Python, an introduction

Samarth Ramesh

Chapter 1

Hello World

Python is a programming language, developed by the Python Software Foundation and released under the PSFL License. I quote Wikipedia:

> 'Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant white space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

For those who have never programmed in their lives, Python offers an easy in of sorts, as the language is very easy to both read and use.

1.1 Python installed?

To check if you have Python already installed, open the command prompt and type

> Python

If Python is installed, the command prompt should reply

> Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
> Type "help", "copyright", "credits" or "license" for more information.
> >>>

However, if you don't see a message like above, but instead see:

'python' is not recognized as an internal or external command,
operable program or batch file.

then, you don't have a copy of Python on your computer. To install Python, go to www.python.org/downloads/ and follow the instructions
Now that all of us have Python installed on our computers, we may begin or exploration of the language.

1.2 First program

Finally, our first program. Our aim is to print the statement "Hello World" (Interestingly, the tradition began with Ritchie & Kerningham's book 'The C Programming Language')

```
print("Hello World!")
```

Would simply return:

```
Hello World!
```

In Python 3, the statement *print()* was used to print a statement, which in the above case was a string which was enclosed in "double quotes".

1.3 More Printing

If we so desire, we may also print multiple strings at the same time using the *print()* function as shown:

```
print("Hello World!","Hoorah!!")
```

which would return

```
Hello World! Hoorah!!
```

The *print()* function inserts a newline after printing the arguments passed to it. So in other words,

```
print("Hello World!","Hoorah!!")
```

and

```
print("Hello World!")
print("Hoorah!!")
```

Would return two different outputs as shown

```
Hello World! Hoorah!!
```

and

```
Hello World!
Hoorah!!
```

respectively.
To force the printing of a newline, we use "\n" in our string as shown.

```
print("Hello World! \n Hoorah!!")
```

The above script returns the same output as:

```
print("Hello World!")
print("Hoorah!!")
```

But the former is preferred for the sake of brevity.
In a similar manner, we may also format output with "\t" to force a new tab, and "\r" to force the cursor to return to the start of the line.
It is useful to know that "\" character is also called an "Escape Character", that converts a special(reserved) symbol into an ordinary character.

1.4 Variable Variables

Now that we have started using the *print()* function, we may further explore it.
the print function can also print the value of variables.

```
somevariable="Hello World, I am a variable"
print(somevariable)
```

returns

```
Hello World, I am a variable
```

The variable can the be reset to another value if needed.
In short, a single 'equals' mark indicates assignment of a variable.
The entire process can be described as:

In Python, all variables are not *declared*, they are *assigned*. This highlights the fact that Python is Dynamically typed.

Dynamically typed means that the type of a variable is checked during run-time, and not during *declaration*.

NB: A language is statically-typed if the type of a variable is known at compile-time instead of at run-time. A more comprehensive list of differences can be found in Appendix A

We may also print multiple variables and/or strings at the same time, as with strings, in a similar fashion. For example:

```
var1="abc"
var2="def"
print(var1,var2,"ghi")
```

Which would return:

```
abc def ghi
```

1.5 Basic Operations

Python, like most programming languages has some basic math in it's Standard Library. The four operators viz. '+' '-' '*' and '/' are used with both variables an constants alike.

NB: A more comprehensive list of mathematical functions(among others) can be found at the end of the book in Appendix B

For example to find and print the product of 256 and 456, we can use:

```
a=256
b=456
c=a*b
print(c)
```

The shell would return:

```
116736
```

As explained before, the *print()* function takes both variables and strings as inputs concatenates the values of both in place.This allows us to easily 'dot the i's and cross the t's '. For example, to calculate the value of 234/2 and print it's value, the following script can be used:

```
num=234
dividend=2
print("The quotient of",num,"and",dividend,"is",num/dividend)
```

would return:

```
117.0
```

In the above script, we used *num/dividend*, Note that the shell returns 117.0 and not 117. In Python, the presence of a decimal is indicative of the number being a /emphfloating point number(It will be covered later on). In short, it cannot be compared to an integer (A number without a decimal point).

1.6 Exercises

1. Write a program to convert *Celsius* to*Fahrenheit* for the given temperatures $t = 45°\text{C}$ and $t = -40°\text{C}$.

2. Write a program that calculates the average of three numbers 189086, 127809, 1567801.

Chapter 2

Control Flow

So far, our programs have dealt with direct statements and have not had to make any decisions. The backbone of most languages is the *if* statement. In the first chapter, while introducing python, I had quoted wikepedia and wrote:

> Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

in other words, every whitespace, that is a group characters representing horizontal or vertical space, for the purpose of improving readability and in the case of Python demarcating blocks of code,is , an indentation, a space or a newline. The use of indentation, while optional in many other languages, is as a result, compulsory in Python. The beginning of a code block is marked by an increase in indentation and the end with the decrease in indentation.

