EDULiTO

A Complete Guide to Python Programming Part 2

5. The use of sequences and selection (including Boolean and comparison operators)

6. The use of iteration (count and condition-controlled loops)

Photocopiable Resources

Terms and Conditions of Use

Your school has permission to copy this resource as many times as you require and to use it as you wish within your school/organisation.

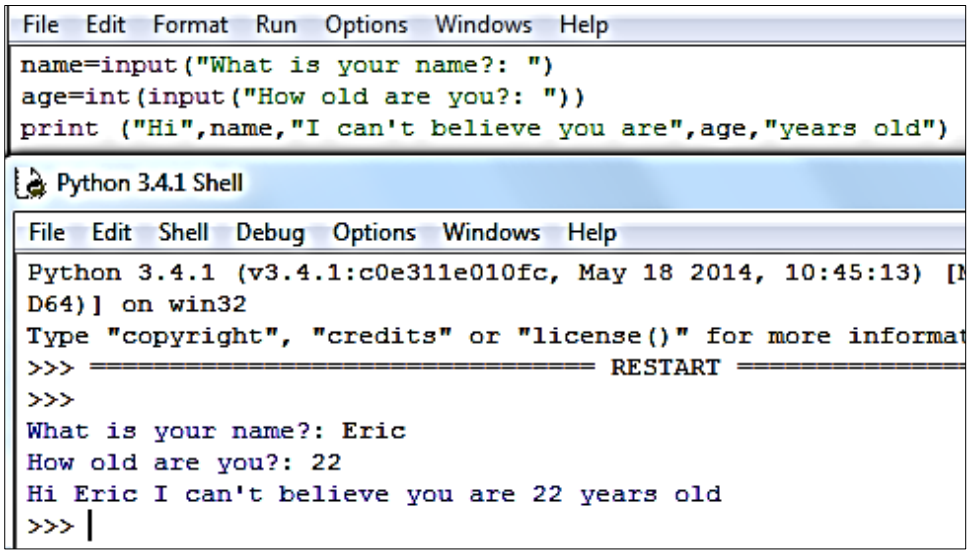
You do not have permission to distribute it as a paper or electronic document to other schools or organisations.

5. The use of sequences and selection (including Boolean and comparison operators)

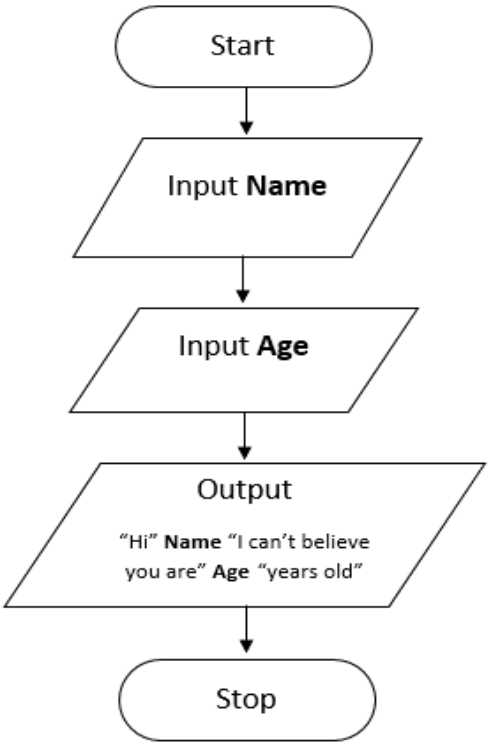
When creating a computer program there are a number of structures that can be created.

Sequences

Some programs are very straightforward. Each instruction follows the instruction before until the program is complete.
Here is an example of a sequence. The first line of code is run, then the second line of code followed by the third line of code.



A flow chart can also be used to represent this sequence:



Comparison (Relational) Operators

These are often used in programming to compare two values and to determine whether one value is less than, greater than or equal to another value.

