VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELGAVI-590018, KARNATAKA

PYTHON PROGRAMMING LAB
15CS664 (2015 scheme)

LAB MANUAL

Prepared By:

Mr. Vishnuvardhan. Y
Assistant Professor
Dept. of CSE, MSEC

Department of Computer Science and Engineering

M.S Engineering College

(An ISO 9001:2008 Certified Institution)
(Affiliated to Visvesvaraya Technological University Belgaum and approved by AICTE, New Delhi)
NAVARATHNA AGRAHARA, SADAHALLI POST, BANGALORE- 562 110, Tel: 080-3252 9939, 3252 957

CONTENTS

PYTHON PROGRAMMING LAB

15CS664 (2015 scheme)

[As per Choice Based Credit System (CBCS) scheme] (Effective from the academic year 2016 -2017)

SEMESTER – VI

Subject Code	15CS664 (2015 scheme)	**Practice Lab**

Syllabus

1) Write a Python program to find GCD of two numbers.
2) Write a Python Program to find the square root of a number by Newton's Method
3) Write a Python program to find the exponentiation of a number.
4) Write a Python Program to find the maximum from a list of numbers.
5) write a Python Program to perform Linear Search
6) write a Python Program to perform Binary Search
7) Write a Python Program to perform selection sort.
8) Write a Python Program to perform insertion sort.
9) Write a Python Program to perform Merge sort.
10) Write a Python program to find first n prime numbers.
11) Write a Python program to multiply matrices.

Course Objective:

- Learn Syntax and Semantics and create Functions in Python.
- Handle Strings and Files in Python.
- Understand Lists, Dictionaries and Regular expressions in Python.
- Implement Object Oriented Programming concepts in Python
- Build Web Services and introduction to Network and Database Programming in Python.

Course Outcomes:

- Examine Python syntax and semantics and be fluent in the use of Python flow control and functions.
- Demonstrate proficiency in handling Strings and File Systems.

- Create, run and manipulate Python Programs using core data structures like Lists, Dictionaries and use Regular Expressions.
- Interpret the concepts of Object-Oriented Programming as used in Python.
- Implement exemplary applications related to Network Programming, Web Services and Databases in Python.

Program 1: Write a Python program to find GCD of two numbers.

Aim:

To write a Python program to find GCD of two numbers.

Algorithm:

1. Define a function named compute GCD()

2. Find the smallest among the two inputs x and y

3. Perform the following step till smaller+1

Check if ((x % i == 0) and (y % i == 0)), then assign GCD=i

4. Print the value of gcd

Program:

```
def computeGCD(x, y):
   if x > y:
      smaller = y
   else:
      smaller = x
   for i in range(1, smaller+1):
      if((x % i == 0) and (y % i == 0)):
         gcd = i
         return gcd
num1 = 54
num2 = 24
# take input from the user
# num1 = int(input("Enter first number: "))
# num2 = int(input("Enter second number: "))
print("The GCD. of", num1,"and", num2,"is", computeGCD(num1, num2))
```

Sample Output:

$python main.py

('The GCD. of', 54, 'and', 24, 'is', 6)

Program 2: Write a Python Program to find the square root of a number by Newton's Method

Aim:

To write a Python Program to find the square root of a number by Newton's Method.

Algorithm:

1. Define a function named newtonSqrt().
2. Initialize approx as 0.5*n and better as 0.5*(approx.+n/approx.)
3. Use a while loop with a condition better!=approx to perform the following,
    i. Set approx.=better
    ii. Better=0.5*(approx.+n/approx.)
4. Print the value of approx..

Program:

```
def newtonSqrt(n):
   approx = 0.5 * n
   better = 0.5 * (approx + n/approx)
   while better != approx:
      approx = better
      better = 0.5 * (approx + n/approx)
   return approx
print('The square root is' ,newtonSqrt(100))
```

Sample Output:

The square root is 10

Program 3: Write a Python program to find the exponentiation of a number.

Aim:

To write a Python program to find the exponentiation of a number.

