

# **Geetanjali Institute of Technical Studies**

(Approved by AICTE, New Delhi and Affiliated to Rajasthan Technical University Kota (Raj.))

**DABOK, UDAIPUR, RAJASTHAN 313022**

## **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

### **B. Tech. - VI SEMESTER**



**ACADEMIC YEAR - 2021-22**

## **Python Lab** **(6CS4-23)**

**Submitted to: Mr. Rakshit Kothari**

**Submitted by:**

**Roll No.:**

## **VISSION & MISSION OF INSTITUTE**

### **INSTITUTE VISION**

To achieve excellence in technical and management education through quality teaching and innovation.

### **INSTITUTE MISSION**

**M1:** To provide an excellent learning environment to produce socially responsible and productive technical professionals.

**M2:** To set up the state-of-the-art facilities for quality education and innovation.

**M3:** To impart knowledge & Skills leading to shaping a budding manager as a quality executive.

**M4:** To encourage for life-long learning and team-based problem solving through learning environment.

## **VISION & MISSION OF DEPARTMENT**

### **VISION**

To nurture the students to become employable graduates who can provide solutions to the societal issues through ICT.

### **MISSION**

M1: To focus on practical approach towards learning and exposing the students on the latest ICT technologies.

M2: To foster logical thinking among the students to solve real-time problems using innovative approaches.

M3: To provide state-of-the-art resources that contributes to inculcate ethical and life-long learning environment.

**COURSE OUTCOMES (COs)**

<b>CO1</b>	Describe the Python language syntax including control statements, loops and functions to write programs for a wide variety problem in mathematics, science, and games.
<b>CO2</b>	Examine the core data structures like lists, dictionaries, tuples and sets in Python to store, process and sort the data.
<b>CO3</b>	Interpret the concepts of Object-oriented programming as used in Python using encapsulation, polymorphism and inheritance.
<b>CO4</b>	Discover the capabilities of Python regular expression for data verification and utilize matrices for building performance efficient Python programs.
<b>CO5</b>	Identify the external modules for creating and writing data to excel files and inspect the file operations to navigate the file systems.

# INDEX

---

S.No.	Description	Date	Signature
1.	Write a python program to demonstrate basic datatype in python		
2.	a) Write a python program to calculate distance between two points taking input from the user. b) Write a python program that takes two numbers and prints their sum.		
3.	a) Write a python program to check whether the given number is even or not b) Write a python program, using for loop that prints out the decimal equivalents of $1/2, 1/3, 1/4, \dots, 1/10$		
4.	a) Write a program to demonstrate list in python b) Write a program to demonstrate tuple in python c) Write a program using for loop that loops over a sequence d) Write a program using while loop that asks the user for a number and prints a countdown from that number to zero		
5.	a) Find the sum of all the primes below two million b) By considering the terms in Fibonacci seq. whose value does not exceed 4 million WAP to find the sum of even valued terms		
6.	WAP to count the number of characters in the string and store them in a dictionary data structure		
7.	WAP to count frequency of characters in a given file. Can you use character frequency to test whether the given file is Python program file, C program file or a text file		
8.	a) WAP to print each line of a file in reverse order. b) WAP to compute the number of characters, words and lines in a file		
9.	a) Write a function nearly equal to test whether the two strings are nearly equal. Two strings are nearly equal when a can be generated by a single mutation on. b) Write a program to compute gcd, lcm of two numbers. Each function shouldn't exceed one line.		
10.	a) WAP to implement Insertion sort b) WAP to implement Selection sort c) WAP to implement Merge sort		

**Program 1: Write a python program to demonstrate basic datatype in python**

```
x = 20    #int
print(x)
print(type(x))
x = 20.5  #float
print(x)
print(type(x))
x = 1j    #complex
print(x)
print(type(x))
```

Output:

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
20
<class 'int'>
20.5
<class 'float'>
1j
<class 'complex'>

Process finished with exit code 0
```

**Program 2: a) Write a python program to calculate distance between two points taking input from the user**

```
x1=int(input("enter x1 : "))
x2=int(input("enter x2 : "))
y1=int(input("enter y1 : "))
y2=int(input("enter y2 : "))

result= (((x2 - x1 )**2) + ((y2-y1)**2) )**0.5

print("distance between",(x1,x2),"and",(y1,y2),"is : ", result)
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
enter x1 : 4
enter x2 : 8
enter y1 : 5
enter y2 : 4
distance between (4, 5) and (8, 4) is : 4.123105625617661

