**ICAPRG301A**

**Lesson 3 Notes**

**Loopy Programming**

In programming we often want to repeat code a number of times. This is called looping and there are two basic ways to do this, using the **for** command and using the **while** command. There are other methods of looping (notably recursion) but these will not be covered in this course.

**For** loops are useful when you know how many times you want to loop the run in advance (ie you want to loop 10 times) or you have a certain set of values that you want to do the same manipulation on (ie I want to change each member of a list).

**While** loops are useful when you don’t know how many times you want to run the loop in advance and you work this out during the loop.

Create a new file (you can use the template if you like) and create the following:

for x in range(10):
 print(x)

This will print out the numbers 0 to 9. It works as follows:
**for** is the command to start the loop
**x** is the variable that is used inside the loop, it will change for each pass through the loop
**in** is the command which says where the x will get its value
**range(**10**)** is a built in command that creates a set of numbers. Range can have three inputs. With just one input it creates a set from 0 to that number, two inputs it creates a set from the starting number to the end number, three inputs it creates a set from the starting number to the end number incremented by the third value.

Try these loops: (not all of them will work)

for x in range(1,10):
for x in range(1,10,2):
for x in range(12,2):
for x in range(12,2,-1):

**Activity 1**

Using a for loop create a function called countdown, which prints out the number from 10 to 1 followed by the word Blast Off!!

Create the following:

x = 0
while x<10:
 print (x)
 x = x + 1

This uses more code and creates the same output as the first for loop.

**Activity 2:**

Now create another function which does the same job as countdown from activity 1 but using a while loop.

*Make sure you keep this file, we will use it in a later Lesson*

In programming it is usually best to use the method that creates the result with least code but is still easy to read. Now you know the basics of for and while loops let’s explore them a bit more. Create the following and run it

x = [1,2,3]
for y in x:
 print (y)

For loops are a very powerful way to move through a list and work with each element. Create the following:

x = [1,2,3]
for y in x:
 y = y\*5
 print (y)
print (x)

Notice that inside the loop we changed the value of y to something else but that did not affect the value of the elements in the original list. (we will spend more time on this at a later lesson, however if you want to read up about this look for articles on variable scope and mutability, which was mentioned in the variables notes)

Create the following:

x = 0
while True:
 print (x)
 x = x + 1
 if x >10:
 break

This is a dangerous loop since there is nothing to stop the loop, we start with while True and nowhere in the loop can we change True to False. If we did not add a stop somewhere the loop would go on forever until you turned off the computer. However there is an if statement which has the command break. This breaks out of the while loop. This means there are two ways to exit a while loop, the condition can change or your code hits the break.

*NB. There are a number of shortcuts in programming. One is based upon the line:*

*x = x + 1*

*The shortcut method for this is*

*x +=1*

*This also works with subtraction and multiplication, and Strings as well as numbers.*

**Activity 5:**

Rewrite your while countdown function so it uses the break command. Finally create the countdown function so it will work on from any number.

**Boolean Operations**

Boolean mathematics is a branch of mathematics which is interested in reducing **expressions** to one of two values, True or False. In computing these are used by two commands **if** and **while** (and a few other commands, we will show some of these soon). In computing this is often shown as

if <expression>: or while <expression>:

where **<expression>** can be anything which is a boolean operation, ie it can be reduced to True or False. (There is also another Boolean condition called None which is used as a place holder for empty and in Python has the Boolean equivalent of False).

The simplest expressions just compare two values such as **x>5**. The expression has a left hand side (x), a comparison operator (>) and a right hand side (5). This expression is evaluated as True or False depending on the value of x. When used in an **if** statement, if x>5: it will result in the block of code being run.

There are a number of other commands which can be used to build these expressions:

**and** is used to combine two separate expressions together and when used it means that BOTH expressions must be True for the whole expression to be True

if x>5 and x<10:

**or**  is used to combine two separate expressions together and when used it means that EITHER expressions must be True for the whole expression to be True

if x<5 or x>10:

**not** is used to reverse an expression (turning it from True to False or False to True)

if not x<5 or x>10:

**in** is used to look inside a list to work out whether the left hand side value is a member of the list

if x in {1,2,3]:

**is** is used to compare variable to their types

if x is None:

There is no real limit to how complex you want to make the Boolean operation, using as many ands, ors etc as you like. However it is best to keep these expressions simple and chain them together rather than trying to include everything in the one expression.

It is possible to combine a number of **if** statements together so that only one is run. The chaining of **if** commands starts with an **if**, the next and subsequent if statements use the command **elif** and the final catch all that is left statements uses **else**.

if x>5:
 do something
elif x=4:
 do something when 4
else:
 do something for all other cases

For more practise on Boolean operations go to <http://codingbat.com/python/Logic-1>

**Operations**

You know the four basic mathematical operations (in mathematics the symbols are called operands): addition (+), subtraction (-), multiplication (\*) and division (/). However there are more, both in Python and in mathematics. The full list of python standard operands can be found here (<http://docs.python.org/library/stdtypes.html>) or through Google searches (<http://www.informit.com/articles/article.aspx?p=459269> or <http://www.tutorialspoint.com/python/python_basic_operators.htm> have good lists). Here are some of the more important one

|  |  |  |
| --- | --- | --- |
| % stands for modulo | 3%2 is 112%4 is 017%5 is 2 | This provides the remainder in a division, really useful when you want to know the mulitples of a number |
| \*\* stands for exponential or power | 2\*\*2 is 4 (same as 22)3\*\*3 is 27 (same as 33) | The number multiplied by itself as many times as the last number |
| != stands for not equal (remember == means equal) | 3!=2 is True4!=4 is False | Used in if and while statements |
| >= greater than or equal  | 3>=1 is True5>=5 is True | Used in if and while statements |

Create these and run it:

for x in range (10):
 if x%2==0:
 print (x)

for x in range (10):
 if x%2!=0:
 print (x)

The first prints out all the even numbers, the second prints out all the odd numbers.

**Assignment 1**

**Fizz Buzz**

This is a primary school children’s game. Children count but do not say any multiple of 3, they say Fizz instead. With multiples of 5 they say Buzz. If the number is a multiple of both 3 and 5 they say FizzBuzz. Create a function which does this.

For example fizzbuzz(20) would output

1,2,fizz,4,buzz,fizz,7,8,fizz,buzz,11,fizz,13,14,fizzbuzz,16,17,fizz,19,buzz

Note this is quite hard to do and has been used in interviews for programming jobs. Competent programmers should be able to write this in 5 minutes.

**Hint**

* you need if statements and modulo to do this
* the order they are interpreted is important.
* Don’t print directly in your function, create a variable to output so it can be changed depending on the if statements. By doing this you are making a function which can be used in a variety of ways (is can be printed on the console or a text file or it can be used to help build other strings or list)

You must also complete the Program Documentation and Feedback sheet and you must use comments in your code.

Good programmers always document their code and the processes around their code. Even at this early stage of your learning you need to get into the habit of documenting your code.

Your code must contain comments. At the very least you should have a header section in your code with your name, student number, assignment number and unit code.