Puja Patidar  
  
December 10, 2019  
  
Foundations of Programming: Python Assignment 09

**Working with class**

**Introduction**

In this paper I am going over how to work with creating scripts using modules and how Modules are used to organize function and ,inherit code from one class to another,relationships between classes using UML , command console to work with GitHub

**Main Module**

The following main module code was added to the scripts that should not be directly executed:

**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **raise** Exception(**"This file is not meant to ran by itself"**)

Our IO class had the following because it referenced the DataClasses in a static method and needed a way to find the objects located in another file.

**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **raise** Exception(**"This file is not meant to ran by itself"**)  
**else**:  
 **import** DataClasses **as** DC

The main script had the following. It needed a map to all the other files it would need.

**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **import** sys  
 **from** DataClasses **import** Employee **as** Emp  
 **from** ProcessingClasses **import** FileProcessor **as** Fp  
 **from** IOClasses **import** EmployeeIO **as** Eio  
**else**:  
 **raise** Exception(**"This file was not created to be imported"**)

**Class Inheritance**

By default classes inherit the base object class. You can define it any of these three ways:

**class** Person:

**class** Person():

**class** Person(object):

Any additional class that are added can use ‘object’ also or you can indicate a parent class.

**Programming Assignment**

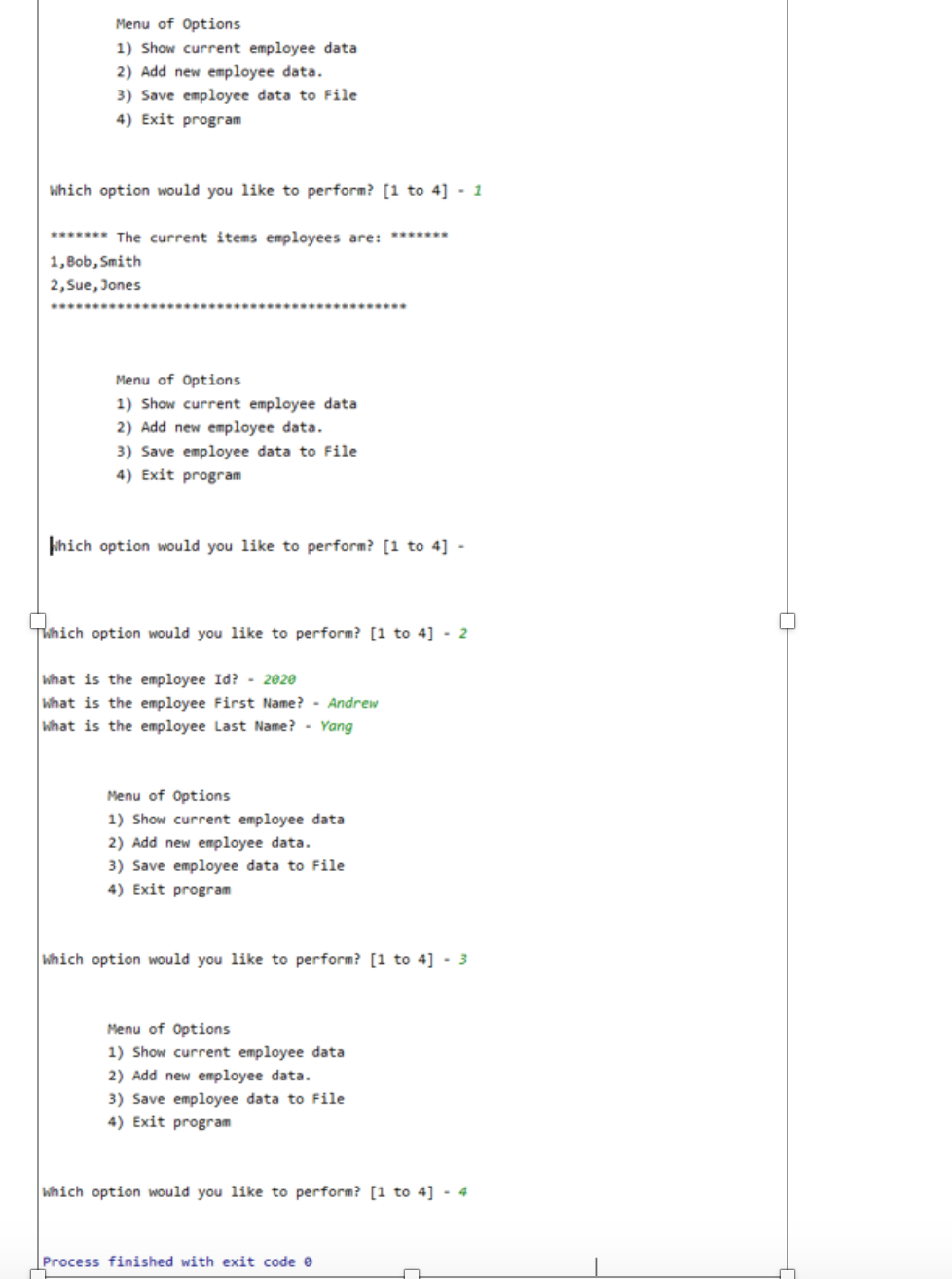
This week we created a Person (parent) class and Employee (child) class. When defining the employee class, we set it up like this:

class Employee(Person):

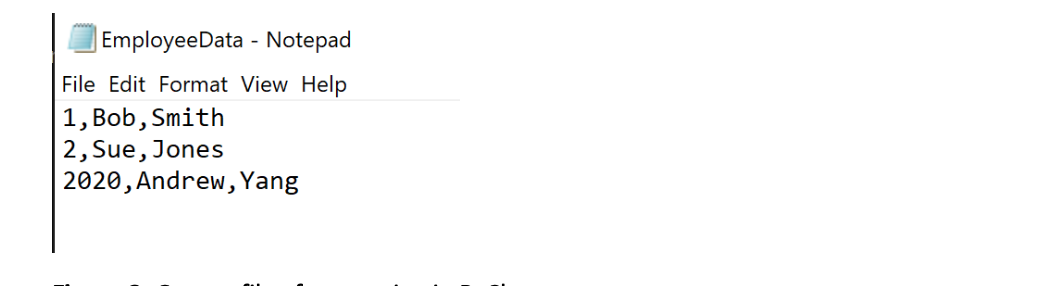
In the main program, after listing the classes to import, we declared our variables and then started the main body of the script. First the file was read and the objects were stored into a list table. Then this was passed as an argument for various operations like printing the current list, adding to the list and writing the list to the file.

After running through the Test Harness code and then building the main functioning code the final output is below (figure 1 - 4).

Figure 1: Run of program in PyCharm



**Figure2. Output file after running in PyCharm**



**UML**: Unified Modeling Language (UML) is a standard way of modeling relationships between software components. UML consists of many types of modeling diagrams, but let’s look at three of the most common ones.

**Use case diagrams** : "Use Case" diagrams are another popular way to show how components within the software, and the actions they perform, will be used by humans and other software. The person or software using the component is called “Actor,” even when the actor is software

If two classes in a model need to communicate with each other, there must be a link between them, and that can be represented by an association (connector).

Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case an arrow is on both sides, the association is known as a bidirectional association.

We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association. The example indicates that a Student has one or more Instructors: External site : <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/>

Git Command Line

To date we have been creating repositories on GitHub. Last week we learned how to update them with GitHub Desktop. This week we learned Git Command line. Here are the steps to get that working.

1. Installed Git
2. Started the Git Bash shell
3. Checked version with:   
   git –version
4. Set global config variables:  
   $ git config --global user.name "Puja Patidar"

$ git config --global user.email [<input](mailto:yukkutobu777@gmail.com) my email>

1. Created an IntroToProg-Python-Mod09 repository on GitHub.
2. Cloned it locally.  
   https://github.com/ppatidar701/Assignment09.git
3. Went into new local directory and did some initial configuration.  
   git init  
   vi .gitignore
4. Checked status, did commit and then pull/push up to the remote repository.  
   git add -A   
   git status  
   git commit -m "Added .gitignore file"  
   git pull origin master

Verified new local file was pushed up to repository (See figure 5).  
  
Summary

As more complex programs are constructed, Unified Modeling Language is a good tool to lay out the idea, map the code and determine dependencies. There are three different types of diagrams, class (shows parent/child relationship), use case (actor/system explain components) or composition (looks at objects made from classes, composition/aggregation). A combination of a three can be used on a new project as appropriate. I look forward to using those on future assignments.