Techno India NJR Institute of Technology

Lab Manual
Python Lab (6CS4-23)

Akhilesh Deep Arya

Department of CSE

For Techno India NJR Institute of Technology

पंकज पोरवाल

Dr. Pankaj Kumar Porwal
(Principal)

RAJASTHAN TECHNICAL UNIVERSITY, KOTA
Syllabus
III Year-VI Semester: B.Tech. Computer Science and Engineering

6CS4-23: Python Lab

Credit: 1.5
0L+0T+3P

Max. Marks: 75(IA:45, ETE:30)
End Term Exam: 2 Hours

SN	List of Experiments
1	Write a program to demonstrate basic data type in python.
2	Write a program to compute distance between two points taking input from the user Write a program add.py that takes 2 numbers as command line arguments and prints its sum.
3	Write a Program for checking whether the given number is an even number or not. Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10
4	Write a Program to demonstrate list and tuple in python. Write a program using a for loop that loops over a sequence. Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.
5	Find the sum of all the primes below two million. 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.
6	Write a program to count the numbers of characters in the string and store them in a dictionary data structure Write a program to use split and join methods in the string and trace a birthday of a person with a dictionary data structure
7	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? 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?
8	Write a program to print each line of a file in reverse order. Write a program to compute the number of characters, words and lines in a file.
9	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. Write function to compute gcd, lcm of two numbers. Each function shouldn't exceed one line.
10	Write a program to implement Merge sort. Write a program to implement Selection sort, Insertion sort.

For Techno India NJR Institute of Technology

पंकज पोरवाल
Dr. Pankaj Kumar Porwal
(Principal)

Course Outcome:

6CS423	Python Programming Lab
CO1	Student Should be able to write, test and debug Python programs
CO2	Student should be able to write Python programs using conditional statements, looping statements and represent Compound data using Lists, Tuples and Dictionaries
CO3	Students should be able to write Python programs to perform read and write operations on text files.
CO4	Student should be able to write Python program to implement selection, bubble and merge sort.

Mapping COs, POs and PSOs:

Python Programming Lab Year of study: 2019-20														
Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	1	-	1	2	-	-	-	-	-	-	-	-	-	-
CO2	2	1	-	1	2	-	-	-	-	-	-	-	-	-	-
CO3	2	1	-	1	2	-	-	-	-	-	-	-	-	-	-
CO4	2	1	-	1	2	-	-	-	-	-	-	-	-	-	-

Mapping Justification:

CO	PO	Justification
CO1 CO2 CO3 CO4	PO1	To write programs in Python student must require the knowledge of mathematics, that's why CO1 is moderately mapped with PO1.
	PO2	When a student write or design a Python program it requires an analysis of problem statement but it does not requires complex analysis so CO1 is mapped with PO2 with low level.
	PO4	While writing the python code for complex problems such as find prime numbers below 2 million with a time constraint, requires investigation of complex algorithms such as sieve algorithm so CO1 is mapped with PO4 with low level.
	PO5	Students required to use different IDEs to write and execute python programs, so CO1 is moderately mapped with PO5

SUBJECT: PYTHON LAB (6CS4-23)

Class: III Year VI Semester

Sr. No.	Experiment Name
1.	Write a program to demonstrate basic data type in python.
2.	2 Write a program to compute distance between two points taking input from the user
3.	Write a program add.py that takes 2 numbers as command line arguments and prints its sum.
4.	3 Write a Program for checking whether the given number is an even number or not.
5.	Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10 4
6.	Write a Program to demonstrate list and tuple in python.
7.	Write a program using for loop that loops over a sequence.
8.	Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.
9.	Find the sum of all the primes below two million.
10.	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.
11.	Write a program to count the numbers of characters in the string and store them in a dictionary data structure
12.	Write a program to use split and join methods in the string and trace a birthday of a person with a dictionary data structure
13.	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?
14.	Write a program to print each line of a file in reverse order.
15.	Write a program to compute the number of characters, words and lines in a file.
16.	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.
17.	Write function to compute gcd, lcm of two numbers. Each function should exceed one line.

