SVR ENGINEERING COLLEGE

AYYALURUMETTA (V), NANDYAL, KURNOOL DT.
ANDHRA PRADESH – 518502

2020 – 2021

LABORATORY MANUAL

OF

PYTHON PROGRAMMING LABORATORY

(19A05304P)

(R-19 REGULATION)

Prepared by

Mr. G.MAHESH BABU
Asst. Professor

For

B.Tech II YEAR – I SEM. (CSE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SVR ENGINEERING COLLEGE

(AFFILIATED TO JNTUA ANANTHAPURAM- AICITE-INDIA)
AYYALURUMETTA (V), NANDYAL, KURNOOL DT.
ANDHRA PRADESH – 518502

LAB MANUAL CONTENT

PYTHON PROGRAMMING LABORATORY

(19A05304P**)**

1. Institute Vision & Mission, Department Vision & Mission
2. PO, PEO& PSO Statements.
3. List of Experiments
4. CO-PO Attainment
5. Experiment Code and Outputs

1. Institute Vision & Mission, Department Vision & Mission

Institute Vision:

To produce Competent Engineering Graduates & Managers with a strong base of Technical & Managerial Knowledge and the Complementary Skills needed to be Successful Professional Engineers & Managers.

Institute Mission:

To fulfill the vision by imparting Quality Technical & Management Education to the Aspiring Students, by creating Effective Teaching/Learning Environment and providing State – of the – Art Infrastructure and Resources.

Department Vision:

To produce Industry ready Software Engineers to meet the challenges of 21st Century.

Department Mission:

- Impart core knowledge and necessary skills in Computer Science and Engineering through innovative teaching and learning methodology.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society.
- Cultivate the students with all-round competencies, for career, higher education and self-employability.

2. PO, PEO& PSO Statements

PROGRAMME OUTCOMES (POs)

PO-1: Engineering knowledge - Apply the knowledge of mathematics, science, engineering fundamentals of Computer Science& Engineering to solve complex real-life engineering problems related to CSE.

PO-2: Problem analysis - Identify, formulate, review research literature, and analyze complex engineering problems related to CSE and 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 related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, 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 - Select/Create and apply appropriate techniques, resources and modern engineering and IT tools and technologies for rapidly changing computing needs, 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 CSE professional engineering practice.

PO-7: Environment and Sustainability - Understand the impact of the CSE 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 relevant engineering practices.

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 the society-at-large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, 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 broadcast context of technological changes.

Program Educational Objectives (PEOs):

PEO 1:Graduates will be prepared for analyzing, designing, developing and testing the software solutions and products with creativity and sustainability.

PEO 2: Graduates will be skilled in the use of modern tools for critical problem solvingand analyzing industrial and societal requirements.

PEO 3:Graduates will be prepared with managerial and leadership skills for career and starting up own firms.

Program Specific Outcomes (PSOs):

PSO 1: Develop creative solutions by adapting emerging technologies / tools for real time applications.

PSO 2: Apply the acquired knowledge to develop software solutions and innovative mobile apps for various automation applications

2.1 Subject Time Table

SVR ENGINEERING COLLEGE::NANDYAL DEPARTMENT OF CSE								
G.MAHESH BABU						**II-IIIRD SEM**		
Day/ Time	**9:30 AM**	**10:20 AM**	**11:30 AM**	**12:20 PM-**		**02:00 PM**	**02:50 PM**	**03:40 PM**
	10:20 AM	**11:10AM**	**12:20 PM**	**01:10 PM**		**02:50 PM**	**03:40 PM**	**04:30 PM**
MON								
TUE								
WED								
THU								
FRI						**PP LAB**		
SAT								

LIST OF EXPERIMENTS (SYLLABUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
B.Tech – II-IIIRD Sem
19A05304P PYTHON PROGRAMMING LABORATORY
(Common to CSE & IT)

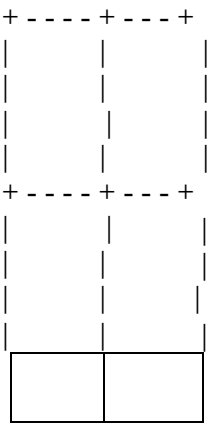
Course Objectives:

- To understand the object-oriented concepts using Python in problem solving.
- To understand the fundamentals of Python programming concepts and its applications.
- To elucidate solving mathematical problems using Python programming language
- To train the students in solving computational problems

Laboratory Experiments

1. Install Python Interpreter and use it to perform different Mathematical Computations. Try to do all the operations present in a Scientific Calculator

2. Write a function that draws a grid like the following:



3. Write a function that draws a Pyramid with # symbols

```
#
# # #
# # # # #
# # # # # # #
```

4. Using turtles concept draw a wheel of your choice

5. Write a program that draws Archimedean Spiral

6. The letters of the alphabet can be constructed from a moderate number of basic elements, like vertical and horizontal lines and a few curves. Design an alphabet that can be drawn with a minimal number of basic elements and then write functions that draw the letters. The alphabet can belong to any Natural language excluding English. You should consider at least Ten letters of the alphabet.

7.The time module provides a function, also named time that returns the current Greenwich Mean Time in "the epoch", which is an arbitrary time used as a reference point. On UNIX systems, the epoch is 1 January 1970. >>> import time >>> time.time() 1437746094.573595Write a script that reads the current time and converts it to a time of day in hours, minutes, and seconds, plus the number of days since the epoch.

8. Given n+r+1 <= 2r . n is the input and r is to be determined. Write a program which computes minimum value of r that satisfies the above.

9. Write a program that evaluates Ackermann function

10. Choose any five built-in string functions of C language. Implement them on your own in Python. You should not use string related Python built-in functions.

11. . Given a text of characters, Write a program which counts number of vowels, consonants and special characters.

12. Given a word which is a string of characters. Given an integer say 'n', Rotate each character by 'n' positions and print it. Note that 'n' can be positive or negative.

13. Given rows of text, write it in the form of columns.

14. Given a page of text. Count the number of occurrences of each latter (Assume case insensitivity and don't consider special characters). Draw a histogram to represent the same

15. Write program which performs the following operations on list's. Don't use built-in functions

a) Updating elements of a list
b) Concatenation of list's
c) Check for member in the list
d) Insert into the list
e) Sum the elements of the list
f) Push and pop element of list
g) Sorting of list
h) Finding biggest and smallest elements in the list
i) Finding common elements in the list

16. Write a program that reads a file, breaks each line into words, strips whitespace and punctuation from the words, and converts them to lowercase.

17. Write a program illustrating the object oriented features supported by Python

18. Design a Python script using the Turtle graphics library to construct a turtle bar chart representing the grades obtained by N students read from a file categorising them into distinction, first class, second class, third class and failed.

19. Design a Python script to determine the difference in date for given two dates in YYYY:MM:DD format(0 <= YYYY <= 9999, 1 <= MM <= 12, 1 <= DD <= 31) following the leap year rules.

20. Design a Python Script to determine the time difference between two given times in HH:MM:SS format.(0 <= HH <= 23, 0 <= MM <= 59, 0 <= SS <= 59)

5.0 EXPERIMENT CODE AND OUTPUTS

EXPERIMENT NO: 1

AIM: Install Python Interpreter and use it to perform different Mathematical Computations. Try to do all the operations present in a Scientific Calculator

```
number_1 = int(input('Enter your first number: '))
number_2 = int(input('Enter your second number: '))
# Addition
print('{} + {} = '.format(number_1, number_2))
print(number_1 + number_2)
# Subtraction
print('{} - {} = '.format(number_1, number_2))
print(number_1 - number_2)
# Multiplication
print('{} * {} = '.format(number_1, number_2))
print(number_1 * number_2)
# Division
print('{} / {} = '.format(number_1, number_2))
print(number_1 / number_2)
```

```
import math
print("""
press -
   1 - Addition(x, y)
   2 - subtraction(x,y)
   3-multiplication(x,y)
   4 - division(x, y)
   5- exponent(x, y)
   6 - tan(x, y)
   7 - sin(x)
   8 - cos(x)
9 - factorial(x)
10 - log(x)
""")

o = input("")
if o == "1":
x = int(input("1st number -"))
y = int(input("2nd number -"))
add(x, y)
print(x + y)
elif o == "2":
x = int(input())
y = int(input())
subtract(x, y)
print(x-y)
elif o == "3":
x = int(input())
```

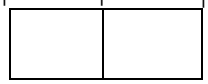
```
y = int(input())
print(x*y)
elif o == "4":
x = int(input())
y = int(input())
print(x/y)
elif o == "5":
x = int(input())
y = int(input())
print(x**y)
elif o == "6":
x = int(input())
y = int(input())
print(math.tan(x))
elif o == "7":
x = int(input())

print(math.sin(x))
elif o == "8":
x = int(input())
cos(x)
elif o == "9":
x = int(input())
print(math.factorial(x))
elif o == "10":
x = int(input())
print(math.log(x))
```

OUTPUT:

EXPERIMENT NO: 2

AIM: . Write a function that draws a grid like the following:

```
+ - - - - + - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - +
|         |         |
|         |         |
|         |         |
|         |         |
```



```
def grid(row, col):
   x = ('+--- ' * col + '+')
   y = ('\n' +'|    ' *  (col+1))
   return ((x + 4*y) +'\n')*row + x
print(grid(2,2))
```

OUTPUT:

EXPERIMENT NO: 3

AIM: Write a function that draws a Pyramid with # symbols

```
#
###
#####
#######
```

```
n = 0
r = 12
for m in range(1, r+1):
    for gap in range(1, (r-m)+1):
        print(end=" ")
    while n != (2*m-1):
        print("# ", end="")
        n = n + 1
    n = 0
    print()
```

OUTPUT:

EXPERIMENT NO: 4

AIM: Using turtles concept draw a wheel of your choice

```
# Python program to draw square
# using Turtle Programming
import turtle
skk = turtle.Turtle()

for i in range(4):
    skk.forward(50)
    skk.right(90)

turtle.done()
```

```
# Python program to draw star
# using Turtle Programming
import turtle
star = turtle.Turtle()

star.right(75)
star.forward(100)

for i in range(4):
    star.right(144)
    star.forward(100)
turtle.done()
```

```
# Python program to draw hexagon
# using Turtle Programming
import turtle
polygon = turtle.Turtle()

num_sides = 6
side_length = 70
angle = 360.0 / num_sides

for i in range(num_sides):
    polygon.forward(side_length)
    polygon.right(angle)

turtle.done()
```

OUTPUT:

EXPERIMENT NO: 5

AIM: Write a program that draws Archimedean Spiral

```
def spiral(turtle, rotations=6, a=0.0, b=5):
   theta = 0.0

   while theta < rotations * 2 * pi:
      radius = a + b * theta
      x, y = radius * cos(theta), radius * sin(theta)
      turtle.goto(x, y)
      theta += 0.1
```

OUTPUT:

EXPERIMENT NO: 6

AIM: The letters of the alphabet can be constructed from a moderate number of basic elements, like vertical and horizontal lines and a few curves. Design an alphabet that can be drawn with a minimal number of basic elements and then write functions that draw the letters. The alphabet can belong to any Natural language excluding English. You should consider at least Ten letters of the alphabet.

```
from __future___
import
print_function,
division
                    import string
                    import turtle
                    """
                    To use this typewriter, you have to provide a module named letters.py
                    that contains functions with names like draw_a, draw_b, etc.
                    """
                    # check if the reader has provided letters.py
                    try:
                        import letters
                    except ImportError as e:
                        message = e.args[0]
                        if message.startswith('No module'):
                            raise ImportError(message +
                                              '\nYou have to provide a module named
                    letters.py')
                    def teleport(t, x, y):
                        """Moves the turtle without drawing a line.
                        Postcondition: pen is down
                        t: Turtle
                        x: coordinate
                        y: coordinate
                        """
                        t.pu()
                        t.goto(x, y)
                        t.pd()
                    def keypress(char):
                        char: string, letter to draw
                        """
                        # if we're still drawing the previous letter, bail out
                        if bob.busy:
                            return
                        else:
                            bob.busy = True
                        # figure out which function to call, and call it
                        try:
                            name = 'draw_' + char
                            func = getattr(letters, name)
                        except AttributeError:
                            print("I don't know how to draw an", char)
                            bob.busy = False
                            return
                        func(bob, size)
                        letters.skip(bob, size/2)
                        bob.busy = False
                    def carriage_return():
                        """Moves to the beginning of the next line.
                        """
                        teleport(bob, -180, bob.ycor() - size*3)
                        bob.busy = False
                    def presser(char):
                        def func():
                            keypress(char)
                        return func
```

```
# create and position the turtle
size = 20
bob = turtle.Turtle()
bob.busy = False
teleport(bob, -180, 150)
# tell world to call keypress when the user presses a key
screen = bob.getscreen()
for char in string.ascii_lowercase:
    screen.onkey(presser(char), char)
screen.onkey(carriage_return, 'Return')
screen.listen()
turtle.mainloop()
```

OUTPUT:

EXPERIMENT NO: 7

AIM: The time module provides a function, also named time that returns the current Greenwich Mean Time in "the epoch", which is an arbitrary time used as a reference point. On UNIX systems, the epoch is 1 January 1970.

