FOUR WEEKS SUMMER INSTITUTIONAL TRAINING REPORT

On

PYTHON BASICS

at

COURSERA

SUBMITTED IN PARTIAL FULFILLMENT OF THE EWQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer science and Engineering)

Submitted By :

Amritpal Singh

University Roll No.2001192

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BABA BANDA SINGH BAHADUR ENGINEERING COLLLEGE
FATEHGARH SAHIB

INDEX

Chapter 1

INTRODUCTION

1.1 Python

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory managementand has a large and comprehensive standard library. Python interpreters are available forinstallation on many operating systems, allowing Python code execution on a wide variety of systems.

1.2 Scripting Language

A scripting or script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution oftasks that could alternatively be executed one-by-one by a human operator.

Scripting languages are often interpreted (rather than compiled). Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting includesoftware applications, web pages within a web browser, the shells of operating systems (OS), embedded systems, as well as numerous games.

A scripting language can be viewed as a domain-specific language for a particular environment; in the case of scripting an application, this is also known as an **extension language**. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction, oras control languages.

1.3 Object Oriented Programming Language

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known asattributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and oftenmodify the data fields of the object with which they are associated (objects have a notion of "this" or "self").
In OO programming, computer programs are designed by making them out of objectsthat interact with one another. There is significant diversity in objectoriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.

1.4 History

Python was conceived in the late 1980s, and its implementation was started in December1989 by Guido van Rossum at CWI in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given tohim by the Python community, benevolent dictator for life (BDFL).

Chapter 2

DATA TYPES

Data types determine whether an object can do something, or whether it just would not make sense. Other programming languages often determine whether an operation makessense for an object by making sure the object
can never be stored somewhere where the operation will be performed on the object (thistype system is called static typing). Python does not do that.
Instead it stores the type of an object with the object, and checks when the operation is performed whether that operation makes sense for that object (this is called dynamic typing).

2.1 Python has many native data types. Here are the important ones:

Booleans are either True or False.

Numbers can be integers (1 and 2), floats (1.1 and 1.2), fractions (1/2 and 2/3).

Strings are sequences of Unicode characters, e.g. an HTML document.

Bytes and byte arrays, e.g. a JPEG image file.

Lists are ordered sequences of values.

Tuples are ordered, immutable sequences of values.

Sets are unordered bags of values.

2.2 Variables

Variables are nothing but reserved memory locations to store values. This means thatwhen you create a variable you reserve some space in memory.
Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Ex:

```
counter = 100 # An integer assignment
miles = 1000.0 # A floating point
name = "John" # A string
```

2.3 String

In programming terms, we usually call text a string. When you think of a string as acollection of letters, the term makes sense.
All the letters, numbers, and symbols in this book could be a string.
For that matter, your name could be a string, and so could your address.