18.	Write a program to implement Merge sort.
19.	Write a program to implement Selection sort, Insertion sort.

1. Write a program to demonstrate basic data type in python.

Program:
```
a=10
print(type(a))

b=1.7
print(type(b))

c="Akhil"
print(type(c))

d=1j
print(type(d))

e=[1,2,3]
print(type(e))

f=(1,2,3)
print(type(f))

g=range(6)
print(type(g))

h={"name":"Akhil", "Age":26}
print(type(h))

i={"John","Marry"}
print(type(i))

j=frozenset({"Akhil","Addi","Divi"})
print(type(j))

k=True
print(type(k))
```

Output:
```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'complex'>
<class 'list'>
<class 'tuple'>
<class 'range'>
<class 'dict'>
<class 'set'>
```

<class 'frozenset'>
<class 'bool'>

Note:

Difference between **set** and **frozen set**:

Frozen set is just an immutable version of a Python set object. While elements of a set can be modified at any time, elements of frozen set remain the same after creation.

Due to this, frozen sets can be used as key in Dictionary or as element of another set. But like sets, it is not ordered (the elements can be set at any index).

```python
# random dictionary
person = {"name": "John", "age": 23, "sex": "male"}

fSet = frozenset(person)
print('The frozen set is:', fSet)
```

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

Formula:

Distance=$\sqrt{(x2-x1)^2+(y2-y1)^2}$

Program:

```
import math
print("Enter cordinates of first point: ")
x1=int(input("X1: "))
y1=int(input("Y1: "))

print("Enter cordinates of second point: ")
x2=int(input("X2: "))
y2=int(input("Y2: "))

distance=math.sqrt((x2-x1)**2+(y2-y1)**2)
print(distance)
```

Output:

```
Enter cordinates of first point:
X1: 3
Y1: 2
Enter cordinates of second point:
X2: 7
Y2: 8
7.211102550927978
```

3. Write a program add.py that takes 2 numbers as command line arguments and prints its sum.

Program:

```
%With 2 inputs only
import sys
x=int(sys.argv[1])
y=int(sys.argv[2])
sum=x+y
print(sum)

%With variable num of inputs
import sys
x=list(sys.argv[1:])
sum=0
for i in x:
    sum=sum+int(i)
print(sum)
```

Output:

4. Write a Program for checking whether the given number is an even number or not.

Program:

```python
while 1:
    x=int(input("Enter Number: "))
    if x%2:
        print("ODD")
    else:
        print("EVEN")
    y=(input("Do you want to continue y/n: "))
    if y=="n":
        break;
```

Output:

```
Enter Number: 2
EVEN
Do you want to continue y/n: y
Enter Number: 4
EVEN
Do you want to continue y/n: y
Enter Number: 5
ODD
Do you want to continue y/n: n
```

5. Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3, 1/4, . . . , 1/10

Program:

```
x=1
for y in range(11):
    if y!=0 and y!=1:
        print(x/int(y))
```

Output:

0.5

0.3333333333333333

0.25

0.2

0.16666666666666666

0.14285714285714285

0.125

0.1111111111111111

0.1

6. Write a Program to demonstrate list and tuple in python.

List is a collection which is ordered and changeable. It also allows duplicate values.

Program:

```
#Creating List
mylist=["BMW","Farari","Toyota"]
print("Printing all items of a list: ",mylist)

#Enter new element  in list
x=input("Enter New Item: ")
mylist.append(x)
print("Printing updated list: ",mylist)

#Insert Element to specific loction
x=input("Enter New Item: ")
mylist.insert(1,x)
print("Printing updated list: ",mylist)

#Remove Element from the list
x=input("Enter Item You Want To Remove: ")
mylist.remove(x)
print("Printing updated list: ",mylist)

#Create copy of list
newlist=list(mylist)
print("New list is ",newlist)

#Clear list
mylist.clear()
print("List is now empty: ",mylist)
```

Output:

```
Printing all items of a list:  ['BMW', 'Farari', 'Toyota']
Enter New Item: Tata
Printing updated list:  ['BMW', 'Farari', 'Toyota', 'Tata']
Enter New Item: Mahindra
Printing updated list:  ['BMW', 'Mahindra', 'Farari', 'Toyota', 'Tata']
Enter Item You Want To Remove: Farari
Printing updated list:  ['BMW', 'Mahindra', 'Toyota', 'Tata']
New list is ['BMW', 'Mahindra', 'Toyota', 'Tata']
List is now empty:  []
```

Tuple is a collection which is ordered and unchangeable. It also allows duplicate values.