Here is a list of Comparison (Relational) Operators:

Operator	Meaning	Example
==	Are both values the same	`>>> 3==8` `False`
!=	Are the values not equal	`>>> 3!=8` `True`
<	Is the first value less than the second value?	`>>> 3<8` `True`
>	Is the first value greater than the second value	`>>> 3>8` `False`
<=	Is the first value less than or equal to the second value	`>>> 3<=8` `True`
>=	Is the first value greater than or equal to the second value	`>>> 3>=8` `False`

Boolean (Logical) Operators

Operator	Meaning	Example

AND	Checks whether BOTH conditions are True or False. Therefore BOTH conditions must be True for it be True.	`>>> x=10`
		`>>> x>9 and x<100`
		`True`

OR	Checks whether EITHER condition is True or False. Therefore if EITHER condition is True it is True.	`>>> x=50`
		`>>> x<40 or x>10`
		`True`

NOT	This does the opposite of the existing value.
	True becomes False and False become True
		`>>> x=30`
		`>>> y=10`
		`>>> not(x<y)`
		`True`

Activity 5.1

1. Write a programme using a **sequence** that includes at least 4 lines of code that asks someone their name, their age and their favourite place to go on holiday.

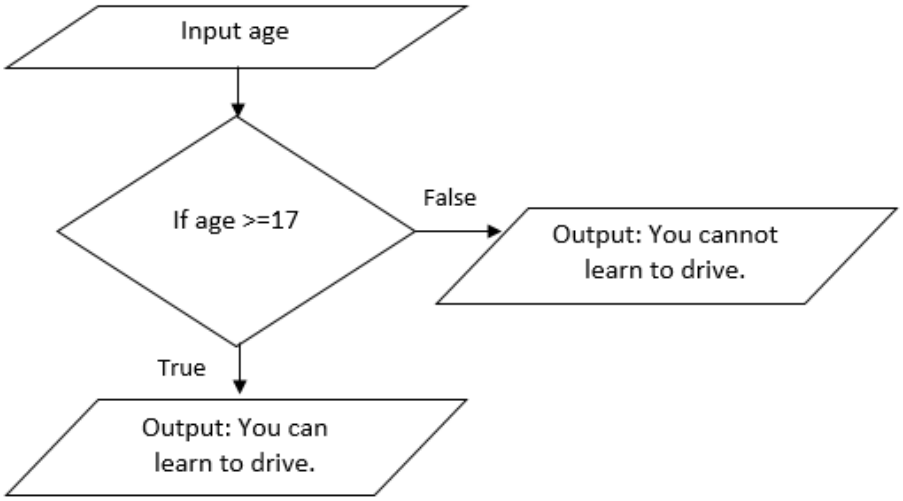
2. Find out whether the following programs output true or false
   a. 7==19
   b. 7!=19
   c. 7>19
   d. 7<19
   e. 19>=7
   f. 19<=7

3. Find out whether the following programs output true or false. In all cases x=10 and y=20.
   a. x==y/2 and x>=9
   b. y/x==2 and y>=10
   c. x+y==20 or y+x==30
   d. x>=11 or y>=31
   e. not(x+y==30) and y==20
   f. not(x+y==30) or y==20

Selection

As mentioned at the beginning of this chapter, some algorithms/programs are very straightforward. Each instruction follows the instruction before until the program is complete. However, sometimes when you are writing an algorithm/program there needs to be a choice e.g.

