Python Lab Manual

GENERAL LAB INSTRUCTIONS

DO’S

- Enter the lab on time and leave at proper time.
- Wait for the previous class to leave before the next class enters.
- Keep the bag outside in the respective racks.
- Utilize lab hours in the corresponding.
- Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
- Leave the labs at least as nice as you found them.
- If you notice a problem with a piece of equipment (e.g. a computer doesn't respond) or the room in general (e.g. cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

DON’TS

- Don't abuse the equipment.
- Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
- Do not attempt to reboot a computer. Report problems to lab staff.
- Do not remove or modify any software or file without permission.
- Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
- Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, log out before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
- Don’t use internet, internet chat of any kind in your regular lab schedule.
- Do not download or upload of MP3, JPG or MPEG files.
- No games are allowed in the lab sessions.
- No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
- No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling anything on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact has) If you need to eat or drink, take a break and do so in the canteen.
- Don’t bring any external material in the lab, except your lab record, copy and books.
- Don’t bring the mobile phones in the lab. If necessary then keep them in silence mode.
- Please be considerate of those around you, especially in terms of noise level. While labs are a natural place for conversations of all types, kindly keep the volume turned down.

If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

Experiment–1

AIM: Write a program to demonstrate basic data type in python.

Program:

```
print(type(1))
print(type(2.2))
print(type('ABCD'))
print(type([1,2,3]))
print(type({1:'a'}))
print(type((1,2,3)))
print(type({1,2,3}))
```

Output:

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'list'>
<class 'dict'>
<class 'tuple'>
<class 'set'>
```

Experiment–2

AIM:

- Write a program to compute distance between two points taking input from the user.

Program:

```
#formula : sqrt((x2 - x1)^2 + (y2 - y1)^2).
x1= int(input("enter X1 value"))
x2= int(input("enter X2 value"))
y1= int(input("enter y1 value"))
y2= int(input("enter y2 value"))
result=(((x2-x1)**2 + (y2-y1)**2)**0.5)
print("the distance between two points",(x1,x2),"and",(y1,y2),"=",result)
```

Output:

```
enter X1 value5
enter X2 value6
enter y1 value4
enter y2 value3
the distance between two points (5, 6) and (4, 3) = 1.4142135623730951
```

- Write a program add.py that takes 2 numbers as input from user and prints its sum.

Program:

```
num1=int(input("Enter the first Number"))
num2=int(input("Enter the Second Number"))
sum=num1+num2
print("The Sum of Two numbers is", sum)
```

Output:
Enter the first Number 5
Enter the Second Number 4
The Sum of Two numbers is 9

Experiment–3

AIM:

- Write a Python program to check given number is even or not.

Program:

```
number=int(input("enter the number to be check"))
if (number%2==0):
print("number is even")
else:
print("number is not even")
```

Output:
enter the number to be check5
number is not even

- Using a for loop, write a program that prints out the decimal equivalents of ½, 1/3, 1/4… 1/10

Program:

```
for i in range(2,11):
print(1/i)
```

Output:
0.5
0.3333333333333333
0.25
0.2
0.16666666666666666
0.14285714285714285
0.125
0.1111111111111111
0.1

Experiment–4

AIM:

- Write a program using a for loop that loops over a sequence.

Program:

```
#For loop in list
color_list=['red','green','white','purple','pink','marron']
for i in color_list:
print(i)
#For loop in string
fr_string="Goeduhub technologies"
for i in fr_string:
print(i)
#For loop in dictionary
fr_dict={1:'q',2:'w',3:'e',4:"r"}
for i in fr_dict:
print(i)
print(fr_dict[i])
#Sum of all the list elements
Num=[1,2,3,4,5,6,7,8,9,10]
Sum=0
for i in Num:
Sum=Sum+i
print(Sum)
```

Output:

```
red
green
white
purple
pink
marron
G
o
e
d
u
h
u
b
t
e
c
```

h
n
o
l
o
g
i
e
s
1
q
2
w
3
e
4
R
55

- Write a program using while loop that asks the user for a number, and prints a countdown from that number to zero.

Program:
num=int(input("Enter your number"))
while(num>=0):
print(num)
num=num-1
Output:
Enter your number5
5
4
3
2
1
0

- Write a program to demonstrate list in python

Program:
#Create a List:
thislist = ["apple", "banana", "cherry"]
print(thislist)
Print the number of items in the list:
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
#Print the second item of the list:
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
#Return the third, fourth, and fifth item:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon"]
print(thislist[2:5])
```