Program:

```
#Creating tuple
mytuple=("BMW","Farari","Toyota")
print("Printing all items of a tuple: ",mytuple)
```

```
#Print elements in reverse
print("Reverse of tuple: ",mytuple[-1: :-1])

#Print total nu of elements in tuple
print("Total elelmets in tuple is: ",len(mytuple))

#Add 2 tuples
newtuple=(1,2,3,4,2,5)
final=mytuple+newtuple
print("Tuple affter addition: ",final)

#Calculate frequency of any element in tuple
x=newtuple.count(2)
print("Frequency of 2 in tuple:  "x)
```
Output:

    Printing all items of a tuple: ('BMW', 'Farari', 'Toyota')
    Reverse of tuple: ('Toyota', 'Farari', 'BMW')
    Total elelmets in tuple is: 3
    Tuple affter addition: ('BMW', 'Farari', 'Toyota', 1, 2, 3, 4, 2, 5)
    Frequency of 2 in tuple: 2

7. Write a program using for loop that loops over a sequence (Print table of any number).

Program:

```
n=int(input("Enter Number: "))

for i in range(1,11):
    print(n," * ", i," = ",i*n)
```

Output:

```
Enter Number: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

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

Program:

```
import time
n=int(input("Enter Number: "))

while n:
    print(n)
    time.sleep(1)
    n-=1
```

Output:

```
Enter Number: 5
5
4
3
2
1
```

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

Program:

```python
def Sieve(n):

    # Create a boolean array "prime[0..n]" and initialize
    #  all entries it as true. A value in prime[i] will
    # finally be false if i is Not a prime, else true.
    sump=0
    prime = [True for i in range(n+1)]
    p = 2
    while (p * p <= n):

        # If prime[p] is not changed, then it is a prime
        if (prime[p] == True):

            # Update all multiples of p
            for i in range(p * p, n+1, p):
                prime[i] = False
        p += 1

    # Print all prime numbers
    for p in range(2, n):
        if prime[p]:
            sump=sump+p
    print(sump)

# driver program
n = 2000000
print ("Following are the prime numbers smaller")
print ("than or equal to", n)
Sieve(n)
```

Output:

Following are the prime numbers smaller than or equal to 2000000
142913828922

10. 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:

```
a, b = 0, 1
total = 0
while True:
    a, b = b, a + b
    if b >= 4000000:
        break
    if b % 2 == 0:
        total += b
print(total)
```

Output:

Sum of all even terms till 4000000 = 4613732

11. Write a program to count the numbers of lower case and upper case characters in the string and store them in a dictionary data structure.

Program:

```
string=input("Enter string: ")

characters={"Lower":0,"Upper":0}
for i in string:
    if i.islower():
        characters["Lower"]+=1
    elif i.isupper():
        characters["Upper"]+=1
print(characters)
```

Output:

```
Enter string: A Quick BROWN Fox 1234
{'Lower': 6, 'Upper': 8}
```

12. 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:

```
x=input("Enter date month and year of birth: ")
date_list=x.split(' ')
print(date_list[1])
date_dict={"Date":" ","Month":" ","Year":" "}
date_dict["Date"]=date_list[0]
date_dict["Month"]=date_list[1]
date_dict["Year"]=date_list[2]

x="-".join(date_dict.values())
print(x)
```

Output:

```
Enter date month and year of birth: 23 October 1989
October
23-October-1989
```

13. 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?

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

Program:

```
with open('File1.txt','r') as f,  open('output.txt', 'w') as fout:

    fout.writelines(reversed(f.readlines()))

with open('output.txt','r') as f:

    print(f.read())
```

Output:

This is line 4

This is line 3

This is line 2

This is line 1

Note:

Where File1.txt contains

This is line 1

This is line 2

This is line 3

This is line 4

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

Program:

```
#opening a file in read mode
with open('File1.txt','r') as f:
    res=f.read()
#counting letters Frequency
mydict={i: res.count(i) for i in set(res)}
print("Frequency of characters present in the file = ",mydict)
#counting words
word=res.split()
print('No of words = ',len(word))
#counting lines
lines=res.split('\n')
print("No of Lines = ",len(lines))
#counting characters
print("No of characters = ",len(mydict))
```

Output:

Frequency of characters present in the file = {'#': 3, 'e': 5, 'k': 1, 'w': 3, '\n': 1, 'T': 1, 'i': 6, 'r': 1, ' ': 17, 'A': 1, 'p': 1, 'y': 3, 'h': 4, 'a': 7, 'L': 1, 'd': 1, 'f': 1, 's': 4, 't': 3, 'u': 1, '.': 2, 'l': 2, 'm': 3, 'n': 9, 'o': 2, 'H': 1, 'g': 2}

No of words = 19

No of Lines = 2

No of characters = 27

16. 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 b.

Program:

```python
def isEditDistanceOne(s1, s2):
    # Find lengths of given strings
    m = len(s1)
    n = len(s2)

    # If difference between lengths is more than 1,
    # then strings can't be at one distance
    if abs(m - n) > 1:
        return false

    count = 0   # Count of isEditDistanceOne

    i = 0
    j = 0
    while i < m and j < n:
        # If current characters dont match
        if s1[i] != s2[j]:
            if count == 1:
                return False

            # If length of one string is
            # more, then only possible edit
            # is to remove a character
            if m > n:
                i+=1
            elif m < n:
                j+=1
            else:    # If lengths of both strings is same
                i+=1
                j+=1

            # Increment count of edits
            count+=1

        else:    # if current characters match
            i+=1
            j+=1

    # if last character is extra in any string
    if i < m or j < n:
        count+=1
```

```python
        return count == 1

    # Driver program
    s1 = "gfg"
    s2 = "gfds"
    if isEditDistanceOne(s1, s2):
        print("Yes")
    else:
        print("No")
```

Output:

No

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

Program: GCD

```
def gcd(a,b):
    if a == 0:
        return b
    return gcd(b % a, a)


# Driver program to test above function
a = 36
b = 15
if(gcd(a, b)):
    print('GCD of', a, 'and', b, 'is', gcd(a, b))
else:
    print('not found')
```

Output:

GCD of 36 and 15 is 3

Program: LCM

a x b = LCM(a, b) * GCD (a, b)

LCM(a, b) = (a x b) / GCD(a, b)

```
def gcd(a,b):
    if a == 0:
        return b
    return gcd(b % a, a)

# Function to return LCM of two numbers
def lcm(a,b):
    return (a*b) / gcd(a,b)

# Driver program to test above function
a = 15
b = 20
print('LCM of', a, 'and', b, 'is', lcm(a, b))
```

Output:

LCM of 15 and 20 is 60

18. Write a program to implement Merge sort.

Program:

```python
# Python program for implementation of MergeSort
def mergeSort(arr):
    if len(arr) > 1:
        mid = len(arr)//2 #Finding the mid of the array
        L = arr[:mid] # Dividing the array elements
        R = arr[mid:] # into 2 halves

        mergeSort(L) # Sorting the first half
        mergeSort(R) # Sorting the second half

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i+=1
            else:
                arr[k] = R[j]
                j+=1
            k+=1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i+=1
            k+=1

        while j < len(R):
```

```python
            arr[k] = R[j]
            j+=1
            k+=1


# Code to print the list
def printList(arr):
    for i in range(len(arr)):
        print(arr[i],end=" ")
    print()


# driver code to test the above code
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print ("Given array is", end="\n")
    printList(arr)
    mergeSort(arr)
    print("Sorted array is: ", end="\n")
    printList(arr)
```

Output:

Given array is

12 11 13 5 6 7

Sorted array is:

5 6 7 11 12 13

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

Program: Selection Sort

```
def selection(arr):
    for i in range(len(arr)):
        temp=arr[i]
        indx=i
        for j in range(i+1,len(arr)):
            if temp>arr[j]:
                indx=j
                temp=arr[j]
        arr[indx]=arr[i]
        arr[i]=temp
    print(arr)

mylist=[21,12,34,43,15,36,52]
print("Before Sorting: ")
print(mylist)
print("After Sorting: ")
selection(mylist)
```

Output:

```
Before Sorting:
[21, 12, 34, 43, 15, 36, 52]
After Sorting:
[12, 15, 21, 34, 36, 43, 52]
```

Program: Insertion Sort

```
def insertion(arr):
    for i in range(1,len(arr)):
        j=i-1
        temp=arr[i]
        while temp<arr[j] and j>=0:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=temp
    print(arr)

mylist=[21,12,34,43,15,36,52]
print("Before Sorting: ")
print(mylist)
print("After Sorting: ")
insertion(mylist)
```

Output:

```
Before Sorting:
```

[21, 12, 34, 43, 15, 36, 52]
After Sorting:
[12, 15, 21, 34, 36, 43, 52]