PAL Consolidation 6 *University of Hertfordshire – Week 8 – December 2015 –*

1 Aim & Objectives

- Review last session
- Why testing is important
- How to test

2 Introduction

In this session we'll be thinking about testing our code and why doing so is important.

Firstly we must consider why we test and the importance of doing so. Below we list a few reasons:

- Correctness – Writing code is easy, enough monkeys and a type writer would be enough to write some syntactically good code. It's something else to write some code that works.
- Robustness – People who use your code may not always use it in the way you think they will. Others may want to use your code for scenarios it wasn't designed to handle. It's up to you as the programmer to try and cover as many of these cases as possible to make your code as robust as possible for future scenarios.
- Error Handling – Sometimes the code you write can't handle some scenarios. If you're asking for a number to loop for example, does it make sense to loop for a minus number of times? Perhaps you ask for a number from the user and the user gives you a string? Maybe you ask the user for a number to divide by and they give you zero, causing a divide by zero error? There are many scenarios that could cause your code to break and it's important that you try to think of them all.
- Fixing Bugs – So on the odd occasion your code contains a bug, how do you know you've fixed it? More importantly, how do you know that you haven't introduced another bug? "*Test Data*" is an important part of programming and should be used to test whether your code works. When new bugs are found, simply add the cases that expose it and test for the bug. We'll cover this later.

Using testing to decide how your code is written is called test driven development or TDD. As a programmer it's usually hard to commit to but ultimately yields the best written code.

3 Exercise - Writing Test Cases

Writing test cases is an art in itself, you need to think of values that will break your code. Some good values are usually found at the extremes and where important decisions are made. Examples of these could be the following:

```
# Divide a value by zero
value = 1 / 0
```

Hopefully you see that we get an error when we run this which is obviously bad, if for example, we were writing a program that monitors a patients health in a hospital program. Numbers that are likely to cause an error need to be tested. Consider the following code:

```
def function(name, numA, numB):
    numC = numA - numB
    numD = numA / (numC - 1)
    return numD
```

Okay, so this code doesn't do anything useful! Consider the possible errors you might get with this code and cases you would want to test. Write them down in the following table and we'll discuss them - one has been given for you! Make sure you don't spend too much time on this exercise if you find it difficult, we'll go over things to look for in class.

Desc.	name	numA	numB	Exp Out
Sunny Day	Somebody	10	5	5 / 2
Divide 0	Who is?	6	5	???
Divide 0	Who is?	-1	-2	???

If you find others, makes sure you bring them up in class!

4 Exercise - Test Driven Development

Next, we'll practice writing a simple piece of code by defining what tests it shall pass first. Like the table you wrote before, write case for the following algorithm:

```
# Take two numbers and add them
# Take the result and times it by 6
# Print the result
```

What do you think is important to test? What's important here is that there are no right or wrong answers. There are certainly better tests than others and ultimately you need to go as far as you're happy with.

Now... Write the algorithm using your test table to test whether you have written it correctly.

5 Exercise - Automated Code Testing

So we know how to test code, but we don't want to keep writing code to test our functions every time. It's tedious. So, as programmers we need to automate this process. Here is an example:

```
def func(num):
    return num + 1

# Assert that the code is running correctly
assert func(1) == 2
assert func(-1) == 0
assert func(0) == 1
```

Change how the function operates, what happens with your assertions? How can you use this to test the code you wrote in the previous exercise automatically?

6 Resources & Further Reading

'*http://homepages.herts.ac.uk/~db12aba/*' – All content from these sessions updated weekly.

'*http://code.org/*' – A good resource testing your programming skills.

'*http://stackoverflow.com/*' – Highly recommended online help for programmers (NOTE: Employers are interested to know whether you're an active member of this site!).

'*http://draw.io*' – A very good, free online drawing tool that exports to many formats, including '`XML`' and '`JPG`'.