Algorithm:

1. Define a function named power()

2. Read the values of base and exp

3. Use 'if' to check if exp is equal to 1 or not

i. if exp is equal to 1, then return base

ii.if exp is not equal to 1, then return (base*power(base,exp-1))

4. Print the result.

Program:

```
def power(base,exp):
   if(exp==1):
      return(base)
   if(exp!=1):
      return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value: "))
print("Result:",power(base,exp))
```

Sample Output:

```
Enter base: 7
Enter exponential value: 2
Result:49
```

Program 4: Write a Python Program to find the maximum from a list of numbers.

Aim:

To write a Python Program to find the maximum from a list of numbers.

Algorithm:

1. Create an empty list named l

2. Read the value of n

3. Read the elements of the list until n

4. Assign l[0] as maxno

5. If l[i]>maxno then set maxno=l[i]

6. Increment i by 1

7. Repeat steps 5-6 until i<n

8. Print the value of maximum number

Program:

```
l=[]
n=int(input("enter the upper limit"))
for i in range(0,n):
   a=int(input("enter the numbers"))
   l.append(a)
maxno=l[0]
for i in range(0,len(l)):
   if l[i]>maxno:
      maxno=l[i]
print("The maximum number is %d"%maxno)
```

Sample Output:

```
Enter the upper limit 3
Enter the numbers 6
Enter the numbers 9
Enter the numbers 90
The maximum number is 90
```

Program 5: Write a Python Program to perform Linear Search

Aim:

To write a Python Program to perform Linear Search

Algorithm:

1. Read n elements into the list
2. Read the element to be searched
3. If alist[pos]==item, then print the position of the item
4. else increment the position and repeat step 3 until pos reaches the length of the list

Program:

```
items = [5, 7, 10, 12, 15]
print("list of items is", items)
x = int(input("enter item to search:")
i = flag = 0
while i < len(items):
    if items[i] == x:
        flag = 1
        break
    i = i + 1
if flag == 1:
    print("item found at position:", i + 1)
else:
    print("item not found")
```

Sample Output:

$python main.py

(list of items is: [5, 7, 10, 12, 15])

enter item to search: 7

(item found at position:, 2)

Program 6: Write a Python Program to perform Binary Search

Aim:

To write a Python Program to perform binary search.

Algorithm:

1. Read the search element
2. Find the middle element in the sorted list
3. Compare the search element with the middle element
    i. if both are matching, print element found
    ii. else then check if the search element is smaller or larger than the middle element
4. If the search element is smaller than the middle element, then repeat steps 2 and 3 for the left sublist of the middle element
5. If the search element is larger than the middle element, then repeat steps 2 and 3 for the right sublist of the middle element
6. Repeat the process until the search element if found in the list
7. If element is not found, loop terminates

Program:

```
# Python code to implement iterative Binary Search.
# It returns location of x in given array arr
# if present, else returns -1

def binarySearch(arr, l, r, x):
        while l <= r:
                mid = l + (r - l)/2;
                # Check if x is present at mid
                if arr[mid] == x:
                        return mid
                # If x is greater, ignore left half
                elif arr[mid] < x:
                        l = mid + 1
                # If x is smaller, ignore right half
                else:
                        r = mid - 1
        # If we reach here, then the element
        # was not present
        return -1
# Test array
arr = [ 2, 3, 4, 10, 40 ]
x = 4
# Function call
result = binarySearch(arr, 0, len(arr)-1, x)
if result != -1:
```

```
        print "Element is present at index % d" % result
else:
        print "Element is not present in array"
```

Sample Output:

```
$python main.py
Element is present at index  2
```

Program 7: Write a Python Program to perform selection sort.

Aim:

To write a Python Program to perform selection sort.