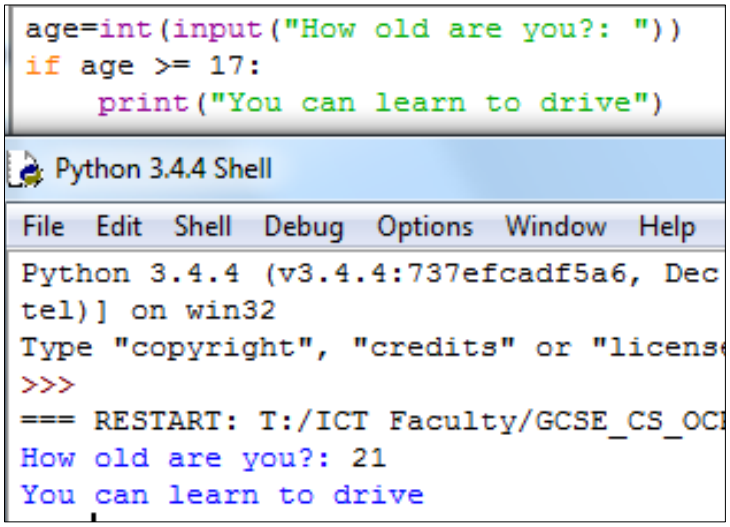
How old are you?
If you are 17 or over you can learn to drive



Simple IF statement

When you use selection there is always a **test** or **condition**. In the example above the test is whether you are 17 or over.

E.g. If the person who is being asked is 21 then the condition is **True**.



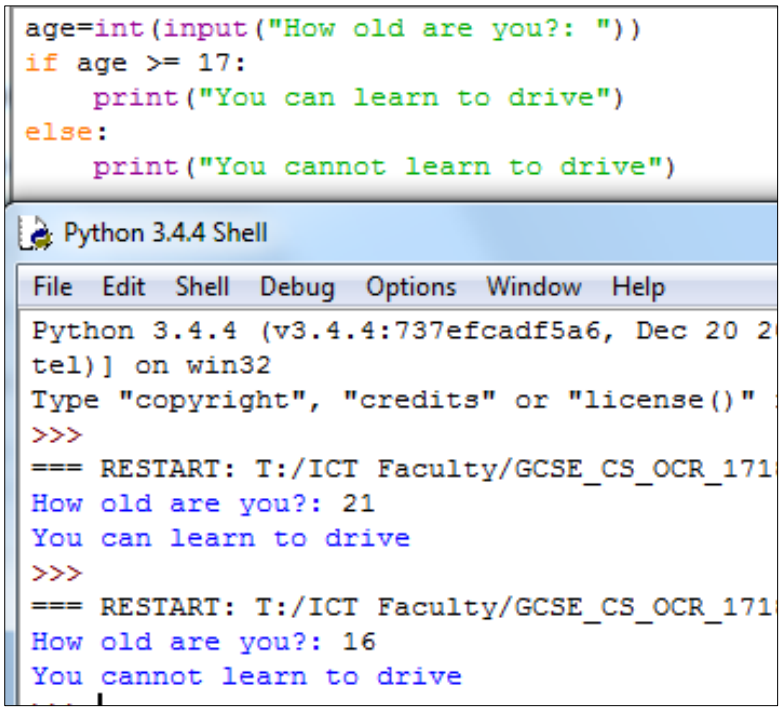
If then Else Statement

The above code shows a simple if that only deals with a condition that is **True**, but usually we need to deal with both possible outcomes from a test or condition. To do this we use **IF** then **ELSE**.

