# Government College of Engineering, Jalgaon

**(An Autonomous Institute of Govt of Maharashtra)**

**Department of Computer Engineering**

## Lab Manual

## For

## CO210U Problem Solving using Python Lab

## S. Y. B. Tech.(Computer)

**Prepared by**

**Mr. Harish D. Gadade**

**Asst. Professor in Computer Technology**

**Computer Department**

**Mr. Priyanka Gadade**
**Course Teacher**

**Mr. Harish D. Gadade**
**Course Coordinator**

# Experiment No.

**Student Name :**_____ **PRN**_____

**Course Teacher: Mrs. Priyanka Gadade, Computer Engg, Govt College of Engg, Jalgaon**

---

**Objective:** Get started with Python and learn the basic types and control flow statements.

**Theory:**

## 1. Introduction

Python is a high level, interpreted, general purpose, dynamic programming language. Python was conceived in the late 1980s and its usage began from December 1989. The syntax of Python programs can express concepts in fewer lines as compared to programs in C, C++ and Java.

Python can be used in multiple programming styles, including Object Oriented, Functional Programming, Procedural programming and Iterative styles. Python can be used in almost all operating system because its interpreter is available for many operating systems. Python is free and open source software.

## 1.1 Installation

- ❏ Download python installer from link
  https://www.Python.org/downloads/
- ❏ Download the .tgz file of Python from the above provided download link
  http://www.python.org/ftp/python.2.7.11/python-2.7.11.tgz
- ❏ Unzip the downloaded file.
  Tar xvfz Python-2.7.11.tgz
- ❏ Go to the directory
  cd Python-2.7.11
- ❏ ./configure
- ❏ Build it.
- ❏ Make
- ❏ Su or sudo su if there is no root user
- ❏ Make install

## 1.2 What is python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side)
- software development

- mathematics
- system scripting.

Python Syntax compared to other programming languages

- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

### 1.3 How to write program in any editor:

To write any python program, first open any editor like vm,pico, text or nano and then type python code in it which is given in below examples. After typing complete program code, you can directly run your python program.

### 1.4 How to Run?

In python, we can directly run our program using following syntax

```
harish@harish-Inspiron-3537:~$ python file_name.py
```

Example

```
harish@harish-Inspiron-3537:~$ python sample.py
```

### 1.5 Python Indentations:

In other programming languages, indentation is used for formating for for readability only but in python indentation is very important. It is used to indicate block of codes

Example:

```
If 10<20:
    print("10 is smaller that 20")
```

If you type code like

```
If 10<20:
print("10 is smaller that 20")
```

It will give an error.

But in other languages it will not give an error.

### 1.6 Print command:

Print command is used to display text or data on the screen

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

Forms of print command

```
print("Hello")
print('Hello')
print "Hello"   #This is recommended in python 2.0
```

## 2. Variables

### 2.1 Declaring Variables

A variable holds a value that may change. The process of writing the variable name is called Declaring the variable. In Python, variables need not be declared explicitly in order to reserve memory space as in other programming languages like C, Java etc. When we initialize the variable in Python, Python Interpreter automatically does the declaration process.

### 2.1 Initializing a Variable

- The general format of assignment statement is
  variable=expression
  E.g.

1.
```
>>>year=2017
>>> name='Harish'
```

2.
```
>>>name1='Harish'
>>>name2=name1
>>>name2
'Harish'                # Output
>>
```

3.
```
>>>year=2016
>>>year=2017
>>>year
2017                    # Output
```

4.
```
>>>amount=50
>>>amount
50                      # Output
>>>amount='Fifty'
>>>amount
'Fifty'                 # Output
```

## 3. Standard Data Types

- ❑ Python has six basic data types

1. Numeric
2. String
3. List
4. Tuple
5. Dictionary
6. Boolean

## 3.1 Numeric

- ❑ Numeric data can be divided into integers and real numbers. Integer can themselves be positive or negative.
- ❑ Unlike many other programming languages, Python does not have any upper bound on the size of integers.

E.g.

```
1.  >>> n1=2                  #Integer Number
    >>> n2=2.5                # Real Number
    >>>n1
    2                         # Output
    >>>n2
    2.5                       # Output
    >>>
2.  >>>5.2
    2                         # Output it is in version of Python 3 only
    >>>
3.  >>>5.0/2
    2.5                       # Output
    >>>
4.  >>>5/2
    2.5                       # Output: It is in the above version of Python 3
    >>>
```

## 3.2 String

- ❑ There are several operators such as slice operator([ ]) and ([:]), concat operator (+), repetition operator(*) etc. Slicing is used to take take out a subset of a string, repetition is used to repeat the same string several times.

