

## Lab 12: Python (Basic)

# Python

**Python** is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## **Python:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## **Applications of Python**

- Easy-to-learn
- Easy-to-read
- Easy-to-maintain
- A broad standard library
- Interactive Mode
- Portable
- Extendable
- Databases
- GUI Programming
- Scalable

## **Python Syntax**

Python syntax can be executed by writing directly in the Command Line:

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

Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

## **Python Indentation**

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

## **Example**

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

## Example

Syntax Error:

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

the number of spaces is up to you as a programmer, but it has to be at least one.

## Example

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

## Example

Syntax Error:

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

```
    print("Five is greater than two!")
```

Python Variables

In Python, variables are created when you assign a value to it:

## Example

Variables in Python:

```
x = 5
```

```
y = "Hello, World!"
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

## Example

Comments in Python:

```
#This is a comment.
```

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

## Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type: `str`

Numeric Types: `int`, `float`, `complex`

Sequence Types: `list`, `tuple`, `range`

Mapping Type: `dict`

Set Types: `set`, `frozenset`

Boolean Type: `Bool`

Binary Types: `bytes`, `bytearray`, `memoryview`

## Getting the Data Type

You can get the data type of any object by using the `type()` function:

## Example

Print the data type of the variable x:

```
x = 5
```

```
print(type(x))
```

## Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

Example	Data Type
x = "Hello World"	str
x = 20	int

<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset
<code>x = True</code>	bool
<code>x = b"Hello"</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

Equals: `a == b`

Not Equals: `a != b`

Less than: `a < b`

Less than or equal to: `a <= b`

Greater than: `a > b`

Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

### Example

If statement:

```
a = 33
```

```
b = 200
```

```
if b > a:
```

```
    print("b is greater than a")
```

In this example we use two variables, `a` and `b`, which are used as part of the if statement to test whether `b` is greater than `a`. As `a` is 33, and `b` is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

Elif

The `elif` keyword is python's way of saying "if the previous conditions were not true, then try this condition".

### Example

```
a = 33
```

```
b = 33
```

```
if b > a:
```

```
    print("b is greater than a")
```

```
elif a == b:
```

```
    print("a and b are equal")
```

In this example `a` is equal to `b`, so the first condition is not true, but the `elif` condition is true, so we print to screen that "a and b are equal".

Else

The `else` keyword catches anything which isn't caught by the preceding conditions.

## Example

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

## Tasks:

1. **Implement all the example given in the manual**
2. Write a Python program which accepts the radius of a circle from the user and compute the area
3. Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.

*Sample data :* 3, 5, 7, 23

*Output :*

List : ['3', ' 5', ' 7', ' 23']

Tuple : ('3', ' 5', ' 7', ' 23')

4. Write a code to find whether the number entered by user is positive or negative.

## Lab 13: Python (Advance)

### Rubrics for Lab 12 and Lab 13

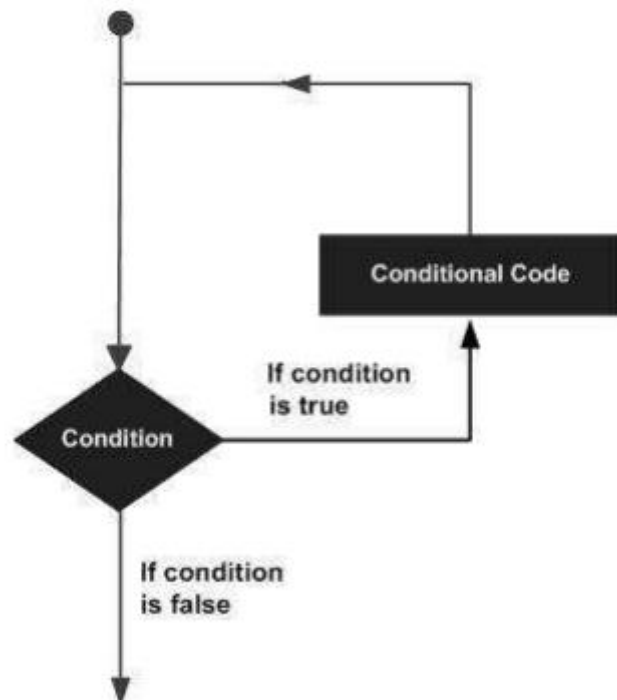
Task	0	1	2	3	4
Python Programming (Basic)	Student is not able to use software and write code	Student able to learn basic concept and syntax of language (Datatype, input, output etc.)	Student able to make correct logic	Coding tasks are partially correct	Output is according to the given query
Python Programming (Advance)	Student is not able to use software and write basic code	Student able to learn both basic and advance concept (Syntax of datatype, input, output. If. Else if, loops etc.)	Student able to make correct logic using advance statement.	Coding tasks are partially correct	Output is according to the given query

#### Loops:

In general, statements are executed sequentially- The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times. Programming languages provide various control structures that allow more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times.

The following diagram illustrates a loop statement.



Python programming language provides the following types of loops to handle looping requirements.

**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, or for loop.

### **while Loop Statements**

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

#### **Syntax**

The syntax of a while loop in Python programming language is while expression:

**statement(s):** statement(s) may be a single statement or a block of statements with uniform indent. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

#### **Example**

```
#!/usr/bin/python3

count = 0
while (count < 9):
    print ('The count is:', count)
    count = count + 1

print ("Good bye!")
```

When the above code is executed, it produces the following result-

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
```

```
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```



**The Infinite Loop** A loop becomes infinite loop if a condition never becomes FALSE. You must be cautious when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop. An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as and when required.