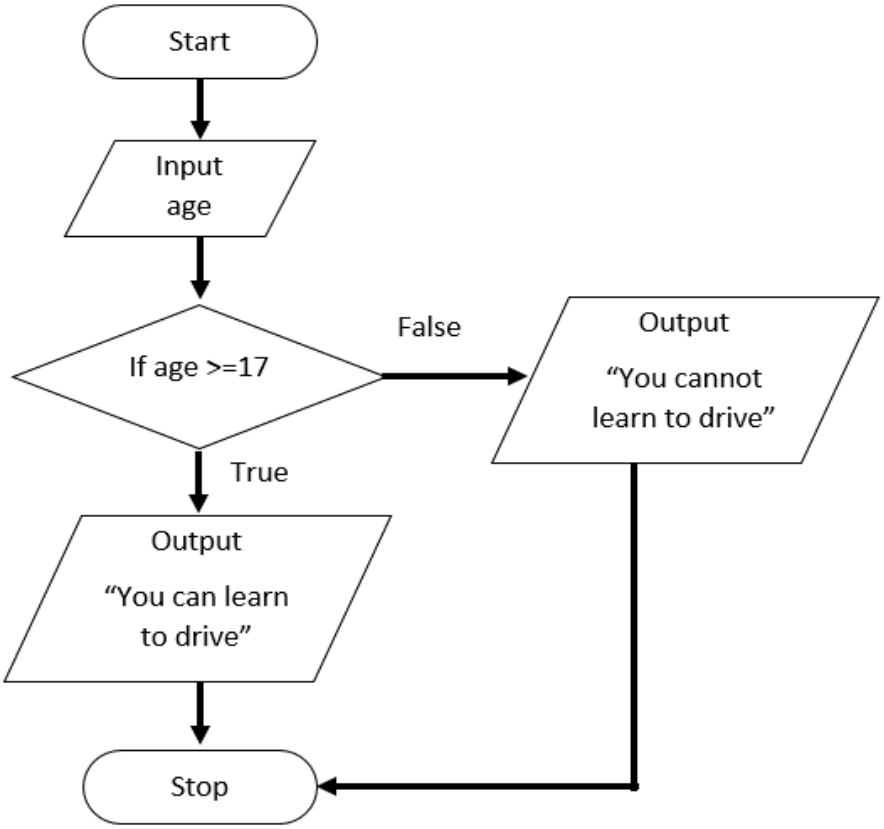
In relation to the example shown above

- IF the person who is being asked is 17 or more then the condition is **True**.
- ELSE the condition is **False**.

In this example the statement displayed depends on whether the statement is **True** or **False**.



This IF then ELSE can also be represented using a flow chart.



Activity 5.2

1. You work for passport control. You have been asked to write a program that sends out the correct form for passport renewal. If the person applying for the passport is under 16 they will need to be sent a "Child" application form. Anyone else applying must be sent the "Adult" application form.

2. You have been asked to write a program that manages security on the personal details stored on a database. Your program must store a particular person's username and their password. When this person tries to login, if their username and password are both entered correctly they will be allowed to continue. However, if either one of these is entered incorrectly they will not be able to proceed and they will receive a message telling them that either their password or username is incorrect.

3. You work for Twitter and you have been asked to create a program that informs the user if they have used too many characters in their message. The program allows you to enter your message and then the program calculates the length of the message in characters. If the message is greater than 140 characters, you get a warning message to say that your message is too long.

