SI 506 DYU 8

Compared to previous couple of weeks, this week's topics were relatively simple. Different accumulation functions such as map, filter, reduce, zip and list comprehension were introduced. These functions make it possible to perform list operations in a single line, typically. We also studied about Test-Cases, more in-depth than last time. We learnt about the structure of test cases and how to go about defining and calling them. Although test cases may be unappealing for a confident programmer, they help while writing convoluted and long code. You never know when a certain input can destroy the whole code you had beautifully crafted, and you would be left scratching your head trying to debug it.

Advanced Accumulation:

map - to map elements onto a new list
filter - to use specific values from a list to create new list
list comprehension - to create a new list from existing lists
reduce - reduce the list to a different list using operations on two elements cumulatively
zip - to turn elements of lists into a new list of tuples of those elements

Unit tests:

1. import unittest module
2. Define a subclass of unit test.TestCase
3. Define functions and pass the variable self
4. The functions should start with the word 'test'
5. Inside the function, include the test case
    a. self.assert<Equal/True/False/Raises*>()(*parameter to be compared, value to be compared against, string to be displayed*)
6. Invoke the main() function from the unittest module by using unittest.main()

*Use Equal to check for an expected result
Use True/ False to verify a condition
Use Raises to verify for an exception