**PAL**
**Program Input**

*University of Hertfordshire*
*– Week 2 – October 2015 –*

## Aim & Objectives

- Re-cap on previous session
- Getting user input
- Working with user input
- Converting an mathematical algorithm to code

## Introduction

### What's the deal with user input?

First things first, what is user input? In this case we're talking about things typed into the keyboard but input in general isn't limited to what the user types. Other than the obvious other inputs, your mouse or touch screen, an extremely frequent use of programs will take string or file input and perform some operation. Your text editor takes an existing file or new file and puts text into it, your music player will take sound files and output them over some kind of speaker - you get the point.

What takes a boring and predictable program into the realm of exciting is essentially it's input, what it does to the data and it's output. We know how to output with our ‘`print()`’ statements, we're about to cover getting user input and then we'll be looking at how we can use this to our advantage.

### What's an algorithm?

An algorithm is essentially a way of writing instructions. Some people shy away from the word ”algorithm” - but it is nothing to be afraid of. Once you understand what they are, you'll know they are very simple. It could be:

$$A = \frac{1}{n}\sum_{x=0}^{n-1} x_i$$

... Or a list instructions:

- Plus ‘`1`’ to ‘`x`’
- If ‘`x`’ is equal to ‘`n`’ then:
  - Divide ‘`A`’ by ‘`n`’ and store the result in ‘`A`’
  - ‘`stop`’
- Add the ‘`i`’th number in the number list ‘`x`’ to ‘`A`’
- Go to the top again

... Or, as I'm sure you are most comfortable with:

“*Take the mean (average) of a list of numbers.*”

You may or may not be pleased to hear that you won't be asked to do the first one, but it's important to remember that math equations and instructions are effectively the same and are not something to be afraid of. Everything you do in both programming and life in general can be written as a list of simple instructions, which in turn makes algorithms very powerful.

## Exercise - Hello Input

Go to the homepages site you used last week (in the “*Resources*” section) and copy the code from ‘*`Resources/Week1/helloinput`*’. Make sure you have the latest version of the idle editor open - the sheet from last week is on the page.

Experiment with this code, change the ‘`"input_text>"`’ for ‘`"What is your name?  "`’ and use the ‘`print()`’ function to display ‘`"Hello NAME_HERE, nice to meet you!"`’.

## Exercise - Hello Decision

A program's ability to make a decision is paramount to it being useful! In the world of Computer Science, we call these decisions ”conditions”, where the result is ”yes” or ”no”, ”true” or ”false”, ”1” or ”0”. The idea is there are only to outputs, you could say the output is binary. An example of this would be our next example in ‘*`Resources/Week1/hellodecisions`*’. Load this and run it. What year do you have to be born in to be cool?

As we look at the program's code, you can see the first line is:

```
year = int(input("When were you born?  "))
```

What this is doing is running the function ‘`input("When were you born?  ")`’, which takes one paramter, text to prompt the user with and returns a string containing what the user typed before pressing enter. Working our way outwards (functions, like math, is evaluated with the inner most brackets first and works outwards) we cast our value to an ‘`int`’, i.e. an ”integer”. A ”cast” is basically a conversion, you're saying make A into a B. Of course this can fail, for example if you run the program and type text you'll break it. An ”integer” or ”int” for short (programmers are usually lazy) is a large number that can be both positive and negative, but may not have a decimal like ‘`0.5`’ for example.

Next, in the code a question is asked. We say ‘`if`’ the next thing is ”true”, then run the code below, ‘`else`’ run the bit below the ‘`else :`’ statement. ‘`year`’ is the number from the user and we're checking whether that number is less than (‘`<`’) or equal to (‘`=`’) ‘`1990`’. What happens when you change the ‘`<`’ to a ‘`>`’ and input your age to the program? What happens when you put something in that is a decimal?

## Exercise - Factorials

Now we will be building a simple factorial program. Factorials are basically all the numbers up to and including that number timesed to that number. To make more sense of it, see the following:

- $1! = 1$
- $2! = 1 * 2$
- $3! = 1 * 2 * 3$
- $4! = 1 * 2 * 3 * 4$

The special case is 0! which is also 1. (It's worth reading up on why this is the case - it's interesting). Using the skills you have acquired, build a new program that does the following:

- Gets user input and casts this value to an integer
- Checks that this number is not negative
- Loops through and:
  - Times the user input with the current sum and saves the result as the new sum
  - Decrements (minus one) the user input
- Print the result

You have 5 cases you can check whether you pass. Writing algorithms with expected input and output code tests the robustness of what you have written, making sure you don't accidentally break something. We'll cover more on testing in later sessions.

Remember this is about you learning to use you resources and gained skills, have a good attempt at this before asking for help.

## Exercise - Fibonacci Numbers (Advanced)

This exercise is purposely vague and difficult. You don't have to complete this if it's too difficult but it is advised you attempt it. Using the skills from the previous exercise, research what "fibbonacci" numbers are and write a program to produce a number of them determined by the input.

## 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://www.mathsisfun.com/numbers/factorial.html*' – Resource for what factorials are.