Control Flow

In the programs we have seen till now, there has always been a series of statements faithfully executed by Python in exact top-down order. What if you wanted to change the flow of how it works? For example, you want the program to take some decisions and do different things depending on different situations, such as printing 'Good Morning' or 'Good Evening' depending on the time of the day?

As you might have guessed, this is achieved using control flow statements. There are three control flow statements in Python - `if`, `for` and `while`.

The `if` statement

The `if` statement is used to check a condition: *if* the condition is true, we run a block of statements (called the *if-block*), *else* we process another block of statements (called the *else-block*). The *else* clause is optional.

Example (save as `if.py`):

```
number = 23
guess = int(input('Enter an integer : '))

if guess == number:
    # New block starts here
    print('Congratulations, you guessed it.')
    print('(but you do not win any prizes!)')
    # New block ends here
elif guess < number:
    # Another block
    print('No, it is a little higher than that')
    # You can do whatever you want in a block ...
else:
    print('No, it is a little lower than that')
    # you must have guessed > number to reach here

print('Done')
# This last statement is always executed,
# after the if statement is executed.
```

Output:

```
$ python if.py
Enter an integer : 50
No, it is a little lower than that
Done

$ python if.py
Enter an integer : 22
No, it is a little higher than that
Done

$ python if.py
Enter an integer : 23
Congratulations, you guessed it.
(but you do not win any prizes!)
Done
```

How It Works

In this program, we take guesses from the user and check if it is the number that we have. We set the variable `number` to any integer we want, say `23`. Then, we take the user's guess using the `input()` function. Functions are just reusable pieces of programs. We'll read more about them in the next chapter.

We supply a string to the built-in `input` function which prints it to the screen and waits for input from the user. Once we enter something and press kbd:[enter] key, the `input()` function returns what we entered, as a string. We then convert this string to an integer using `int` and then store it in the variable `guess`. Actually, the `int` is a class but all you need to know right now is that you can use it to convert a string to an integer (assuming the string contains a valid integer in the text).

Next, we compare the guess of the user with the number we have chosen. If they are equal, we print a success message. Notice that we use indentation levels to tell Python which statements belong to which block. This is why indentation is so important in Python. I hope you are sticking to the "consistent indentation" rule. Are you?

Notice how the `if` statement contains a colon at the end - we are indicating to Python that a block of statements follows.

Then, we check if the guess is less than the number, and if so, we inform the user that they must guess a little higher than that. What we have used here is the `elif` clause which actually combines two related `if else-if else` statements into one combined `if-elif-else` statement. This makes the program easier and reduces the amount of indentation required.

The `elif` and `else` statements must also have a colon at the end of the logical line followed by their corresponding block of statements (with proper indentation, of course)

You can have another `if` statement inside the if-block of an `if` statement and so on - this is called a nested `if` statement.

Remember that the `elif` and `else` parts are optional. A minimal valid `if` statement is:

```
if True:
    print('Yes, it is true')
```

After Python has finished executing the complete `if` statement along with the associated `elif` and `else` clauses, it moves on to the next statement in the block containing the `if` statement. In this case, it is the main block (where execution of the program starts), and the next statement is the `print('Done')` statement. After this, Python sees the ends of the program and simply finishes up.

Even though this is a very simple program, I have been pointing out a lot of things that you should notice. All these are pretty straightforward (and surprisingly simple for those of you from C/C++ backgrounds). You will need to become aware of all these things initially, but after some practice you will become comfortable with them, and it will all feel 'natural' to you.

> **Note for C/C++ Programmers**
>
> There is no `switch` statement in Python. You can use an `if..elif..else` statement to do the same thing (and in some cases, use a dictionary to do it quickly)

The while Statement

The `while` statement allows you to repeatedly execute a block of statements as long as a condition is true. A `while` statement is an example of what is called a *looping* statement. A `while` statement can have an optional `else` clause.

Example (save as `while.py`):