DAY 3

Exceptions - slide 30

Create exception classes: introduce them into your library system code where you have yet to implement methods but make sure they are specific for the need. One at least should take a single argument.

If you make these classes in a separate file (module) you'll be able to extend them to use in any of your code ☺

Create at least one exception class derived from one of your new ones. Add initialisation arguments, ensure you can access the base class methods and attributes. **HINT**: you'll need to experiment with the 'super' keyword.

Testing - slide 31

Write a test module for your class: use a test class. This can include success cases as well as cases where you expect exceptions to be thrown.

The syntax for this is…

```
@raises(TypeError, ValueError)
def test_raises_type_error():
    raise TypeError("This test passes")
```

Also refer to the code samples in the slideset.
Write tests for functionality which doesn't yet exist but you know needs to be implemented at some stage.

Design Patterns - slides 32-36

Exercise the design patterns and follow their use in the Debugger: they have been implemented for you. Draw sequence diagrams to explain the behaviour for one of them.

Debugging - slide 37

If you're not already familiar with the powers of debugging, use this time to find out about it. Step through your library code to investigate the state of the objects you create and their message passing. You may choose to follow a simple test to start with, and move onto a more complex scenario instigated by your main program.

Use break points; go into the call stack; step in, over, and out; use the evaluation and break condition facilities.