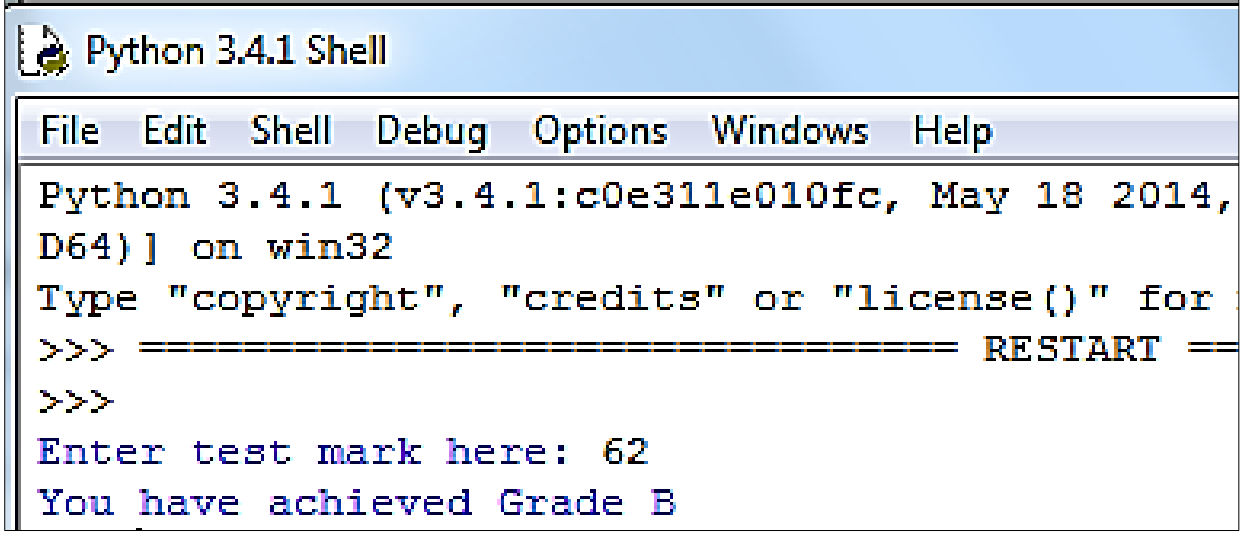
IF and ELIF Statements

If then Else can be used if there is one condition, but sometimes you have more than one condition and in this case you will need to use IF and ELIF statements.
Here is the structure the code should follow.

```
if Condition 1= True:
   Do Code 1
elif  Condition 2= True:
   Do Code 2
else:
   Do Code 3
```

For example if you have a number of test results you can use a program to generate a grade for each student.

```
test_mark=int(input("Enter test mark here: "))
if test_mark >=80:
    print("You have achieved Grade A")
elif test_mark >=60:
    print("You have achieved Grade B")
elif test_mark >=40:
    print("You have achieved Grade C")
else:
    print("You have Failed")
```



This structure can be used for as many conditions as you want.

Nested If statements

Some programs have one if statement nested inside another if statement.
For example:
Ruby owns a car showroom. At the moment she has a special offer but only for cars in her show room that cost £20,000 or more. If the customer has the special discount code, they can get a 5% discount off the cost of the car. However, if the car costs less than £20,000 there is no discount.
In this case a nested if has been used to deal with customers whose car costs more than £20,000. These customers will only get the discount when they have the special discount code. If this is not the case they will not get the discount.

```
car_cost=int(input("How much is the car?: "))
discount_code="XMAS99"
if car_cost >=20000:
    discount_code=input("What is the discount code?: ")
    if discount_code=="XMAS99":
        print("Car cost with discount is £",car_cost-(car_cost/100*5))
    else:
        print("Incorrect code. Car cost is £",car_cost)
else:
    print("Sorry your car needs to cost more than £20,000 to get a discount.")
```

Activity 5.3

1. You have been asked to write a program that converts raw student marks in an exam into 9 to 1 grades. The maximum mark is 160, if you achieve 74 or more, but less than 86 you achieve a grade 4.
Here are the grade boundaries:

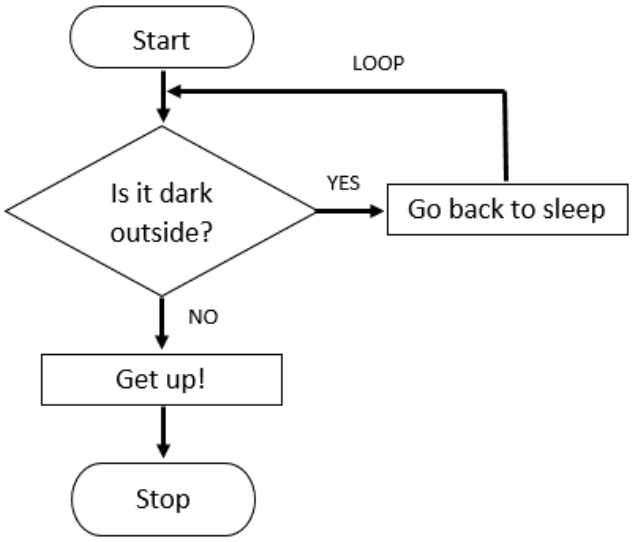
	Max Mark	9	8	7	6	5	4	3	2	1	U
Raw	160	134	123	112	99	86	74	52	30	9	0

Produce a program that asks for your raw exam result and then converts this result into the correct grade.