Algorithm:

1. Create a function named selection sort
2. Initialise pos=0
3. If alist[location]>alist[pos] then perform the following till i+1,
4. Set pos=location
5. Swap alist[i] and alist[pos]
6. Print the sorted list

Program:

```
def selectionSort(alist):
    for i in range(len(alist)-1,0,-1):
        pos=0
        for location in range(1,i+1):
            if alist[location]>alist[pos]:
                pos= location
        temp = alist[i]
        alist[i] = alist[pos]
        alist[pos] = temp
alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

Sample Output:

$python main.py

[17, 20, 26, 31, 44, 54, 55, 77, 93]

Program 8: Write a Python Program to perform insertion sort.

Aim:

To write a Python Program to perform insertion sort.

Algorithm:

1. Create a function named insertionsort
2. Initialise currentvalue=alist[index] and position=index
3. while position>0 and alist[position-1]>currentvalue, perform the following till len(alist)
4. alist[position]=alist[position-1]
5. position = position-1
6. alist[position]=currentvalue
7. Print the sorted list

Program:

```
def insertionSort(alist):
    for index in range(1,len(alist)):
        currentvalue = alist[index]
        position = index
        while position>0 and alist[position-1]>currentvalue:
            alist[position]=alist[position-1]
            position = position-1
    alist[position]=currentvalue
alist = [54,26,93,17,77,31,44,55,20]
insertionSort(alist)
print(alist)
```

Sample Output:

```
 $python main.py
 [20, 54, 54, 54, 54, 54, 93, 93, 93]
```

Program 9: Write a Python Program to perform Merge sort.

Aim:

To write a Python Program to perform Merge sort.

Algorithm:

1. Create a function named mergesort
2. Find the mid of the list
3. Assign lefthalf = alist[:mid] and righthalf = alist[mid:]
4. Initialise i=j=k=0
5. while i < len(lefthalf) and j < len(righthalf), perform the following
    if lefthalf[i] < righthalf[j]:
        alist[k]=lefthalf[i]
        Increment i
    else
        alist[k]=righthalf[j]
        Increment j
        Increment k
6. while i < len(lefthalf),perform the following
    alist[k]=lefthalf[i]
    Increment i
    Increment k
7. while j < len(righthalf), perform the following
    alist[k]=righthalf[j]
    Increment j
    Increment k
8. Print the sorted list

Program:

```
# Python program for implementation of MergeSort
# Merges two subarrays of arr[].
# First subarray is arr[l..m]
# Second subarray is arr[m+1..r]
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r- m

    # create temp arrays
    L = [0] * (n1)
    R = [0] * (n2)

    # Copy data to temp arrays L[] and R[]
    for i in range(0 , n1):
        L[i] = arr[l + i]
```

```
    for j in range(0 , n2):
        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]
    i = 0     # Initial index of first subarray
    j = 0     # Initial index of second subarray
    k = l     # Initial index of merged subarray

    while i < n1 and j < n2 :
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    # Copy the remaining elements of L[], if there
    # are any
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    # Copy the remaining elements of R[], if there
    # are any
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

# l is for left index and r is right index of the
# sub-array of arr to be sorted
def mergeSort(arr,l,r):
    if l < r:
        # Same as (l+r)/2, but avoids overflow for
        # large l and h
        m = (l+(r-1))/2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)

# Driver code to test above
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print ("Given array is")
```

```
for i in range(n):
        print ("%d" %arr[i]),

mergeSort(arr,0,n-1)
print ("\n\nSorted array is")
for i in range(n):
        print ("%d" %arr[i]),
```

Sample Output:

```
$python main.py
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```

Program 10: Write a Python program to find first n prime numbers.

Aim:

To write a Python program to find first n prime numbers.

Algorithm:

1. Read the value of n
2. for num in range(0,n + 1), perform the following
3. if num%i is 0 then break

   else print the value of num
4. Repeat step 3 for i in range(2,num)

Program:

```
n = int(input("Enter the upper limit: "))
print("Prime numbers are")
for num in range(0,n + 1):
   # prime numbers are greater than 1
   if num > 1:
      for i in range(2,num):
         if (num % i) == 0:
            break
      else:
         print(num)
```

Sample Output:

```
$python main.py
Enter the upper limit: 20
 Prime numbers are
2
3
5
7
11
13
17
19
```

Program 11: Write a Python program to multiply matrices.

Aim:

To write a Python program to multiply matrices.