```
#!/usr/bin/python3
var = 1
while var == 1 : # This constructs an infinite loop
    num = int(input("Enter a number :"))
    print ("You entered: ", num)
print ("Good bye!")
```

When the above code is executed, it produces the following result-

```
Enter a number :20
You entered: 20
Enter a number :29
You entered: 29
Enter a number :3
You entered: 3
Enter a number :11
You entered: 11
Enter a number :22
You entered: 22
Enter a number :Traceback (most recent call last):
  File "examples\test.py", line 5, in
    num = int(input("Enter a number :"))
KeyboardInterrupt
```

The above example goes in an infinite loop and you need to use CTRL+C to exit the program.

### Using else Statement with Loops

Python supports having an else statement associated with a loop statement.

□ If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.

□ If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

The following example illustrates the combination of an else statement with a while statement that prints a number as long as it is less than 5, otherwise the else statement gets executed.

```
#!/usr/bin/python3
count = 0
while count < 5:
    print (count, " is less than 5")
    count = count + 1
else:
    print (count, " is not less than 5")
```

When the above code is executed, it produces the following result-

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
```

## for Loop Statements

The for statement in Python has the ability to iterate over the items of any sequence, such as a list or a string.

### Syntax

for iterating\_var in sequence:

statements(s)

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating\_var. Next, the statements block is executed. Each item in the list is assigned to iterating\_var, and the statement(s) block is executed until the entire sequence is exhausted.

## The range() function

The built-in function range() is the right function to iterate over a sequence of numbers. It generates an iterator of arithmetic progressions.

```
>>> range(5)
range(0, 5)
>>> list(range(5))
[0, 1, 2, 3, 4]
```

range() generates an iterator to progress integers starting with 0 upto n-1. To obtain a list object of the sequence, it is typecasted to list(). Now this list can be iterated using the for statement.

```
>>> for var in list(range(5)):
    print (var)
```

This will produce the following output.

```
0
1
2
3
4
```

---

## Example

```
#!/usr/bin/python3
for letter in 'Python':    # traversal of a string sequence
    print ('Current Letter :', letter)
print()
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:      # traversal of List sequence
    print ('Current fruit :', fruit)

print ("Good bye!")
```

## Nested loops

Python programming language allows the use of one loop inside another loop. The following section shows a few examples to illustrate the concept.

### Syntax

#### **for iterating\_var in sequence:**

for iterating\_var in sequence:

statements(s)

statements(s)

The syntax for a nested while loop statement in Python programming language is as Follows

#### **while expression:**

while expression:

statement(s)

statement(s)

A final note on loop nesting is that you can put any type of loop inside any other type of loop. For example a for loop can be inside a while loop or vice versa.

Example:

```
for i in range(1,11):
    for j in range(1,11):
        k=i*j
        print (k, end=' ')
    print()
```

Output

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

## Do While Loop

Python doesn't have do-while loop. But we can create a program like this. The do while loop is used to check condition after executing the statement. It is like while loop but it is executed at least once.

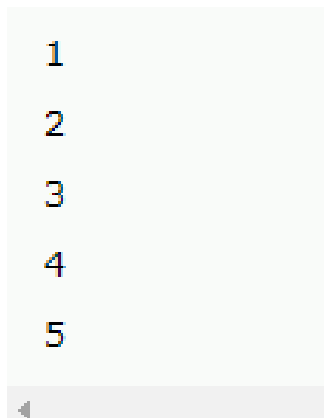
### General Do While Loop Syntax

```
do {  
    //statement  
} while (condition);
```

### Python Do While Loop Example

```
i = 1  
while True:  
    print(i)  
    i = i + 1  
    if(i > 5):  
        break
```

### Output:



```
1  
2  
3  
4  
5
```

## Lab Tasks

1. Write a Python program to convert temperatures to and from celsius, fahrenheit.  
[ Formula :  $c/5 = f-32/9$  [ where c = temperature in celsius and f = temperature in fahrenheit ]  
*Expected Output :*  
60°C is 140 in Fahrenheit  
45°F is 7 in Celsius

2. Write a Python program to construct the following pattern, using a nested for loop.

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * *  
* * * *  
* * *  
* *  
*
```

3. Write a Python program that accepts a word from the user and reverse it.
4. Make a function which prints the table of the number entered by user from starting value to

last value. Function must takes 3 inputs x, y, z from the user and pass them to the user.

- X be the number whose table we want to print
- Y be the starting value
- Z be the ending value.

Make sure ending value must be greater than the starting value