Process finished with exit code 0
```

**b) Write a python program that takes two numbers and prints their sum.**

```
a=int(input("enter first number : "))
b=int(input("enter second number : "))

print("sum of", a, "and", b, "is", a + b)
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
enter first number : 5
enter second number : 8
sum of 5 and 8 is 13

Process finished with exit code 0
```

**Program 3: a) Write a python program to check whether the given number is even or not**

```
number=int(input("Enter your number : "))
```

```
if number%2 ==0:
```

```
    print("Number is even")
```

```
else:
```

```
    print("Number is not even ")
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Enter your number : 5
Number is not even

Process finished with exit code 0
```

**b) Write a python program, using for loop that prints out the decimal equivalents of 1/2,1/3,1/4...1/10**

```
for i in range(2,11):
```

```
    print((1/i),end=" ")
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
0.5 0.3333333333333333 0.25 0.2 0.16666666666666666 0.14285714285714285 0.125 0.11111111111111111 0.1

Process finished with exit code 0
```

**Program 4: a) Write a program to demonstrate list in python**

```
List = []  
  
print("empty List: ")  
  
print(List)
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py  
empty List:  
[]  
  
Process finished with exit code 0
```

**b) Write a program to demonstrate tuple in python**

```
t = ()  
  
print (t)  
  
tup = 'python', 'tuple'  
  
print(tup)
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py  
()  
('python', 'tuple')  
  
Process finished with exit code 0
```

**c) Write a program using for loop that loops over a sequence**

```
s=0  
  
for i in range(1,14,3):  
  
    print(i)  
  
    s=s+i  
  
print("sum is =",s)
```



### Output:

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
1
4
7
10
13
sum is = 35

Process finished with exit code 0
```

d) Write a program using while loop that asks the user for a number and prints a countdown from that number to zero

```
num=int(input("Enter your number"))
```

```
while(num>=0):
```

```
    print(num)
```

```
    num=num-1
```

### Output:

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Enter your number 5
5
4
3
2
1
0

Process finished with exit code 0
```

### Program 5:

a) Write a python program to find the sum of all the primes below two million

```
l=[]

for i in range(1,201):

    for j in range(1,i):

        if i%j==0 and i!=j:

            l.append(i)

print("sum : ",sum(l))

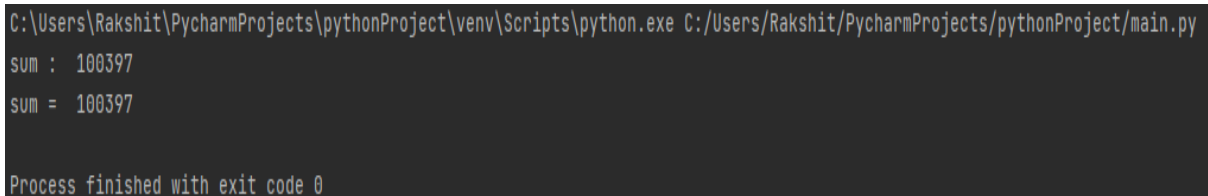
total=0

for i in range(0,len(l)):

    total=total+l[i]

print("sum = ",total)
```

### Output:



```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
sum : 100397
sum = 100397

Process finished with exit code 0
```

b) By considering the terms in Fibonacci sequence whose value does not exceed 4 million WAP to find the sum of even valued terms

```
def fib(n):

    a=0

    b=1

    a1=[]

    a1.append(a)

    a1.append(b)
```

```
s=0

for i in range(1,n+1):

    c=a+b

    a1.append(c)

    a=b

    b=c

print(a1)

for j in a1:

    if j%2==0:

        s=s+j

print("sum of even terms upto",n,"are : ",s)

fib(50)
```

### Output:

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/Py
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765,
sum of even terms upto 50 are : 26658145586

Process finished with exit code 0
```

**Program 6: Write a program to count the number of characters in the string and store them in a dictionary data structure**

```
str=input("enter string : ")
```

```
f = { }
```

```
for i in str:
```

```
    if i in f:
```

```
        f[i] += 1
```

```
    else:
```

```
        f[i] = 1
```

```
print(f)
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
enter string : Hello World
{'H': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'W': 1, 'r': 1, 'd': 1}

Process finished with exit code 0
```

**Program 7: Write a program to count frequency of characters in a given file. Can we use character frequency to test whether the given file is Python program file, C program file or a text file**

```
f="C:\\Users\\Rakshit\\g.txt"
```

```
file = open ( f, "r" )
```

```
a=[]
```

```
b={ }
```

```
for i in file:
```

```
    for j in range(0,len(i)):
```

```
        a.append(i[j])
```

```
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:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
{'H': 1, 'e': 1, 'l': 2, 'o': 1, 'h': 1, 'i': 1}

it is a text file
```

**Program 8: a) Write a program to print each line of a file in reverse order.**

for line in reversed(list(open("C:\\Users\\Rakshit\\g.txt"))):

print(line.rstrip())

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
5. dd
4. cc
3. bb
2. aa
1. Hellohi
Process finished with exit code 0
```

**b) Write a program to compute the number of characters, words and lines in a file**

file\_\_IO = "C:\\Users\\Rakshit\\g.txt"

with open(file\_\_IO, 'r') as f:

data = f.read()

line = data.splitlines()

words = data.split()

spaces = data.split(" ")

charc = (len(data) - len(spaces))

print("\n Line number ::', len(line), '\n Words number ::', len(words), '\n Spaces ::', len(spaces), '\n Characters ::', (len(data)-len(spaces)))

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Line number :: 6
Words number :: 10
Spaces :: 6
Characters :: 30
Process finished with exit code 0
```

**Program 9: Write a function nearly equal to test whether the two strings are nearly equal. Two strings are nearly equal when a can be generated by a single mutation on.**

```
string1 = input("Enter first string: ")
string2 = input("Enter second string: ")

if string1 == string2:

    print("\nBoth strings are equal to each other.")

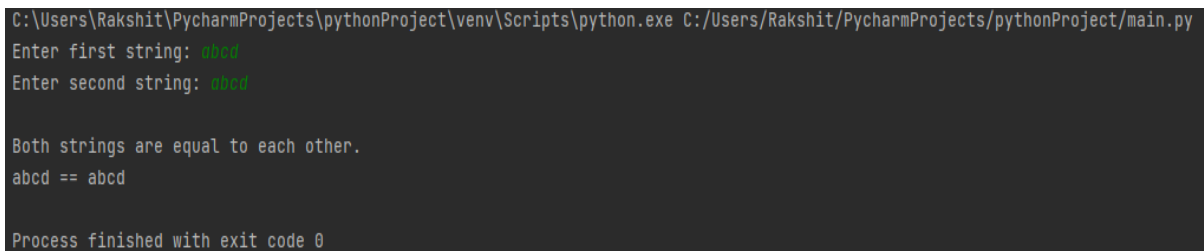
    print(string1,"==",string2);

else:

    print("\nStrings are not equal.")

    print(string1,"!=",string2);
```

**Output:**



```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Enter first string: abcd
Enter second string: abcd

Both strings are equal to each other.
abcd == abcd

Process finished with exit code 0
```

**b) Write a program to compute GCD, LCM of two numbers. Each function should not exceed one line.**

```
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:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
The L.C.M. is 240
GCD of two numbers is 12

Process finished with exit code 0
```



**Program 10: a) Write a program to implement Insertion sort**

```
# Python program for implementation of Insertion Sort

# creating a Function to do insertion sort

def insertion_Sort(arr):

    # Outer Loop to Traverse through 1 to len(arr)

    for i in range(1, len(arr)):

        key = arr[i]

        # Move elements of arr [0..i-1], that are

        # greater than key, to one position ahead of their current position

        j = i-1

        while j >=0 and key < arr[j]:

            arr[j+1] = arr[j]

            j = j-1

        arr[j+1] = key

    return arr

# Driver code to test above

arr = [80, 10, 60, 40, 70, 50]

print("Unsorted array is:", arr)

print ("Sorted array is:",insertion_Sort(arr))
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Unsorted array is: [80, 10, 60, 40, 70, 50]
Sorted array is: [10, 40, 50, 60, 70, 80]

Process finished with exit code 0
```