E.g.

```
1.  >>>str1=”GCOE”            # Store String Values
    >>>str1
    ‘GCOE’                    # Output
    >>>str1 + “Jalgaon”       # Use of + Operator
```

```
'GCOEJalgaon'
>>>str1 * 3                    # Use of * Operator
'GCOEGCOEGCOE'
```

2.
```
>>>str1="Harish"
>>>str1[1]                     # Output
'a'
>>>str1[0:2]
'Ha'
```

3.
```
>>>str1="GCOEJalgaon"
>>>str1[1:10:2]                # Display all the alternate characters b/w
'CEago'                         index 1 to 8 i.e 1,3,5,7,9
```

4.
```
>>>length="GCOEJalgaon"
>>>print len(length)
11
```

## 3. Standard Data Types

- Python has six basic data types

7. Numeric
8. String
9. List
10. Tuple
11. Dictionary
12. Boolean

### 3.1 Numeric

- Numeric data can be divided into integers and real numbers. Integer can themselves be positive or negative.
- Unlike many other programming languages, Python does not have any upper bound on the size of integers.

E.g.

1.
```
>>> n1=2                      #Integer Number
 >>> n2=2.5                   # Real Number
>>>n1
2                             # Output
>>>n2
```

```
        2.5                 # Output
        >>>
2.      >>>5.2
        2                   # Output it is in version of Python 3 only
        >>>
3.      >>>5.0/2
        2.5                 # Output
        >>>
4.      >>>5/2
        2.5                 # Output: It is in the above version of Python 3
        >>>
```

### 3.2 String

- There are several operators such as slice operator([ ]) and ([:]), concat operator (+), repetition operator(*) etc. Slicing is used to take out a subset of a string, repetition is used to repeat the same string several times.

E.g.

```
1.      >>>str1="GCOE"          # Store String Values
        >>>str1
        'GCOE'                  # Output
        >>>str1 + "Jalgaon"     # Use of + Operator
        'GCOEJalgaon'
        >>>str1 * 3             # Use of * Operator
        'GCOEGCOEGCOE'

2.      >>>str1="Harish"
        >>>str1[1]              # Output
        'a'
        >>>str1[0:2]
        'Ha'

3.      >>>str1="GCOEJalgaon"
        >>>str1[1:10:2]         # Display all the alternate characters b/w
        'CEago'                   index 1 to 8 i.e 1,3,5,7,9

4.      >>>length="GCOEJalgaon"
        >>>print len(length)
        11
```

### 3.6 Boolean

- True and False data is known as boolean data. The data types which stores this boolean data are known as boolean data types.

E.g.

1.
```
>>> a = True
>>> type (a)
<type 'bool'>

>>> x = False
>>> type (x)
<type 'bool'>
```

### 3.7 Sets

- The list and dictionaries in python are known as sequence or order collection of data. However in python, we also have one data type which is an unordered collection of data known as set.
- Union, Intersection, Difference and symmetric difference are some operations which are performed on sets.

E.g.

# Defining Sets

1.
```
>>>set1 = set([1, 2, 4, 1, 2, 8, 5, 4])
>>>set2 = set([1, 9, 3, 2, 5])

>>>print set1
set([8, 1, 2, 4, 5])
>>>print set2
set([1, 2, 3, 5, 9])

>>> intersection = set1 & set2
>>> print intersection
set([1, 2, 5])

>>>union = set1 | set2
>>>print union
set([1, 2, 3, 4, 5, 8, 9])

>>>difference = set1 - set2
>>>print difference
set([8, 4])
```

```
>>>symm_diff = set1 ^ set2
>>>print symm_diff
set([3, 4, 8, 9])
```

### 3.8 type() function

- type() function in python programming language is a built in function which returns the data types of any arbitrary object.
- The object is passed as an argument to the type() function. The type() function can take anything as an argument and returns its data types such as integer, strings, dictionaries, lists, classes, modules, tuples, functions etc

E.g.

1.
```
>>> x = 10
>>> type(x)
<type 'int'>
```

2.
```
>>> type('Vihaan'')
<type 'str'>
```

3.
```
>>> import os
>>> type(os)
<type 'module'>
```

4.
```
>>> tup = (1, 2, 3)
>>> type(tup)
<type 'tuple'>
```

5.
```
>>> li = [1, 2, 3]
>>> type(li)
<type 'list'>
```

## 4. Operators

- Operators are construct used to modify the values of operands.

### 4.1 Arithmetic Operators

```
>>> x = 10
>>> y = 20
>>> z = 0

>>>z = x + y
```

```
>>> print z
30
```

**4.2 Comparison**

```
>>> x = 10
>>> y = 20
>>> z = 0
>>>     if (x == y):
                print " x is equal to y "
        else:
                Print " x is not equal to y "
x is not equal to y                                  # Output
```

**4.3 Assignment Operator**

```
>>> x = 10
>>> y = 20
>>> y + = x
>>> print y
30                          # Output
```

**4.4. Bitwise Operator**

```
>>> x = 10                          # 10 = 0000 1010
>>> y = 12                          # 12 = 0000 1100
>>> z = 0

# Bitwise AND
>>> z = x & y
>>> print z
8                                   # 8 = 0000 1000
```

**4.5 Logical Operator**

```
>>> x = True
>>> y = False
>>>print (x and y)
False
```

## 5. Control Statements

Followings are the Control Statements

**5.1 The for Loop**

Syntax

```
>>>   for x in y :
            Block 1
      else:                          # optional
            Block 2
```

e.g.