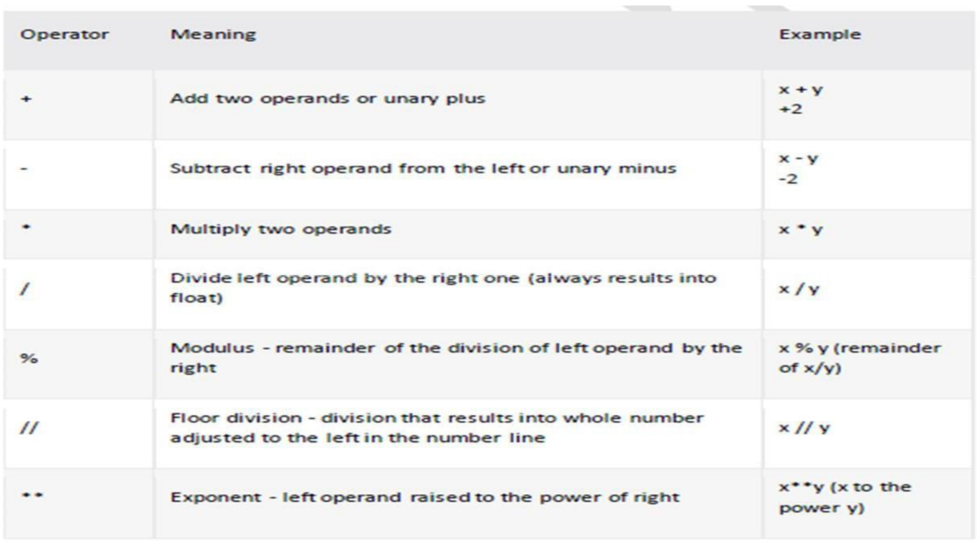
2.4 Creating Strings

In Python, we create a string by putting quotes around text. For example, wecould take our otherwise useless

- "hello"+"world" "helloworld" # concatenation
- "hello"*3 "hellohellohello" # repetition
- "hello"[0] "h" # indexing
- "hello"[-1] "o" # (from end)
- "hello"[1:4] "ell" # slicing
- len("hello") 5 # size
- "hello" < "jello" 1 # comparison
- "e" in "hello" 1 # search

2.5 Python Operator

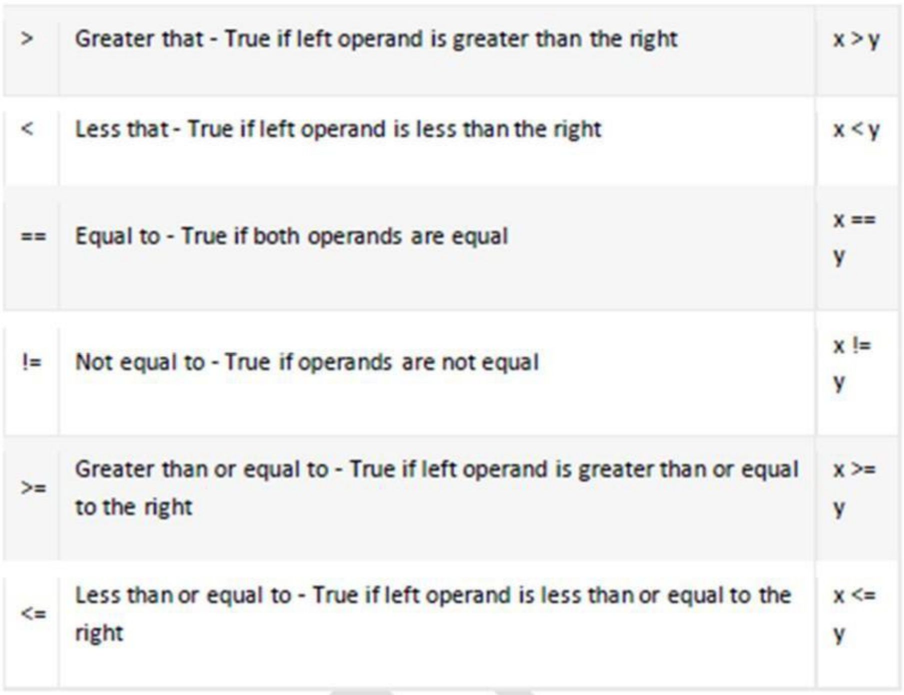
2.5.1 Arithmetic Operator



Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Table 2.1 Arithmatic Operators

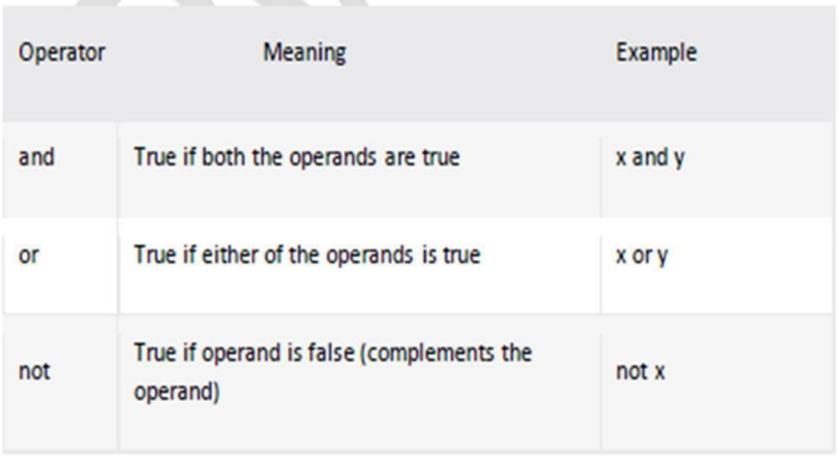
2.5.2 Comparison Operator



>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Table 2.2 Comparision Operators