- Write a program to demonstrate tuples in python

Program:

```
#Example
#Tuples allow duplicate values:
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
#Example
#Print the number of items in the tuple:
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
#Example
#One item tuple, remember the commma:
thistuple = ("apple",)
print(type(thistuple))
#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
#Example
#A tuple with strings, integers and boolean values:
tuple1 = ("abc", 34, True, 40, "male")
print(tuple1)
#Example
#Print the second item in the tuple:
thistuple = ("apple", "banana", "cherry")
print(thistuple[0])
#Example
#Print the last item of the tuple:
thistuple = ("apple", "banana", "cherry")
print(thistuple[-2])
#Example
#Convert the tuple into a list to be able to change it:
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

Experiment–5

AIM:

- Find the sum of all the primes below two million.

Program:

```
def IsPrime(x):
returnVal = True
```

```
for i in range(2,int(x)):
if (int(x) % i)==0:
returnVal=False
break
return returnVal
####################################3

sum=2

for i in range(3,2000000):
if IsPrime(i):
print("adding " + str(i))
sum = sum + i
print(sum)
```

- By considering the terms in the Fibonacci sequence whose values do not exceed four million, WAP to find the sum of the even-valued terms.

Program:

```
sum = 2
a = 1
b = 2
c = 0

while c <= 4000000:
    c = a + b
    if c%2 == 0:
        sum += c

    a,b = b,c

print(sum)
```

Experiment-6

AIM:

- Write a program to count the numbers of characters in the string and store them in a dictionary data structure

Program:

```
str=input("enter string : ")
f = {}
for i in str:
 #print (i)
 if i in f:
 f[i] += 1
```

```
else:
 f[i] = 1
 print(f)
```

Output:

```
enter string : shikha
{'s': 1, 'h': 2, 'i': 1, 'k': 1, 'a': 1}
```

- Write a program to use split and join methods in the string and trace a birthday of a person with a dictionary data structure

Program:

```
bdaystr=input("Enter date of birth:\n")
bdaylist=bdaystr.split("/")
print(bdaylist)
bday='-'.join(bdaylist)
bdaydict={"birthday":bday}
print(bdaydict['birthday'])
req= input("enter bday to match")
if bdaydict['birthday']==req:
print("yes,Birthday date matched")
else:
print("no,Birthday date not matched")
```

Output:

```
Enter date of birth:
22/4/2020
['22', '4', '2020']
22-4-2020
enter bday to match23-4-2020
no,Birthday date not matched
```

Experiment-7

AIM: Write a program to count frequency of characters in a given file. Can you use character frequency to tell whether the given file is a Python program file, C program file or a text file?

Program:

```
f="D:\\myfile\demofile.txt"
file = open ( f, "r" )
a=[]
b={}
for i in file:
for j in range(0,len(i)):
a.append(i[j])
print(a)
for i in a:
if i in b:
```

```
b[i]+=1
else:
b[i]=1
print(b)
c=f.split(".")
if c[1]=="txt":
  print("\n\nit is a text file")
elif c[1]=="cpp":
  print("\n\nit is a c++ file")
else:
  print("\n\nit is a c file")
```

Output:

```
 ['W', 'o', 'o', 'p', 's', '!', ' ', 'l', ' ', 'h', 'a', 'v', 'e', ' ', 'd', 'e', 'l', 'e', 't', 'e', 'd', ' ', 't', 'h', 'e', ' ', 'c', 'o', 'n',
't', 'e', 'n', 't', '!']
{'W': 1, 'o': 3, 'p': 1, 's': 1, '!': 2, ' ': 5, 'l': 1, 'h': 2, 'a': 1, 'v': 1, 'e': 6, 'd': 2, 'l': 1, 't': 4, 'c': 1, 'n': 2}

it is a text file
```

Experiment-8

AIM:

- Write a program to print each line of a file in reverse order.

Program:

```
ofile=open("D:\\myfile\demofile.txt","r")
k=ofile.readlines()
t=reversed(k)
for i in t:
print(i.rstrip())
ofile.close()
```

Output:

```
we are performing experiments
we are performing experiments
we are performing experiments
file handling concepts we are performing experiments we are performing experiments
this is my first file
world
hello
```

- Write a program to compute the number of characters, words and lines in file.

Program:

```
file = open("D:\\myfile\demofile.txt","r")
number_of_lines = 0
number_of_words = 0
```

```
number_of_characters = 0
for line in file:
line = line.strip()
words = line.split()
number_of_lines += 1
number_of_words += len(words)
number_of_characters += len(line)
file.close()
print("lines:", number_of_lines, "words:", number_of_words, "characters:",
number_of_characters)
```

Output:

lines: 7 words: 30 characters: 200
Experiment-9

AIM:

- Write a function nearly equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on.

Program:

```
string1 = input("Enter first string: ")
string2 = input("Enter second string: ")
if string1 == string2:
print("\nBoth strings are equal to each other.")
print(string1,"==",string2);
else:
print("\nStrings are not equal.")
print(string1,"!=",string2);
```

Output:

Enter first string: barkha
Enter second string: barkha
Both strings are equal to each other.
barkha == barkha

- Write function to compute gcd, lcm of two numbers. Each function shouldn’t exceed one line.

Program:

```
def gcd(x, y):
   while(y):
      x, y = y, x % y
   return x
# This function computes LCM
```

```
def lcm(x, y):
   lcm = (x*y)//gcd(x,y)
   return lcm
num1 = 60
num2 = 48
print("The L.C.M. is", lcm(num1, num2))
print("GCD of two numbers is ",gcd(num1,num2))
```

Output:
The L.C.M. is 240
GCD of two numbers is 12

Experiment-10

AIM:

- Write a program to implement Merge sort.

Program:

```
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
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-l)//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],end=" ")

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

Output:

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

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

- Write a program to implement Selection sort, Insertion sort.

Program:

```
#Insertion Sort
a = [16, 19, 11, 15, 10, 12, 14]
for i in a:
j = a.index(i)
#i is not the first element
while j>0:
#not in order
 if a[j-1] > a[j]:
   #swap
   a[j-1],a[j] = a[j],a[j-1]
 else:
   #in order
   break
   j = j-1
print (a)

#Selection Sort

a = [16, 19, 11, 15, 10, 12, 14]
i = 0
while i<len(a):
  #smallest element in the sublist
  smallest = min(a[i:])
  #index of smallest element
  index_of_smallest = a.index(smallest)
  #swapping
  a[i],a[index_of_smallest] = a[index_of_smallest],a[i]
  i=i+1
print (a)
```

Output:

```
[10, 11, 12, 14, 15, 16, 19]

[10, 11, 12, 14, 15, 16, 19]
```

Beyond the Syllabus Experiment-1

AIM: WAP to implement Heap Sort

Program:

```
def heapify(arr, N, i):
    largest = i  # Initialize largest as root
    l = 2 * i + 1     # left = 2*i + 1
    r = 2 * i + 2     # right = 2*i + 2
```

```
    # See if left child of root exists and is
    # greater than root
    if l < N and arr[largest] < arr[l]:
        largest = l

    # See if right child of root exists and is
    # greater than root
    if r < N and arr[largest] < arr[r]:
        largest = r

    # Change root, if needed
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]  # swap

        # Heapify the root.
        heapify(arr, N, largest)

# The main function to sort an array of given size


def heapSort(arr):
    N = len(arr)

    # Build a maxheap.
    for i in range(N//2 - 1, -1, -1):
        heapify(arr, N, i)

    # One by one extract elements
    for i in range(N-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]  # swap
        heapify(arr, i, 0)


# Driver's code
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]

    # Function call
    heapSort(arr)
    N = len(arr)

    print("Sorted array is")
    for i in range(N):
        print("%d" % arr[i], end=" ")
```

Output:

Sorted array is
5 6 7 11 12 13
Beyond the Syllabus Experiment-2

AIM: WAP to implement Binary Tree

Program:

```
from binarytree import Node
root = Node(3)
root.left = Node(6)
root.right = Node(8)

# Getting binary tree
print('Binary tree :', root)

# Getting list of nodes
print('List of nodes :', list(root))

# Getting inorder of nodes
print('Inorder of nodes :', root.inorder)

# Checking tree properties
print('Size of tree :', root.size)
print('Height of tree :', root.height)

# Get all properties at once
print('Properties of tree : \n', root.properties)
```

Output:
Binary tree :
3
/ \
6 8
List of nodes : [Node(3), Node(6), Node(8)]
Inorder of nodes : [Node(6), Node(3), Node(8)]
Size of tree : 3
Height of tree : 1
Properties of tree :
{‘height’: 1, ‘size’: 3, ‘is_max_heap’: False, ‘is_min_heap’: True, ‘is_perfect’: True, ‘is_strict’: True, ‘is_complete’: True, ‘leaf_count’: 2, ‘min_node_value’: 3, ‘max_node_value’: 8, ‘min_leaf_depth’: 1, ‘max_leaf_depth’: 1, ‘is_bst’: False, ‘is_balanced’: True, ‘is_symmetric’: False}