2.1 The *if* statement

The *if*-statement has the following syntax

```
if condition :
    effect
```

Note the use of a colon at the end of the condition. It is what instructs the compiler to look at the next indentation.
For example, the following script tests if two given numbers are equal or not.

```
a=25
b=30
if a==b:
    print("a=b")
```

However, the *if* conditional also includes support for *else* with a similiar syntax.
So a program to check if a number is greater than, lesser than or equal to another number, we can write:

```
a=17890 b=12908 if a>b:
        print("a   b")
else:
        if a<b:
                print("a<b")
else:
        print("a=b")
```

For the sake of brevity, Python also supports elif statement that allow us to say "if the above statement was false, then see if this statement is true" thus, the above script can be rewritten as

```
if a>b:
        print(a> b)
elif a<b:
        print(a< b)
else:
        print(a= b)
```

2.2 *For* loops

The *for* loop is widely regarded as the workhorse of most languages and is used to execute a certain block of code multiple times.
Like the *if* conditional, the *for* loop also needs a new level of indentation. The loop, uses the following syntax

```
for somevariable in iterable : #or sequence
        do this
after exiting do this
```

The 'something' initially is generally range(somethingelse), where 'somethingelse' is a number. The range function by default starts counting from 0. so a script to calculate the various values in fahrenhiet of temperatures in celsius starting from 10° to 100° with increments of 10 would be

```
x=0         #initialzes a variable x to 0 to be safe
for x in range(9):          #we use 10, while 0-10 includes 11 numbers, the range excludes the last number
```

```
celsius=(10*(x+1))        #getting the celsius from numbers
fahr=((1.8*celsius)+32)        #converting celsius to fahr
print(celsius, "degrees C =" , fahr , "degrees F")
```

The script if entered from the command prompt would take time to enter in one by one, and often tends to be more trouble than it it worth. However, if you used some text editor or ide, then the program would have run.
So far, we have dealt only with simple and basic programs that did not have many loops and conditionals. however, as our journey progresses, the complexity will only grow. Until now, we have started python from the command prompt using the *python* command. Henceforth, it is recommened that you use an IDE or something like notepad++ (for windows) or gedit/kate/atom, save the script as something.py, then from the command prompt, cd to the working directory and finally pass the filename as an argument to python.
in other words, follow the given steps:

1. First, open the command prompt by searching for 'cmd'
2. Then, enter the command 'mkdir python'
3. And next enter the command 'cd python'
4. Then in notepad++, type and write your script, and save it into the folder you just created
5. Then, if the name of the script is say script.py, the type "python script.py"

2.3 Excercises

1. Create a program to write the multiplication tables of the first 10 numbers, upto 10 (that is, the program must return the times tables from x*1 to x*10, for all x between 1 and 10, both included)

2. Create a general program to calculate the average of 'n' numbers, where every third natural number upto 25 is taken.(that is, if later on the limit increases, the program remains the same)

Chapter 3

While Loops

In this chapter, we shall discuss the bread and butter of control structure, the *while* loop.

3.1 While Loops

While loops are the bread and butter of infinite loops, most commonly seen in games and the like, where valid user input is needed.

3.1.1 Syntax

Like the for loop, and the conditional, the while loop too creates a new code block. The syntax for a while loop is as follows:

```
while (condition):
    ⟨your code here⟩
```

Note the use of *:* and an additional indentation.

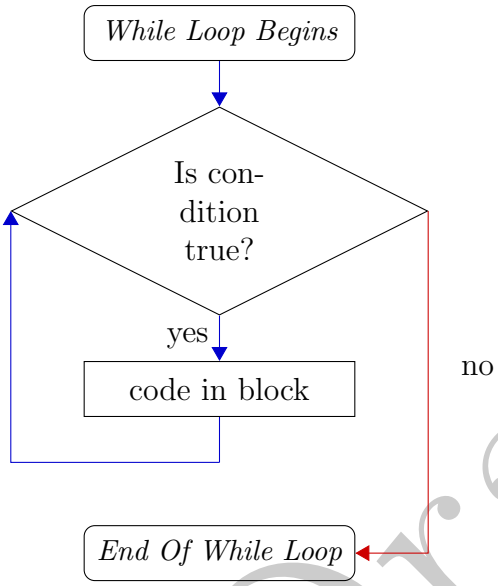
3.1.2 Usage

To calculate the mean of the first 100 numbers, we can use

```
x = 1
sum = 0
while x < 101: #note the use of 101, rather than <= 100
    sum = sum + x
    x = x + 1
print(sum/100)
```

3.1.3 Explanation

The while loop simply put executes the code within it's code block until such a time that the condition evaluates to False. (Note, if the condition evaluates to false at the starting, the code within never gets executed. To ensure that the code executes atleast once, use *do... while*)
Below, you can see a pictographic representation of the while loop



3.1.4 Examples

1. Convert all integral values of celsius into fahrenhiet between 0 and 100 both included.

   We can solve this problem in many different ways. Since we are discussing *while* loops, we will use it to solve our problem.

```
x = 0
FtoCFactor = 1/1.8
while (x < 101):
    print('The Fahranhiet equivalent of', x, 'is', ((x-32)/FtoCFactor))
    x = x+1 # Note, we can also use x += 1
```

   A note about the 5^{th} line, We used $x = x + 1$ because Without it, the while loop will never terminate (because x will never increase)

2. Print out all the possible IPv4 addresses of the form 192.168.0.x and 192.168.1.y

   IPv4 addresses are basically 4 numbers between 0 and 255 (both included) seperated by dots. Some famous IPv4 addresses include 127.0.0.1 (the loopback address), 8.8.8.8 (google public dns) and 192.168.0.1 (a standard home network router IP address)

   We can as always solve this problem in multiple different ways. Since we are asked to find *two* subnets that are very close to each other, we can use a while loop for the inner prefix. Similiarly, we can use a *nested while loop* for the outer prefix.

```
net = '192.168.'
counter1 = 0
counter2 = 0
while (counter1 < 2):
    while (counter2 < 256):
        print(net + counter1 + '.' + counter2)
        counter2 += 1
    counter1 += 1
```

   And there you have a nested while loop, the bread and butter of many algorithms.

3.1.5 Excercises

1. Write a script to calculate the n^{th} power of 8 (for the purposes of testing, use $n = 9$), using a while loop

2. Write a script to calculate the factorial of 15. (n factorial $= 1 \times 2 \times 3...(n-1) \times (n)$)