2. You work for a smart phone reseller. You have a special sale deal but only on android handsets. Customers that purchase an android handset will get a 25% discount each month for the first 6 months on a 12 month contract. The normal cost of the smart phone contract is £20 per month.
Produce a program that asks the customer if they are taking out a contract on an android handset. If they are then the program calculates the monthly cost for the first 6 months and the total cost of the 12 month contract.

6. The use of iteration (count and condition-controlled loops)

Quite often in programming you need to repeat an instruction or process over and over again. This is called Iteration and involves the use of programming constructs called loops.

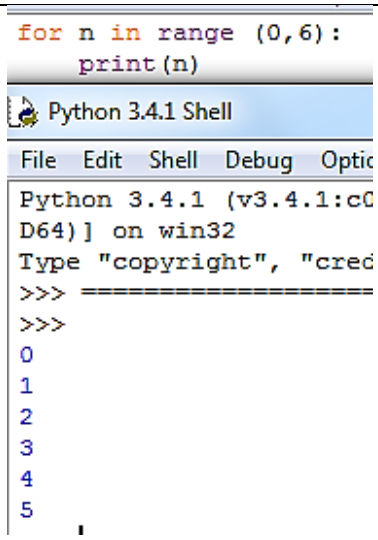
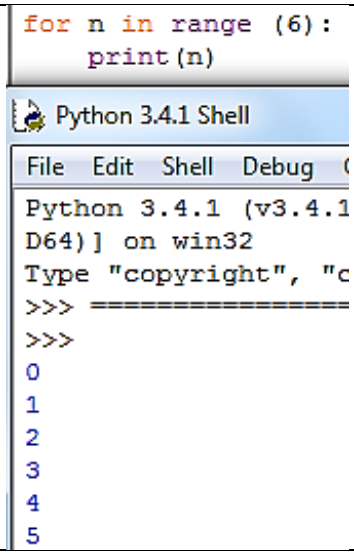


In this example the person repeatedly wakes up in the night and if it is still dark outside they go back to sleep. They continue doing this until it is light outside and at this point they get up.

FOR loops

For loops are examples of definite iteration as when the program starts you know exactly the number of times that you are going to loop. For loops often use the range built-in function.

NB The end of the range means up to but NOT including the final number

Example 1	**Example 2**
In this case the value of n changes as each loop is completed. The first value for n is 0, then the program loops and the second value for n is 1. The final value is 5 as the range is up to, but not including, the final number.	In this case the beginning of the range is not included and so the program starts from 0 and the output is the same as example 1.
	

Example 3	Example 4
In this case the range is from 0 to 5 and the program repeatedly displays the text (loops through 0, 1, 2, 3 and 4).	You can also produce output that lets the numbers increase by 3 each time by adding a third number to the range.

```
for n in range (0,5):
    print("Hello World")
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options
Python 3.4.1 (v3.4.1:c0e3
D64)] on win32
Type "copyright", "credit
>>> ====================
>>>
Hello World
Hello World
Hello World
Hello World
Hello World
```

```
for n in range (0,11,3):
    print(n)
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options
Python 3.4.1 (v3.4.1:c0e3
D64)] on win32
Type "copyright", "credit
>>> ====================
>>>
0
3
6
9
>>>
```

FOR Loops and Strings

We can also iterate a string by using a FOR loop that repeats until the end of the string has been reached.

```
word1="LONDON"
for index in word1:
    print(index)
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug
Python 3.4.1 (v3.4.1
D64)] on win32
Type "copyright", "c
>>> ================
>>>
L
O
N
D
O
N
>>>
```

```
word1="LONDON"
for index in range (0,len(word1)):
    print(word1[index])
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows
Python 3.4.1 (v3.4.1:c0e311e010fc,
D64)] on win32
Type "copyright", "credits" or "li
>>> ================================
>>>
L
O
N
D
O
N
```

