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 whitespace. 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, 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

lorem ipsum dor
must add windows output

If Python is not installed, go to www.Python.org/downloads/
Now that all of us have Python installed on our computrs, 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')
In Python 3, the statement *print()* is used to print a statement, with statement in *either* single *or* double quotes.in other words,

1.3 Variable Variables

Now that we have started working with 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

```
Must add output
```

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

1.4 Arithmetic

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

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

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

or

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

or

```
print(256*456)
```

While all three of them return the same result, all three of them have diifer-ent uses. The first is used while 'Debugging', the second in most programs where brevity is needed and the third while using Python as a calculator. The *print()* function takes both variables and strings as inputs and can concatenate the values of both inplace. thus, to give more clarity to the end user, a statement may also be included. for example, to calculate the value of 234*235 and give a statement about the same, the following two scripts may be used

```
a=256
b=456
print("The product of the given variables is" , a*b)
```

or

```
a=256
b=456
c="The product of the given variables is"
print(c,a*b)
```

Multiple such variables may be referred so, a similiar but better program can also list the values that are being multiplied by name, ie.

```
a=256
b=456
print("The product of " , a ,",", b ,"is" , a*b)
```

1.5 Excercise

1. Write a program to convert *Celsius* to*Fahrenhiet* for a given temperature $t = 45°\text{C}$.

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 need for the entire document to be both saved for later use and the need to execute the same script over and over in a speedy manner will only grow. Until now, we have started python from the command prompt (Shell for unix users) using the *python* command. Henceforth, it is recommened that you use an IDE or something like notepad++ (for windows), 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. Type your script in an IDE or a text processor and save it with the .py extension

2. In the command prompt, *cd* to your working directory

3. 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)