1.
```
>>>   for letter in 'HARISH' :
            Print ' Current Letter : ', letter
# Output:
Current Letter : H
Current Letter : A
Current Letter : R
Current Letter : I
Current Letter : S
Current Letter : H
```

2.
```
>>> subjects = ["APL",  "OS", "DBMS", "DAA", "FMIS", "AIES"]
>>> for x in subjects:
        print(x)
# Output
APL
OS
DBMS
DAA
FMIS
AIES
```

3.
```
>>>   for x in range(4):
            print(x)
      else:
            print ('Else Part')
0
1
2
3
Else Part
```

**5.2 Range ( ) Function**

```
>>> range(8)
[0, 1, 2, 3, 4, 5, 6, 7]
>>> range(3, 9)
[3, 4, 5, 6, 7, 8]

>>> range(3, 40, 5)
[3, 8, 13, 18, 23, 28, 33, 38]

>>> subjects =  [“DSGT”,  “PPS”, “DELD”, “PSP”, “OOPS”, “OS”]
>>> for index in range(len(subjects)):
        print ‘The Subject is : ‘, subjects[index]

# Output
The Subject is : DSGT
The Subject is : PPS
The Subject is : DELD
The Subject is : PSP
The Subject is : OOPS
The Subject is : OS
```

### 5.3 while statement

Syntax:

```
>>>   while condition :
          Block
      else:                          #optional
          Statement
```

E.g:

```
      >>> count = 0
      >>>while count < 6:
          print count
          Count += 1
# Output
      0
      1
      2
      3
      4
      5
```

## 5.4 break and continue Statements

E.g:

1.
```
# Print first five numbers.
>>> count = 2
>>> while True :
        print count
        count = count + 2
        If count >= 12 :
            break
# Output
2
4
6
8
10
```

**2.**
```
#Print first five numbers.
>>> for i in range(1,10):
        If i % 2!= 0 :
            continue
        print i
# Output
2
4
6
8
```

## 5.5 if elif else Statement

Syntax:

1.
```
>>> if expression :
        statement1
    else :
        statement2
```

2.
```
>>> if expression1 :
        statement1
    elif expression2 :
        statement2
    elif expression3 :
        statement3
    else expression4 :
```

statement4

E.g:

```
>>>var = 100
        if (var == 100) :
                print "Value of expression is 100"
#Output
Value of expression is 100
```

## 6. Input from keyboard

There are two ways to provide input from keyboard:

### 6.1 input() function

- ❏ input() function has an optional parameter, which is the prompt string.
- ❏ When the input() function is called in order to take input from the user then the execution of program halts and wait for the user to provide an input.

E.g

```
What is your name? 'John'                    # Output
Hello John!

>>>age = input("Enter your age?")
>>>print age

Enter your age? 32                           # Output
32

>>>hobby = input("What are your hobbies?")
>>>print hobby

What are your hobbies? ['playing' , 'sketching']      #Output
['playing' , 'sketching']

>>> type(name)
<type 'str'>

>>>type(age)
<type 'int'>

>>>type(hobby)
<type 'list'>
```

### 6.2 raw_input() function

- raw_input() also take the input from the user but it does not interpret the input and also it returns the input of the user without doing any changes.

E.g:

```
# No casting
>>> age = raw_input("What is your age?")
What is your age? 46                    #Output
>>>type(age)
<type 'str'>                            #input is stored as string

#Using casting function to convert input to integer
>>>age = int(raw_input("What is your age?"))
What is your age? 46
>>>type(age)
<type 'int'>                            #input is stored as integer
```

## Lab practice:

1. Write a Python program to find square root of a number
2. Write a Python program to find the area of a rectangle
3. Write a Python program to swap the values of two variables
4. Write a function to find HCF of some given numbers.
5. Write a function to display factorial of given numbers
6. Write a Python program to check if the input year is leap or not.
7. Write a function to convert a decimal number into binary, octal, hexadecimal number.
8. Write a program for python data structures: list, dictionaries and tuples.
9. Write a Python program to find whether a number is even or odd.
10. Write a Python program to check the largest among the given three numbers.
11. Write a Python program to display the Fibonacci sequence for n terms.
12. Write a Python program to demonstrate while loop with else.
13. Write a Python program to perform operations on word "governmentcollege", extract second letter, extract first four letters and extract last six letters.
14. Write a function to find sum of several natural numbers using recursion
15. Write a program to check whether a string is a palindrome or not
16. Write a python program to demonstrate various strings functions and operations

**Course Teacher Sign with Date**