Algorithm:

1. Define two matrices X and Y
2. Create a resultant matrix named 'result'
3. for i in range(len(X)):
    i. for j in range(len(Y[0])):
        a) for k in range(len(Y))
        b) result[i][j] += X[i][k] * Y[k][j]
4. for r in result, print the value of r

Program:

```
X = [[12,7,3],
   [4 ,5,6],
   [7 ,8,9]]
Y = [[5,8,1,2],
   [6,7,3,0],
   [4,5,9,1]]
result = [[0,0,0,0],
       [0,0,0,0],
       [0,0,0,0]]
for i in range(len(X)):
   for j in range(len(Y[0])):
      for k in range(len(Y)):
         result[i][j] += X[i][k] * Y[k][j]
for r in result:
   print(r)
```

Sample Output:

[114, 160, 60, 27]

[74, 97, 73, 14]

[119, 157, 112, 23]

VIVAVOCE QUESTIONS

1. What it the syntax of print function?
2. What is the usage of input function?
3. Define a variable.
4. What is type conversion?
5. Mention the data types in Python
6. What are the attributes of the complex datatype?
7. Mention a few escape sequences.
8. Define an expression
9. What is the usage of ** operator in Python?
10. Give the syntax of if else statement.
11. Give the syntax of for statement.
12. How is range function used in for?
13. Give the syntax of while statement.
14. What are multi way if statements?
15. How is random numbers generated?
16. Define a function.
17. Give the syntax of function.
18. What are the types of arguments in function.?
19. What is a recursive function?
20. What are anonymous functions?
21. What are default arguments?
22. What are variable length arguments?
23. What are keyword arguments?
24. Mention the use of map().
25. Mention the use of filter().
26. Mention the use of reduce().
27. Define a string.
28. How is string slicing done?
29. What is the usage of repetition operator?
30. How is string concatenation done using + operator>
31. Mention some string methods
32. How is length of a string found?

33. How is a string converted to its upper case?
34. `Differentiate isalpha() and isdigit().
35. What is the use of split()?
36. Define a file.
37. Give the syntax for opening a file.
38. Give the syntax for closing a file.
39. How is reading of file done?
40. How is writing of file done?
41. What is a list?
42. Lists are mutable-Justify.
43. How is a list created?
44. How can a list be sorted?
45. How are elements appended to the list?
46. How is insert() used in list?
47. What is the usage of pop() in list?
48. Define a tuple.
49. Are tuples mutable or immutable?
50. Mention the use of return statement.
51. What is a Boolean function?
52. How is main function defined?
53. What is a dictionary?
54. How are tuples created?
55. How is a dictionary created?
56. How to print the keys of a dictionary?
57. How to print the values of a dictionary?
58. How is del statement used?
59. Can tuple elements be deleted?
60. What is Python interpreter?
61. Why is Python called an interpreted language?
62. Mention some features of Python
63. What is Python IDLE?
64. Mention some rules for naming an identifier in Python.
65. Give points about Python Numbers.

66. What is bool datatype?
67. Give examples of mathematical functions.
68. What is string formatting operator?
69. Mention about membership operators in Python.
70. How is expression evaluated in Python?
71. What are the loop control statements in Python?
72. What is the use of break statement?
73. What is the use of continue statement?
74. What is the use of pass statement?
75. What is assert statement?
76. Differentiate fruitful function s and void functions.
77. What are required arguments ?
78. Differentiate pass by value and pass by reference.
79. Mention few advantages of function.
80. How is lambda function used?
81. What is a local variable?
82. What are global variables?
83. What are Python decorators?
84. Are strings mutable or immutable?
85. What is join()?
86. What is replace() method?
87. What is list comprehension?
88. Define multidimensional list.
89. How to create lists using range()?
90. What is swapcase() method?
91. What is linear search?
92. How is binary search done?
93. How is merge sort performed?
94. What is sorting?
95. How is insertion sort done?
96. How is selection sort done?
97. What are command line arguments?
98. Name some built in functions with dictionary.

99. What is an exception?

100. How is exception handled in python?

********** **The End** **********