Introduction to programming

Classes and Object Oriented Programming

Exercise 1 – user_class.py

Following on from all of the advancements with the login programs from previous lessons, your line manager has asked you to create a user object that encapsulates a user's username and password. Similar to the reading material for this week, create a user class that has these two member variables.

Test your class by creating a new user object and try assigning some values to the member variables. Then try printing out the values you have set. Save your program as **user_class.py** in your Week 9 folder.

Exercise 2 – user_class_methods.py

Extend the class you created from exercise 1 to include a method called **authenticate** that takes one argument and checks whether this value is equal to a pre-determined string e.g. 'password'. The method should return **True** or **False** depending on the outcome of the Boolean expression.

Test your class as before by creating a new user object and trying to authenticate the user with the correct password. Save your program as **user_class_methods.py** in your Week 9 folder.

Extension: If you feel comfortable with creating the above method, try incorporating both the encryption and/or serialisation functions from previous weeks in to this class.

Exercise 3 Part A – phone_class.py

You have been asked to write a new operating system that will run on any type of phone hardware (iOS and Android just aren't good enough any more!). To begin the project you decide that you should use classes for each of the different types of phones that your operating system may run on.

First, create a **base** class that incorporates some of the member variables and methods which would be common to all phones. For example:

Member variables	**Methods**
Battery life	Switch on
Signal strength	Switch off
Memory size	Make call

Screen size	

The variables and methods don't need to do anything in particular (for example your **switch on** method could just print out 'Switching on phone…'

Test your base class by creating a new phone object and trying to access some of the methods / member variables. Save your file as **phone_class.py** in your Week 9 folder.

Exercise 3 Part B

Once you have your base class defined, you will need to create classes specific to each phone manufacturer that **inherit** from the base phone class.

Here are some suggestions for types of phones that you could create classes for:

- iPhone 6s
- Galaxy S5
- Lumia 550
- Xperia Z5

With each specific phone class you create, set the member variables to something which is relevant to the phone. For example, for the iPhone 6s class you could set the screen size as **4.7**. The values are not too important, just as long as there is some difference between them.

Test your specific classes by creating new objects and printing out some of the different values. You can either save the classes in individual files or one large file containing all classes in your Week 9.

Exercise 3 Part C

As a final task for your phone base class provide a constructor that initialises all of the base member variables. As a final task for your inherited classes, add a similar constructor and also add any methods which may be relevant to that phone. For example for the iPhone, you may want to include a method called **updateiTunes()**. Again, the methods don't need to do anything specific; printing out a message is fine.

Test your classes a final time to check that the new methods and constructors work OK.

Extension task

Repeat exercise 3 but with a different subject matter such as cars, computers etc. In other words, create a base class for an object that incorporates all of the common

elements and then create sub-classes that inherit from this base class providing an extra level of complexity specific to that particular class.