Nested FOR Loops

We can position one loop inside another loop. In this example the first loop controls which times table will be displayed and the second loop which is nested within this loop displays the content of the table.

```
for index in range (2,4):
    print("This is the",index,"times table.")
    for times in range (2,13):
        print(times," x ",index," = ",index*times)
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10
D64)] on win32
Type "copyright", "credits" or "license()" for mor
>>> ================================ RESTART =====
>>>
This is the 2 times table.
2  x  2  =  4
3  x  2  =  6
4  x  2  =  8
5  x  2  =  10
6  x  2  =  12
7  x  2  =  14
8  x  2  =  16
9  x  2  =  18
10  x  2  =  20
11  x  2  =  22
12  x  2  =  24
This is the 3 times table.
2  x  3  =  6
3  x  3  =  9
4  x  3  =  12
5  x  3  =  15
6  x  3  =  18
7  x  3  =  21
8  x  3  =  24
9  x  3  =  27
10  x  3  =  30
11  x  3  =  33
12  x  3  =  36
```

Activity 6.1

1. Use a FOR loop to display numbers from 1 to 10.

2. Use a FOR loop to display numbers from 5 to 10

3. Use a FOR loop to display numbers from 2 to 20 in steps of 2

4. Use a FOR Loop to display the word SCHOOL so that one letter from the word appears on each line.

5. Produce a program that uses only 4 lines of code that will display the 2 to 12 times table.

6. Produce a program that will display the 17 times table.

While Loops (Conditional Loops)

This type of loop iterates (repeats) until a condition is met. In the example below the condition is whether x is greater than or equal to 1. Whilst this condition is true the program will keep looping but as soon as x is equal to 1 the program will stop.

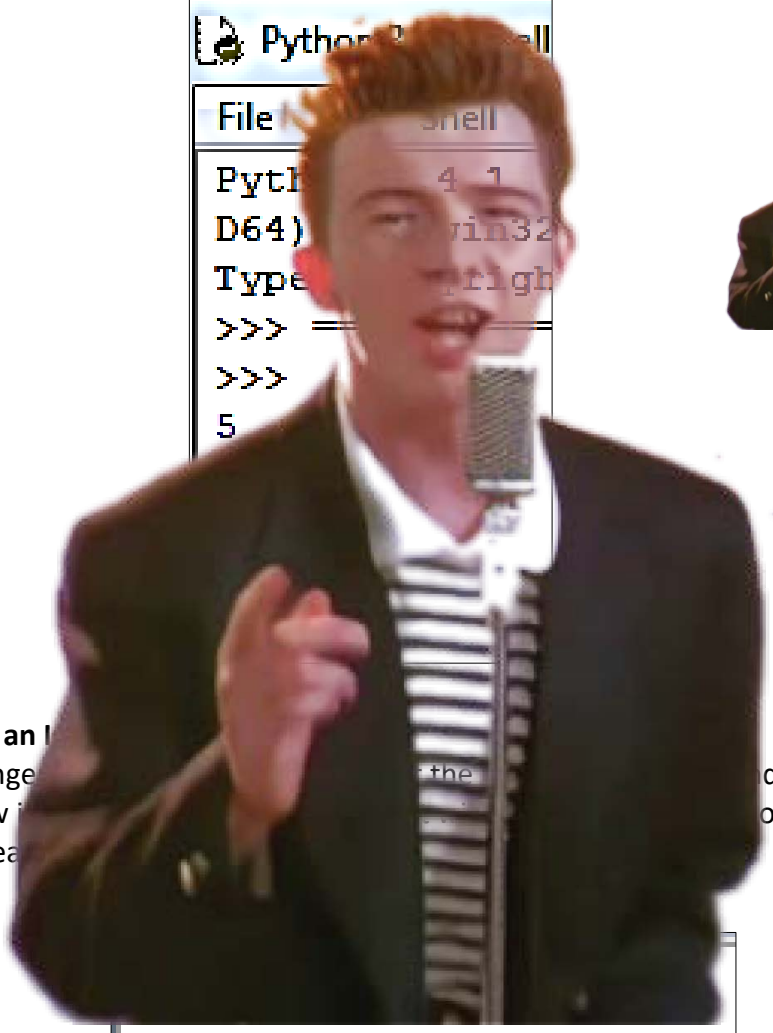
```
x=5
while x >=1:
    print(x)
    x=x-1
```

```
5
```

