

The below 2 were not in the slides .\_.

Composition and inheritance:

Copying Object Data: