LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Python Programming Lab Manual

I Year II Semester (R20)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

PROGRAMME OUTCOMES (POs):

PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team,

	to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success of organization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyze, design and implement database applications.

Introduction: Language Basics and Example Problems

a) Implement Python Script for checking the given year is leap year or not.

Program:

```
year = int(input("Enter a year: "))
if (year % 4) == 0:
  if (year % 100) == 0:
    if (year % 400) == 0:
      print(year, "is a leap year")
    else:
      print(year, "is not a leap year")
  else:
    print(year, "is a leap year")
else:
  print(year, "is not a leap year")
```

Output:

b) Implement Python Script for finding biggest number among 3 numbers.

Program:

```
a = int(input("Enter Number 1: "))
b = int(input("Enter Number 2: "))
c = int(input("Enter Number 3: "))

if a>b and a>c:
   print(a, "is biggest among the three numbers")
elif b>c:
   print(b, "is biggest among the three numbers")
else:
   print(c, "is biggest among the three numbers")
```

Output:

c) Implement Python Script for displaying reversal of a number.

Program:

```
number = int(input("Enter the integer number:  "))

revs_number = 0

while (number > 0):
   remainder = number % 10
   revs_number = (revs_number * 10) + remainder
   number = number // 10

print("The reverse number is :  ", revs_number)
```

Output:

d) Implement Python Script to check given number is Armstrong or not.

Program:

```
num = int(input("Enter an Integer Number:  "))
no_of_digits = len(str(num))
sum = 0
temp = num

while temp > 0:
  digit = temp % 10
  sum += digit ** no_of_digits
  temp //= 10

if num == sum:
  print(num,"is an Armstrong number")
else:
  print(num,"is not an Armstrong number")
```

Output:

e) Implement Python Script to print sum of N natural numbers.

Program:

```
num = int(input("Enter an Integer to print sum of numbers up to num:  "))

if num < 0:
  print("Enter a positive number")
else:
  sum = 0
  while(num > 0):
     sum += num
     num -= 1
  print("The sum is", sum)
```

Output:

f) Implement Python Script to check given number is palindrome or not.

Program:

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

temp=number
reverse_num=0

while(number>0):
   digit=number%10
   reverse_num=reverse_num*10+digit
   number=number//10

if(temp==reverse_num):
   print("The number is palindrome!")
else:
   print("Not a palindrome!")
```

Output:

g) Implement Python script to print factorial of a number.

Program:

```
num = int(input("Enter a Number to find Factorial:  "))
factorial = 1

if num < 0:
  print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
  print("The factorial of 0 is 1")
else:
  for i in range(1,num + 1):
     factorial = factorial*i
  print("The factorial of ",num," is ",factorial)
```

Output:

h) Implement Python Script to print all prime numbers within the given range.

Program:

```
lower_range = int(input("Enter Lower Range: "))
upper_range = int(input("Enter Upper Range:  "))

for num in range(lower_range,upper_range +  1):
  if num > 1:
     for i in range(2, num):
        if (num % i) == 0:
           break
     else:
        print(num, end=",")
```

Output:

i) Implement Python Script to calculate the series: $S=1+x+x^2+x^3+.......x^n$

Program:

```
sum = 1
print("Here we will calculate the sum of series 1+x+x^2+.....+x^n")
x=int(input("Enter the value of x:  "))
n = int(input("Enter the value of n: "))
for i in range(1,n+1):
   sum = sum +(x**i)
print("The sum of the series is:", sum)
```

Output:

j) Implement Python Script to print the following pattern:

```
  *
 * *
* * *
```

Program:

```
n = int(input("Enter a Number to build Pyramid: "))
k = n - 1

for i in range(0, n):
   for j in range(0, k):
      print(end=" ")
   k = k - 1
   for j in range(0, i+1):
      print("* ", end="")
   print("\r")
```

Output:

Module 1: Exercise Programs on Lists

a) Write a Python script to display elements of list in reverse order.

Program:

```
systems = ['Windows', 'macOS', 'Linux']

print('Original List: ', systems)

systems.reverse()

print('Updated List: ', systems)
```

Output:

b) Write a Python script to find the minimum and maximum elements without using built-in operations in the lists.

Program:

```
x = []
n = int(input("How many elements you want to enter in a list? "))

for i in range(0, n):
   a = int(input("Enter a Number to be added to list: "))
   x.append(a)

max_val = x[0]
min_val = x[0]

for check in x:
   if check > max_val:
      max_val = check

for check in x:
   if check < min_val:
      min_val = check


print("Maximum Number in the List is: ", max_val)
print("Minimum Number in the List is:", min_val)
```

Output:

c) Write a Python script to remove duplicates from a list.

Program:

```
x = []

n = int(input("How many elements you want to enter in a list? "))

for i in range(0, n):
   a = int(input("Enter a Number to be added to list:  "))
   x.append(a)

new_list = []

for num in x:
   if num not in new_list:
      new_list.append(num)

print(new_list)
```

Output:

d) Write a Python script to append a list to the second list.

Program:

```
x = []
m = int(input("How many elements you want to enter in a list x? "))
for i in range(0, m):
   a = int(input("Enter a Number to be added to list x: "))
   x.append(a)

n = int(input("How many elements you want to enter in a list y? "))
y = []
for i in range(0, n):
   b = int(input("Enter a Number to be added to list y:  "))
   y.append(b)

new_list = x + y

print(new_list)
```

Output:

e) Write a Python script to count the number of strings in a list where the string length is 2 or more.

Program:

```
x = []
count = 0
m = int(input("How many strings you want to enter in a list x? "))
for i in range(0, m):
    a = input("Enter a String to be added to list x:  ")
    x.append(a)
    if len(x[i]) >= 2:
        count = count + 1

print("Total Strings whose length is >2 in the List are:  ", count)
```

Output:

Module 2: Exercise Programs on Tuples

a) Write a Python script to create a tuple with different data types.

Program:

```
tuplex = ("tuple", False, 3.2, 1)
print(tuplex)
```

Output:

b) Write a Python script to find the repeated items of a tuple.

Program:

```
tuplex = 2, 4, 5, 6, 2, 3, 4, 4, 7
print(tuplex)
count = tuplex.count(4)
print(count)
```

Output:

c) Write a Python script to replace last value of tuples in a list.

Program:

```
l = [(10, 20, 40), (40, 50, 60), (70, 80, 90)]
print([t[:-1] + (100,) for t in l])
```

Output:

d) Write a Python script to sort a tuple by its float element.

Program:

```
price = [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]
print( sorted(price, key=lambda x: float(x[1]), reverse=True))
```

Output:

Module 3: Exercise Programs on Sets and Dictionaries

a) Write a Python script to add member(s) in a set.

Program:

```
color_set = set()
color_set.add("Red")
color_set.update(["Blue", "Green"])
print(color_set)
```

Output:

b) Write a Python script to perform Union, Intersection, difference and symmetric difference of given two sets.

Program:

```
A = {0, 2, 4, 6, 8}
B = {1, 2, 3, 4, 5}
print("Union :", A | B)
print("Intersection :", A & B)
print("Difference :", A - B)
print("Symmetric difference :", A ^  B)
```

Output:

c) Write Python script to test whether every element in S is in T and every element in T is in S.

Program:

```
S = set(["apple", "mango"])
T = set(["mango", "banana"])
print(S <= T)
print(S >= T)
print(S == T)
print(S != T)
```

Output:

d) Write a Python script to sort (ascending and descending) a dictionary by value.

Program:

```
dt = {8:4, 1:6, 6:3}

asc = {key: value for key, value in sorted(dt.items(), key=lambda item: item[1])}

des = {key: value for key, value in sorted(dt.items(), key=lambda item: item[1], reverse=True)}

print(asc)
print(des)
```

Output:

e) Write a Python script to check whether a given key already exists or not in a dictionary.

Program:

```
week = {'Mon':1,'Tue':2,'Wed':3,'Thu':4, 'Fri':5, 'Sat':6, 'Sun':7}
print("Week Dictionary : ", week)
check_key = input("Enter a key to be verified in Dictionary: ")
if check_key in week:
  print(check_key," is Present.")
else:
  print(check_key, " is not Present.")
```

Output:

f) Write a Python script to concatenate following dictionaries to create a new one:

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

Program:

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3):
   dic4.update(d)
print(dic4)
```

Output:

g) Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.

Program:

```
d=dict()
for x in range(1,16):
   d[x]=x**2
print(d)
```

Output:

h) Write a Python program to map two lists into a dictionary.

Program:

```
keys = ['red', 'green', 'blue']
values = ['#FF0000','#008000', '#0000FF']
color_dictionary = dict(zip(keys, values))
print(color_dictionary)
```

Output:

Module 4: Exercise Programs on Functions and Recursion

a) Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.

Program:

```
n = [int(input("Enter Number %d : " %(i+1))) for i in range(3)]
def max_of_three(a,b,c):
   if(a>b and a>c):
      return a
   elif(b>c):
      return b
   else:
      return c
print("Maximum Number is: ", max_of_three(n[0], n[1], n[2]))
```

Output:

b) Write a program which makes use of function to display all such numbers which are divisible by 7 but are not a multiple of 5, between given range X and Y.

Program:

```
def numbers(a,b):
    x = [i for i in range(a,b+1) if(i%7 == 0 and i%5 != 0)]
    print("Numbers divisible by 7, but not by 5 between %d and %d are:" %(a,b))
    print(x)

start_range = int(input("Enter the Starting Range: "))

end_range = int(input("Enter the Ending Range: "))

numbers(start_range, end_range)
```

Output:

c) Define functions to find mean, median, mode for the given numbers in a list.

Program:

```
def mean(n, a):
    return sum(a)/n

def median(n,a):
    a.sort()
    if(n%2 == 0):
        return (a[(n-1)//2] + a[n//2])/2
    else:
        return a[n//2]

def mode(a):
    c = [a.count(i) for i in a]
    b = [i for i in range(len(c)) if(c[i] == max(c))]
    return b[0]

n = int(input("How many elements you want to enter into List? "))
a = [int(input("Enter Element %d: " %(i+1))) for i in range(n)]
print("Mean = %.2f" %(mean(n,a)))
print("Median = %.2f" %(median(n,a)))
print("Mode = %d" %(a[(mode(a))]))
```

Output:

d) Define a function which generates Fibonacci series up to n numbers.

Program:

```
def fib(n):
    a = 0
    b = 1
    if n == 1:
        print(a)
    else:
        print(a)
        print(b)
        for i in range(2,n):
            c = a + b
            a = b
            b = c
            print(c)
num = int(input("Number that you want to Print Fibonacci Sequence: "))
fib(num)
```

Output:

e) Implement a python script for factorial of number by using recursion.

Program:

```
def recur_factorial(n):
  if n == 1:
     return n
  else:
     return n*recur_factorial(n-1)

num = int(input("Enter a number:  "))

if num < 0:
  print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
  print("The factorial of 0 is 1")
else:
  print("The factorial of",num,"is",recur_factorial(num))
```

Output:

f) Implement a python script to find GCD of given two numbers using recursion.

Program:

```
def gcd_fun (x, y):
   if (y == 0):
      return x
   else:
      return gcd_fun (y, x % y)
x =int (input ("Enter the first number:  "))
y =int (input ("Enter the second number: "))
num = gcd_fun(x, y)
print("GCD of two number is:  ", num)
```

Output:

Module 5: Exercise Programs on Date and Time Modules

a) Write a Python script to get the current time in Python.

Program:

```
import datetime
now = datetime.datetime.now()
print ("Current Time : ", now.strftime("%H:%M:%S"))
```

Output:

b) Write a Python script to get current time in milliseconds in Python.

Program:

```
import time
milli_sec = int(round(time.time() * 1000))
print(milli_sec)
```

Output:

c) Write a Python script to print next 5 days starting from today.

Program:

```
import datetime
base = datetime.datetime.today()
for x in range(0, 5):
    print(base + datetime.timedelta(days=x))
```

Output:

Module 6: Exercise Programs on Exception Handling

a) Write a Python script to handle simple errors by using exception handling mechanism.

Program:

```
try:
   a = int(input("Enter a:"))
   b = int(input("Enter b:"))
   c = a/b
   print("a/b = %d"%c)
except Exception:
   print("can't divide by zero")
   print(Exception)
else:
   print("Hi I am else block")
```

Output:

b) Write a Python script to handle multiple errors with one except statement.

Program:

```
try:
  a = int(input("Enter the Value of a: "))
  b = int(input("Enter the Value of b:  "))
  c = a/b
  print(c)
except (TypeError, ZeroDivisionError, ValueError):
  print("There is issue with your value's type or zero division error")
```

Output:

Module 7: Exercise Programs on Strings

a) Implement Python Script to perform various operations on string using string libraries.

Program:

```
name = "Ram Kumar"
name_1 = "Ram_Kumar"
print(name.split())
print(name_1.split('_'))
print(name.isalnum())
print(name.isdigit())
print(name_1.lower())
print(name_1.upper())
print(name.capitalize())
print(len(name))
print(name_1.title())
```

Output:

b) Implement Python Script to check given string is palindrome or not.

Program:

```
my_str = input("Enter any string: ")
my_str = my_str.casefold()
rev_str = reversed(my_str)

if list(my_str) == list(rev_str):
  print("The string is a palindrome.")
else:
  print("The string is not a palindrome.")
```

Output:

c) Implement python script to accept line of text and find the number of characters, number of vowels and number of blank spaces in it.

Program:

```
str1 = input("Please Enter Your Own String :  ")
vowels = 0
consonants = 0

for i in str1:
   if(i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u'
     or i == 'A' or i == 'E' or i == 'I' or i == 'O' or i == 'U'):
      vowels = vowels + 1
   else:
      consonants = consonants + 1

print("Total Number of Vowels in this String = ", vowels)
print("Total Number of Consonants in this String = ", consonants)
```

Output:

d) Implement python script that takes a list of words and returns the length of the longest one.

Program:

```
def find_longest_word(words_list):
   word_len = []
   for n in words_list:
      word_len.append((len(n), n))
   word_len.sort()
   return word_len[-1][0], word_len[-1][1]
result = find_longest_word(["PHP", "Exercises", "Backend"])
print("\nLongest word:  ",result[1])
print("Length of the longest word: ",result[0])
```

Output:

Module 8: Exercise Programs on Regular Expressions

a) Write a Python script to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

Program:

```
import re
def is_allowed_specific_char(string):
   charRe = re.compile(r'[^a-zA-Z0-9.]')
   string = charRe.search(string)
   return not bool(string)

string = input("Enter a String to verify: ")

print(is_allowed_specific_char(string))
```

Output:

b) Write a Python script to check whether password is valid or not.

Conditions for a valid password are:

1. **Should have at least one number.**
2. **Should have at least one uppercase and one lowercase character.**
3. **Should have at least one special symbol.**
4. **Should be between 6 to 20 characters long.**

Program:

```
import re
password = input("Enter a Password: ")
x = True
while x:
  if (len(password)<6 or len(password)>12):
    break
  elif not re.search("[a-z]", password):
    break
  elif not re.search("[0-9]", password):
    break
  elif not re.search("[A-Z]", password):
    break
  elif not re.search("[$#@]", password):
    break
  elif re.search("\s", password):
    break
  else:
    print("You Password is Valid")
    x=False
    break
if x:
  print("Your Password is INVALID")
```

Output:

Module 9: Exercise Programs on Object Oriented Programming

a) Write a Python script to create and access class variables and methods.

Program:

```
class Bike:
  def __init__(self, name, year):
    self.name = name
    self.year = year

  def bikename(self):
    print("Bike Name is: ", self.name)

  def madeyear(self):
    print("This bike was Made in the Year:  ", self.year)

b = Bike("Glamour", 2019)
b.bikename()
b.madeyear()
```

Output:

b) Write a Python script to implement single inheritance.

Program:

```
class Base:
   def p(self ):
      print("Hi")

class D(Base):
   def g(self):
      print("Hello")

sd = D()
sd.g()
sd.p()
```

Output:

c) Write a Python script to implement encapsulation.

Program:

```
class Counter:
   def _init_(self):
      self.__current = 0

   def increment(self):
      self._current = self._current + 1

   def value(self):
      return self.__current

   def reset(self):
      self.__current = 0

c = Counter()
c.__current = 999
c.increment()
print(c.Counter_current)
```

Output:

d) Write a Python script to implement method overriding.

Program:

```
from math import pi

class Shape:

  def area(self):
    print("Hi")

  def fact(self):
    return "I am a two-dimensional shape."

class Square(Shape):
  def __init__(self, length):
    self.length = length

  def area(self):
    return self.length**2

  def fact(self):
    return "Squares have each angle equal to 90 degrees."



class Circle(Shape):
  def __init__(self, radius):
    self.radius = radius
    def area(self):
        return pi*self.radius**2



a = Square(4)
b = Circle(7)
print(b.fact())
print(a.fact())
print(b.area())
```

Output:

Module 10: Exercise Programs on Python Libraries – NumPy, Pandas , Matplotlib

a) Write a NumPy program to generate a matrix product of two arrays.

Program:

```
import numpy as np
x = [[1, 0], [1, 1]]
y = [[3, 1], [2, 2]]
print("Matrix product of above two arrays:")
print(np.matmul(x, y))
```

Output:

b) Write a NumPy program to create a random array with 1000 elements and compute the average, variance, standard deviation of the array elements.

Program:

```
import numpy as np
x = np.random.randn(1000)
print("Average of the array elements:", x.mean())
print("Standard deviation of the array elements:", x.std())
print("Variance of the array elements:", x.var())
```

Output:

c) Demonstrate how to download dataset and how to create DataFrame.

i. **Write a Pandas program to get the first 3 rows of a DataFrame.**
ii. **Write a Pandas program to select the specified columns and rows from a given DataFrame.**
iii. **Write a Pandas program to select the rows where the score is missing, i.e., is NaN.**
iv. **Write a Pandas program to insert a new column in existing DataFrame.**

Procedure:

1) Save your data either in the JSON format or CSV format and can read into program using the read_json( ) / read_csv( ) methods.
2) Create a DataFrame using the above said methods by opening the file and store into a variable.
3) Download the dataset from https://tinyurl.com/SampleDataLab

*****Note: For the above programs, consider that your data is in "sample_dataset.csv" file, which is available in the path E:/dataset/sample_dataset.csv**

Program:

i. Write a Pandas program to get the first 3 rows of a DataFrame.

```
import pandas as pd

df = pd.read_csv('E:/dataset/sample_dataset.csv')

print(df)
```

Output:

ii. Write a Pandas program to select the specified columns and rows from a given DataFrame.

Using the iloc( ) in the Python helps us in selecting the specified rows and columns from a DataFrame.

Syntax: ***dataframe.iloc[[row_numbers], [column_numbers]]***

Program:

```
import pandas as pd

df = pd.read_csv('E:/dataset/sample_dataset.csv')

print(df.iloc[[1,3,5,6],[1, 2]])
```

Output:

iii. Write a Pandas program to select the rows where the score is missing, i.e., is NaN.

Program:

```
import pandas as pd

df = pd.read_csv('E:/dataset/sample_dataset.csv')

print(df[df['Calories'].isnull( )])
```

Output:

iv. Write a Pandas program to insert a new column in existing DataFrame.

Program:

```
import pandas as pd

df = pd.read_csv('E:/dataset/sample_dataset.csv')

df['Weight'] = 60

print(df)
```

Output:

d) Write a Python programming to display a bar chart using different color for each bar.

Program:

```
import matplotlib.pyplot as plt

marks = [79, 82, 75, 96]

subjects = ['C', 'JAVA', 'Python', 'WT']

plt.xlabel("Subjects")

plt.ylabel("Obtained Marks")

plt.title("Student Marks")

plt.bar(subjects, marks, color=['r', 'g', 'b', 'm'])

plt.show()
```

Output:

e) Write a Python programming to create a pie chart with a title.

Program:

```
import matplotlib.pyplot as plt

marks = [79, 82, 75, 96]

subjects = ['C', 'JAVA', 'Python', 'WT']

plt.pie(marks, labels=subjects, autopct =  '%1.1f%%')

plt.title("Student Marks")

plt.legend()

plt.show()
```

Output: