PAL Consolidation 11 *University of Hertfordshire – Week 12 – February 2016 –*

Aim & Objectives

- Discuss previous session
- Lists & Arrays
- File handling
- Advanced Tasks
- Look forward to next session

Introduction

In this session we will be discussing multiple topics, including lists, arrays, file handling and how we can start to bring these together for projects. There's both an element of revision of current topics covered in lecture and learning new ways to use these to achieve interesting goals.

Exercise - Lists & Arrays

Firstly, we need to discuss what we mean by these two different types of data:

Arrays

Arrays are extremely simple collections, where usually they are a dedicated piece of memory allocated linearly in RAM. This means you tend to see something like the following:

```
[ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ]
```

For those of you that have studied how memory can be used, you may know that the allocated memory could contain pointers to a larger set of data, or be the actual data itself if it contains an array of integers for example. The usual description is as follows:

- Change contents of array at a given position (usually in the format '`name[i] = new-value`')
- Get the contents from a specific position (usually in the format '`variable = name[i]`')
- Requires some kind of memory allocation so the space is reserved, where the size to be dedicated has to be specified when the array is first declared

Lists

Lists are a different kettle of fish, although they sometimes appear to do the same job as arrays. Lists offer lots of additional methods that usually far extend on the basic capabilities of arrays, usually:

- Appending – Behind the scenes this could be using methods such as linking, copying the array to another location with one more space or many other methods
- Delete value at index – The additional feature this adds to changing its value is moving all the other elements of the array across.
- Concatenating – This is typically putting two arrays together as one, Strings can be looked as an example of where this works, for example: '`variable = "Hello" + " World"`'

Of course there are many other features which we'll experiment with, but the important thing to realise is that we can think of a list as a wrapper for arrays, extending the normal capabilities.

In Python, these are just some of the ways of using lists:

- '`array.append(elem)`' – Adds something to the end of the array.
- '`array.insert(index, elem)`' – Inserts an element at a given position.
- '`array.remove(elem)`' – Removes the first element from the list that matches the one specified (slow).
- '`array.pop(index)`' – Removes an element at the given position of the list and returns it.
- '`array.indexelem`' – Returns the index of the first element found.
- '`array.count(elem)`' – Returns a count of how many times an element appears in the list.
- '`array.reverse()`' – Reverses the elements in the list, so that $a`[0..N] = a[N..0]$ and $a = a`$.

Naturally, nothing replaces reading the documentation at: '*https://docs.python.org/2/tutorial/datastructures.html*'.

"Data Structures"

This phrase is used lightly. In any real application, it's expected that you'll be using a database to store your data and not an in RAM database. Data structures are a complicated subject in their own right, which is why Databases are a module on their own. Relationships between data can be extremely complicated.

Luckily, because Python doesn't worry about type until you break a strict rule, you can nest arrays within arrays to create much more complex structures. For example:

```
array = [["Frodo", 3, [1, 2, 3]], ["Baggins",
6, [4, 5, 6]], ["My Precious", 9, [7, 8, 9]]]
```

As you can see, it's a trivial task to nest arrays within arrays. Please be aware that your code can quickly become extremely complex and these structures often don't replace a database and an object (more detail on these when we come to Java).

Practice

- Demonstrate each of the list methods using the above methods on the data set: '`nums = [1, 1, 2, 3, 5, 8, 13, 21]`'.
- Write a test to make sure each of these work! Print a smiley face on success '`:)`' and an unhappy face on failure '`:(`'. Test that it is possible to fail your test!
- Write a nested list for your following information: '`Name, Student ID, PAL Session Time, [Module Names]`'.

Exercise - File Handling

It may come as no surprise, but file handling is extremely important for holding permanent storage on computers. A long time ago in a galaxy far far away, back way when the "cloud" (a buzz term for a technology that has existed for years) didn't exist, people had to save things on their own devices. Doing them allowed them to access files even when the dial-up (mega slow internet) was switched off.

Today, in the background, this forms a silent but extremely important part of what makes your browser faster (caching), game save your data and where your videos buffer. RAM is only so large, so in order for you to "Netflix and Chill" without the video freezing a buffer must be kept so that sudden drops in the connection don't ruin your videos. An example of where this can't be done would be live streaming, especially noticeable when the internet isn't all that reliable.

The following is not comprehensive, but should start to give you an idea of the capabilities of files. Be sure to check out the many more ways files can be utilized in the up and coming lecture.

Reading a File

To read from a text file, you must open it like the following:

```
file = open("file.txt", "rt")
```

'`"rt"`' does not mean "re-tweet", but instead refers to '`r`'ead '`t`'ext.

Now we have your file opened, we can start to read the contents a line at a time. This can be done with:

```
data = file.readline()
```

You can now read to your hearts content. Be careful to close the file after you have done, to be kind to other users of that file so they can now access it.

```
file.close()
```

Writing a File

First, we must open the file we wish to right. This can be done with the following:

```
file = open("file.txt", "wt")
```

'`"wt"`' in this case is '`w`'rite '`t`'ext.

To write to the file, we can now use '`file.write("Text here")`'. This writes the string to the file stream.

Lastly - and this is important - make sure that you close the file. Different operating systems behave differently to not doing this, but basically you need to safely close it. Not doing so could lock the file (permission and lockout mechanisms), mean that some of the text isn't saved (hasn't flushed) and maybe even corrupt the file (harsh close whilst writing when program terminates and drops the stream).

To get around this, simply send a close command when you have finished with the file:

```
file.close()
```

Practice

To see how much you have absorbed, write an example program to demonstrate the following:

- Take two numbers from the user, multiply them and save them to a file called '`sum.txt`'. Make sure that when you test it, you test it with suitably large numbers. How large a number can you actually multiply in Python?
- Download the text file from '*http://www.cl.cam.ac.uk/~mgk25/ucs/examples/UTF-8-demo.txt*'. Write a search system that checks line by line for a word or phrase written by the user and tells you want line it was on. (Look at '`grep`' or '`awk`' for an example of a programs that are good at doing this).

Exercise - Advanced Task

This is the stretch and reach challenge, where we'll push the limits on what we have learned. Before we can look at the challenge, we first need an algorithm to calculate Pi. We'll use the Gregory-Leibniz series:

$$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} - \ldots$$

This is known as an infinite series, you'll never stop calculating this number! This makes sense, as Pi is infinitely long (as far as we know) as the number is irrational, meaning it can't be expressed simply. As you can see, the top number is 4, the bottom number increments every off number and the + and − iterate every other number, infinitely to correctly produce Pi.

Your challenge, if you choose to accept it, is to create a program from scratch to generate Pi to a specific accuracy requested by the user. Once this is done, you'll want to save this number to a file. If you feel really confident, you may want to figure out a way of saving current progress of number generation.

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`'.