# DAY 1

## OO Design - object identification - slide 6

What are the methods/operations and attributes of the following objects? (a couple of examples of each will suffice). Can they be related strongly or weakly?

* my red pencil
* a great white shark
* small blue cat collar
* international space station
* this course
* Bob, my white cat
* satellite dish
* my Parker ball-point pen
* elephant
* fish
* Bob’s teeth
* mammal

## Python Objects - slide 7

Use the dot notation to investigate the methods on some simple datatypes e.g. integer, string, float. Also try ‘type()’ on one or two of your variables – what do you get?

Create some more complex datatypes and investigate their methods: suggest looking at list, dictionary, Exception, numpy.array. How do Python types differ in how they may be created?

What happens when you ‘print()’ an object? Do you get what you expect?

What happens when you check for equality of two objects?

## Object Oriented Analysis – slide 9

The following paragraph is a user’s requirements, identify the objects and the messages you’d need for the system, using the nouns and verb phrases:

This application will support the operations of a technical library for an R&D organization. This includes the searching for and lending of technical library materials, including books, DVDs, and technical journals. Users will enter their company IDs in order to use the system; and they will enter material ID numbers when checking out and returning items.

Each borrower can be lent up to five items. Each type of library item can be lent for a different period of time (books 4 weeks, journals 2 weeks, DVDs 1 week). If returned after their due date, the library user's organization will be charged a fine, based on the type of item (books 50p/day, journals £1/day, DVDs £2/day).

Materials will be lent to employees with no overdue lendables, fewer than five articles out, and total fines less than £50.

## CRC exercise - slide 10

Using the library lending system, each person should assume a role and then execute a scenario as a group.  
Possible classes are:

* application
* library
* material
* books
* DVDs
* journal
* company
* system
* ID numbers
* item
* borrower
* date
* organization
* employee
* article
* fine

but… there may be duplicates or some outside the scope of the system. Which are actors?

Your scenarios are:

* What happens when Johnny Codewarrior, who has no accrued fines and one outstanding book, not overdue, checks out a book entitled “*Document, Your job depends on it*”?
* Judy Hacker, has fines of £2 outstanding, would like a DVD “*Debugging to music”*, does have a book out (not overdue) and is bringing back an overdue journal.
* Miss Marple wants to borrow a journal “*Sleuthing in C#”* but can’t find it… is it already out?
* Eric Halfbee comes in with a pile of overdue items, but doesn’t know if he has enough money to pay off his debts. If he has, he’d like to borrow a DVD.

## Drawing structure and behaviour – slides 12-14

Sketch out the structure of the classes in the library system and their relationships. Hint: some may be built-in classes.

Pick one of the scenarios above and draw a sequence diagram to represent the behaviour of all the objects involved.

## Python syntax - slide 15

Create an empty PyCharm project ready to code up the Library System – good practice would suggest a folder (or Python Package as appropriate) for each of source code, test code and documentation.

Each person should take responsibility for one class and sketch it out in a skeleton code Python file. The approach is to define your class and its methods and attributes using docstrings and comments and NO ACTUAL CODE to start with.

## Homework

Ensure you can share code with a GitHub account and revise its use if necessary. Hook up your PyCharm project into your git repository cloned from the shared one; add, commit and push your skeleton code file, and pull the ones made by your colleagues.

# DAY 2

## Review & implementation

The GitHub account should contain skeleton code for all the classes and methods outlined during day 1 as a result of the homework. As a team, we should be able to implement a decent chunk of the library system: we should concentrate on one scenario / sequence diagram.

## Naming - slide 20

Carry out a code review and look for good coding practice generally and especially the OO naming conventions used. Do they adhere to standards? Is the code readable?

## Design Patterns - slides 22-25

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.

## Exceptions - slide 26

Create two 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 28

Write test modules for classes in the library system. 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.

## Debugging – following the logic

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.