2.5.3 Logical Operator



Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Chapter 3

TUPLES

Creating a tuple is as simple as putting different comma-separated values. Optionallyyou can put these comma-separated values between parentheses also.
For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 ); tup3 = "a", "b", "c", "d";
```

3.1 Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index orindices to obtain value available at that index. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 ); print "tup1[0]: ",
tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

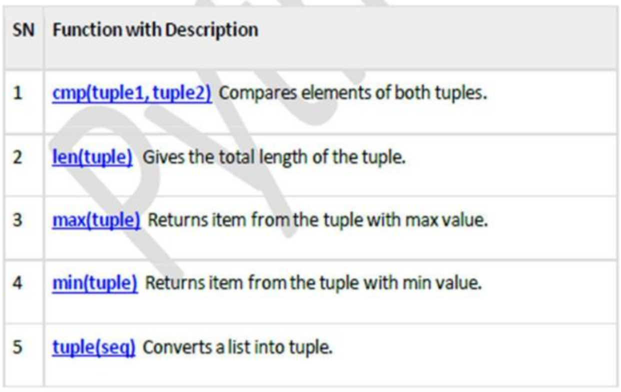
When the above code is executed, it produces the following result –tup1[0]: physics
tup2[1:5]: [2, 3, 4, 5]

3.2 Basic Tuples Operations

Tuples respond to the + and * operators much like strings; they mean concatenation andrepetition here too, except that the result is a new tuple,
not a string. In fact, tuples respond to all of the general sequence operations weused on strings in the prior topics.

3.3 Built-in Tuple Functions

Python includes the following tuple functions –



SN	Function with Description
1	cmp(tuple1, tuple2) Compares elements of both tuples.
2	len(tuple) Gives the total length of the tuple.
3	max(tuple) Returns item from the tuple with max value.
4	min(tuple) Returns item from the tuple with min value.
5	tuple(seq) Converts a list into tuple.

Table 3.1 Built in tulip functions

LIST

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list isthat items in a list need not be of the same type. Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ]; list3 = ["a", "b", "c","d"];
```

Similar to string indices, list indices start at 0, and lists can be sliced,concatenated and so on.

3.4 Accessing Values in Lists:

To access values in lists, use the square brackets for slicing along with the index orindices to obtain value available at that index. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ]; print "list1[0]: ",list1[0]
print "list2[1:5]: ", list2[1:5]
```

Output:

```
list1[0]: physicslist2[1:5]:
[2, 3, 4, 5]
```

Update:

```
list = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 : "
print list[2]
list[2] = 2001;
print "New value available at index 2 : "
print list[2]
```

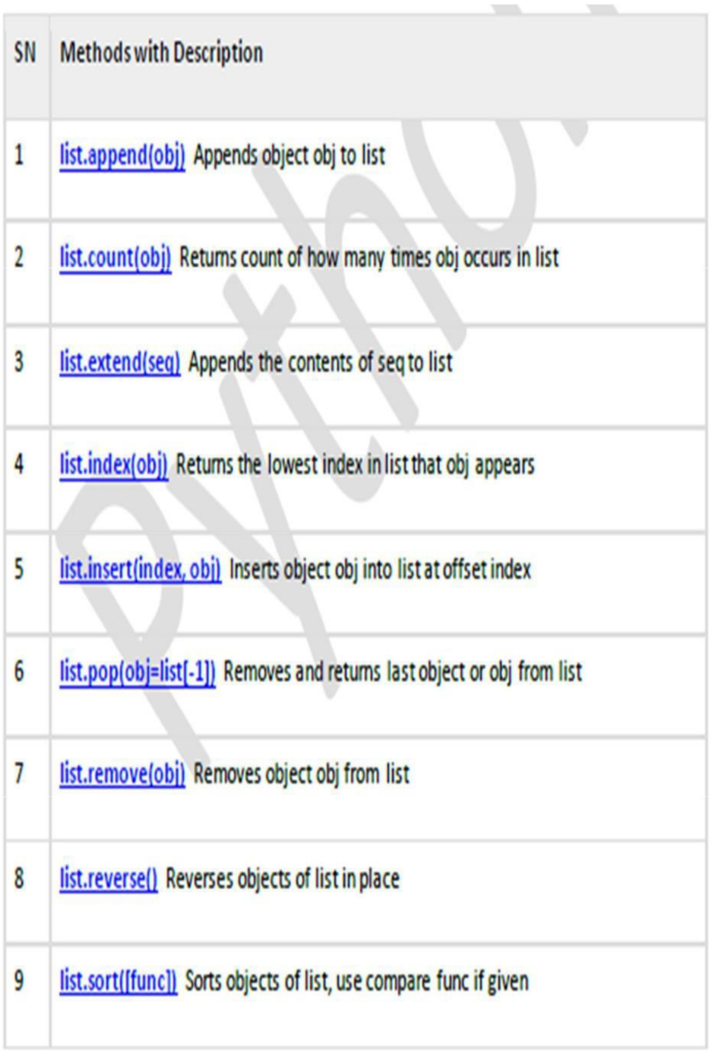
Output:

```
Value available at index 2 : 1997New value
available at index 2 : 2001
```

Delete:

```
list1 = ['physics', 'chemistry', 1997, 2000];
print list1
del list1[2];
print "After deleting value at index 2 : "
print list1
['physics', 'chemistry', 1997, 2000]
```

3.4 Python includes following list methods



SN	Methods with Description
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list.extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj=list[-1]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare func if given

Chapter 4

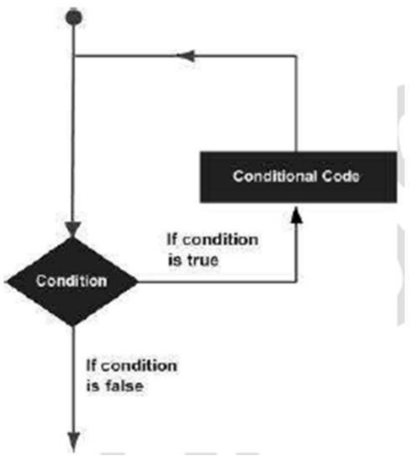
LOOPS

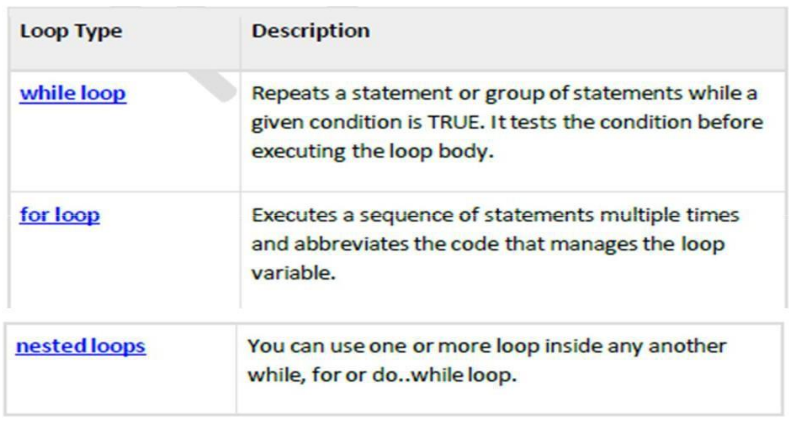
4.1 Loops

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statementsmultiple times. The following diagram illustrates a loop statement –

Python programming language provides following types of loops to handlelooping requirements.





Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
nested loops	You can use one or more loop inside any another while, for or do..while loop.

Fig. 4,1 Conditions and Loop types

Example:

For Loop:

```
>>>  for mynum in [1, 2, 3, 4, 5]:
print "Hello", mynum
Hello
1
Hello
2
Hello
3
Hello
4
Hello
5
```

While Loop:

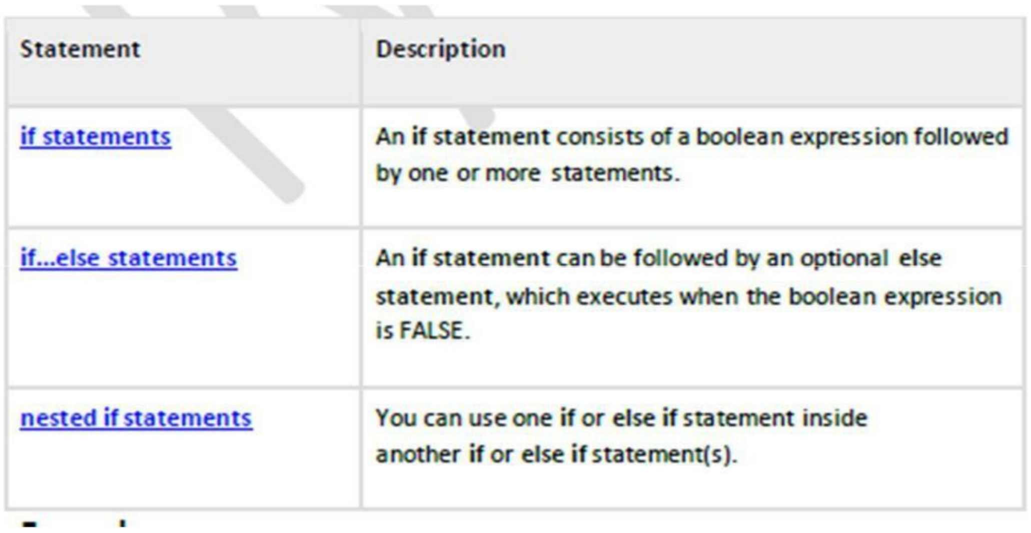
```
>>>   count = 0
>>>  while  (count  <  4):  print
'The  count  is:',  count  count  =
count + 1
The  count  is:  0
The  count  is:  1
The  count  is:  2
The count is: 3
```

4.2 Conditional Statements:

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE orFALSE otherwise.

4.3 Statements

Python programming language provides following types of decision makingstatements. Click the following links to check their detail.



Statement	Description
if statements	An if statement consists of a boolean expression followed by one or more statements.
if...else statements	An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.
nested if statements	You can use one if or else if statement inside another if or else if statement(s).

Fig. 4.3.1 Statements

Example:

If Statement:

```
>>> state = “Texas”
>>> if state == “Texas”:
print “TX
TX
```

If...Else Statement:

```
>>> if state == “Texas”
print “TX”
else:
print “[inferior state]”
```

If...Else...If Statement:

```
>>> if name == “Paige”
print “Hi Paige!”
elif name == “Walker”:
print “Hi Walker!”
else:
print “Imposter!”
```

4.4 Function

Function blocks begin with the keyword **def** followed by the function name andparentheses (( )).

Any input parameters or arguments should be placed within these parentheses.You can also define parameters inside these parentheses. The first statement ofa function can be an optional statement - the documentation string of the function.
The code block within every function starts with a colon (:) and is indented. The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax:

```
def functionname( parameters ):
"function_docstring"
function_suite
return [expression]
```

Example:

- def printme(str):
"This prints a passed string into this function"
print str
return

- *# Function definition is here*
def printme(str):
"This prints a passed string into this function" print
str
return;

# Now you can call printme function printme("I'm first call to user defined function!") printme("Again second call to the same function")

References

a) Training report.
b) Python Basics
c) www.coursera.org