rick roll

Watch out for an I

One of the dange d **never stop!** The example below oop as x is always going to be grea



```
while x >=1:
    print(x)
    x=x+1
```

Password program using WHILE loop

Here is an example of a password program that uses a while loop.

```
password=""
while password!="1234":
    password=input("Please enter your password: ")
    if password=="1234":
        print("Password correct. You may continue.")
    else:
        print("Incorrect password. Please try again.")
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:45:13
D64)] on win32
Type "copyright", "credits" or "license()" for more infc
>>> ========== RESTART ==========
>>>
Please enter your password: 4321
Incorrect password. Please try again.
Please enter your password: 1234
Password correct. You may continue.
```

Using a while loop to create a menu

Using a while loop is very useful when creating a menu system. The menu will keep repeating, while the condition **carry_on** is true. When you select choice 3 (quit) then **carry_on** becomes false and the while loop stops.

```
question1=""
carry_on=True
while carry_on==True:
    print(" Quiz - M A I N   M E N U")
    print ("""
    1.Instructions
    2.Quiz
    3.Quit
    """)
    choice=input("What would you like to do? (Enter 1, 2 or 3):  ")
    if choice=="1":
        print("Instructions: Once you have read the question enter your answer."
    elif choice=="2":
        print("Questions")
        print("")
        question1=input("What is the capital of France?: ")
        if question1=="Paris":
            print("Correct")
    elif choice=="3":
        print("Thanks for playing. Try again soon")
        carry_on=False
    elif choice !="":
        print("Not a valid choice try again")
```

Using "Break" to leave a loop.

It is sometimes useful to use the word Break when the condition is met so that the block of code stops. It can be used with FOR and WHILE loops

```
for i in range (0,11):
    if i>=4:
        break
    print(i)
```

```
Python 3.4.1 Shell
File  Edit  Shell  Debug  Optior
Python 3.4.1 (v3.4.1:c0e
D64)] on win32
Type "copyright", "credi
>>> ====================
>>>
0
1
2
3
>>>
```

```
x=100
while x >10:
    if x==50:
        break
    print(x)
    x=x-10
```

```
Python 3.4.1 Shell
File  Edit  Shell  De
Python 3.4.1 (v
D64)] on win32
Type "copyright
>>> ==========
>>>
100
90
80
70
60
```

Activity 6.2

1. You have been asked to write a program that manages security on the personal details stored on a database. Your program must store a particular person's username and their password.
a. When this person tries to login, if their username and password are both entered correctly they will be allowed to continue. However, if either one of these is entered incorrectly they will be asked again to enter their username and password again. This will continue until the user name and password are entered correctly.
b. Change the program so that the person gets three chances to enter the correct username and password. After these three chances they get a message to say that they have been locked out of their account.

2 a. You have been asked to create a game. The computer stores a number and then you get a chance to guess the number. You have an unlimited number of guesses to guess the correct number. If you get the number correct you are congratulated but every time you get the number wrong you are asked to guess again.
b. Change the game so that it is a two player game and the players take it in turn to guess the number.
c. Create a menu system for the game. The menu system must include the rules of the game, play the game and quit.