**b) Write a program to implement Selection sort**

```
def selection_Sort(L):

    # i indicates how many items were sorted

    for i in range(len(L)-1):

        # To find the minimum value of the unsorted segment

        # We first assume that the first element is the lowest

        min_index = i

        # We then use j to loop through the remaining elements

        for j in range(i+1, len(L)-1):

            # Update the min_index if the element at j is lower than it

            if L[j] < L[min_index]:

                min_index = j

        # After finding the lowest item of the unsorted regions, swap with the first unsorted item

        L[i], L[min_index] = L[min_index], L[i]

    return L

# Driver code to test above

L = [80, 10, 60, 40, 70, 50]

print("Unsorted list is:", L)

# Let's see the list after we run the Selection Sort

print("Sorted list is:", selection_Sort(L))
```

**Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Unsorted list is: [80, 10, 60, 40, 70, 50]
Sorted list is: [10, 40, 60, 70, 80, 50]

Process finished with exit code 0
```

**c) Write a program to implement Merge sort**

```
def mergeSort(nlist):

    print("Splitting ",nlist)

    if len(nlist)>1:

        mid = len(nlist)//2

        lefthalf = nlist[:mid]

        righthalf = nlist[mid:]

        mergeSort(lefthalf)

        mergeSort(righthalf)

        i=j=k=0

        while i < len(lefthalf) and j < len(righthalf):

            if lefthalf[i] < righthalf[j]:

                nlist[k]=lefthalf[i]

                i=i+1

            else:

                nlist[k]=righthalf[j]

                j=j+1

            k=k+1

        while i < len(lefthalf):

            nlist[k]=lefthalf[i]

            i=i+1

            k=k+1

        while j < len(righthalf):

            nlist[k]=righthalf[j]
```

```
        j=j+1

        k=k+1

    print("Merging ",nlist)

nlist = [1,2,45,23,24,22]

mergeSort(nlist)

print(nlist)
```

### **Output:**

```
C:\Users\Rakshit\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Rakshit/PycharmProjects/pythonProject/main.py
Splitting  [1, 2, 45, 23, 24, 22]
Splitting  [1, 2, 45]
Splitting  [1]
Merging  [1]
Splitting  [2, 45]
Splitting  [2]
Merging  [2]
Splitting  [45]
Merging  [45]
Merging  [2, 45]
Merging  [1, 2, 45]
Splitting  [23, 24, 22]
Splitting  [23]
Merging  [23]
Splitting  [24, 22]
Splitting  [24]
Merging  [24]
Splitting  [22]
Merging  [22]
Merging  [22, 24]
Merging  [22, 23, 24]
Merging  [1, 2, 22, 23, 24, 45]
[1, 2, 22, 23, 24, 45]

Process finished with exit code 0
```

## Rubrics Evaluation

<b>Performance Criteria</b>	<b>Scale 1 (0-25%)</b>	<b>Scale 2 (26-50%)</b>	<b>Scale 3 (51-75%)</b>	<b>Scale 4 (76-100%)</b>	<b>Score (Numerical)</b>
<b>Understandability</b>  Ability to analyse Problem and Identify solution	Unable to understand the problem.	Able to understand the problem partially and unable to identify the solution	Able to understand the problem completely but unable to identify the solution	Able to understand the problem completely and able to provide alternative solution too.	
<b>Logic</b>  Ability to specify Conditions & control flow that are appropriate for the problem domain.	Program logic is incorrect	Program logic is on the right track but has several errors	Program logic is mostly correct, but may contain an occasional boundary error or redundant or contradictory condition.	Program logic is correct, with no known boundary errors, and no redundant or contradictory conditions.	
<b>Debugging</b>  Ability to execute /debug	Unable to execute program	Unable to debug several errors.	Able to execute program with several warnings.	Able to execute program completely	
<b>Correctness</b>  Ability to code formulae and algorithms that reliably produce correct answers or appropriate results.	Program does not produce correct answers or appropriate results for most inputs.	Program approaches correct answers or appropriate results for most inputs, but can contain miscalculations in some cases.	Program produces correct answers or appropriate results for most inputs.	Program produces correct answers or appropriate results for all inputs tested.	
<b>Completeness</b>  Ability to demonstrate and deliver on time.	Unable to explain the code. And the code was overdue.	Unable to explain the code and the code submission was late.	Able to explain code and the program was delivered within the due date.	Able to explain code and the program was delivered on time.	
<b>TOTAL</b>					