>>> import time
>>> time.time()
1437746094.5735958

```
import time

epoch = time.time()
seconds_in_a_day = 24 * 60 * 60
seconds_in_an_hour = 60 * 60
seconds_in_a_minute = 60

days = epoch // seconds_in_a_day
hours = (epoch % seconds_in_a_day) // seconds_in_an_hour + 8
minutes = (epoch % seconds_in_a_day) % seconds_in_an_hour // seconds_in_a_minute
seconds = (epoch % seconds_in_a_day) % seconds_in_an_hour % seconds_in_a_minute
print("%s: %s: %s: %s" %(days, hours, minutes, seconds))
print("Beijing Current time is %d: %d: %d: %d" %(days, hours, minutes, seconds"))
```

OUTPUT:

EXPERIMENT NO: 8(A)

AIM: Given $n+r+1 <= 2_r$. n is the input and r is to be determined. Write a program which computes minimum value of r that satisfies the above.

```
# Python3 Program to find
# minimum value of X
# in equation X = P * A + Q * B
 # Function to return gcd of a and b

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

a = 2
b = 4
print(gcd(a, b))
```

OUTPUT:

EXPERIMENT NO: 8(B)

AIM: Given $n+r+1 <= 2_r$. n is the input and r is to be determined. Write a program which computes minimum value of r that satisfies the above.

```
def solution (a, b, n):

    # traverse for all possible values
    i = 0
    while i * a <= n:

        # check if it is satisfying
        # the equation
        if (n - (i * a)) % b == 0:
            print("x = ",i ,", y = ",
               int((n - (i * a)) / b))
            return 0
        i = i + 1

    print("No solution")

# driver program to test the above function
a = 2
b = 3
n  =  7
solution(a, b, n)
```

OUTPUT:

EXPERIMENT NO: 9

AIM: Write a program that evaluates Ackermann function

```
# Python program to illustrate Ackermann function

def A(m, n, s ="% s"):
    print(s % ("A(% d, % d)" % (m, n)))
    if m == 0:
        return n + 1
    if n == 0:
        return A(m - 1, 1, s)
    n2 = A(m, n - 1, s % ("A(% d, %% s)" % (m - 1)))
    return A(m - 1, n2, s)

print(A(1, 2))
```

OUTPUT:

EXPERIMENT NO: 10

AIM: . Choose any five built-in string functions of C language. Implement them on your own in Python. You should not use string related Python built-in functions.

```
defining strings in Python
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
#Accessing string characters in Python
str = 'programiz'
print('str = ', str)

#first character
print('str[0] = ', str[0])

#last character
print('str[-1] = ', str[-1])

#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])

#slicing 6th to 2nd last character
print('str[5:-2] = ', str[5:-2])
```

EXPERIMENT NO: 11

AIM: Write a program which counts number of vowels, consonants and special characters.

```
# Python3 Program to count vowels,
# consonant, digits and special
# character in a given string.

# Function to count number of vowels,
# consonant, digits and special
# character in a string.
def countCharacterType(str):

    # Declare the variable vowels,
    # consonant, digit and special
    # characters
    vowels = 0
    consonant = 0
    specialChar = 0
    digit = 0

    # str.length() function to count
    # number of character in given string.
    for i in range(0, len(str)):

        ch = str[i]

        if ( (ch >= 'a' and ch <= 'z') or
             (ch >= 'A' and ch <= 'Z') ):

            # To handle upper case letters
            ch = ch.lower()

            if (ch == 'a' or ch == 'e' or ch == 'i'
                        or ch == 'o' or ch == 'u'):
                vowels += 1
            else:
                consonant += 1

        elif (ch >= '0' and ch <= '9'):
            digit += 1
        else:
            specialChar += 1

    print("Vowels:", vowels)
    print("Consonant:", consonant)
    print("Digit:", digit)
    print("Special Character:", specialChar)


# Driver function.
str = "geeks for geeks121"
countCharacterType(str)
```

EXPERIMENT NO: 12

AIM: Given a word which is a string of characters. Given an integer say 'n', Rotate each character by 'n' positions and print it. Note that 'n' can be positive or negative.

```
# Python program to right rotate a list by n
# Returns the rotated list
def rightRotate(lists, num):
    output_list = []

    # Will add values from n to the new list
    for item in range(len(lists) - num, len(lists)):
        output_list.append(lists[item])

    # Will add the values before
    # n to the end of new list
    for item in range(0, len(lists) - num):
        output_list.append(lists[item])
    return output_list
# Driver Code
rotate_num = 3
list_1 = [1, 2, 3, 4, 5, 6]
print(rightRotate(list_1, rotate_num))
```

OUTPUT:

EXPERIMENT NO: 13

AIM: Given rows of text, write it in the form of columns.

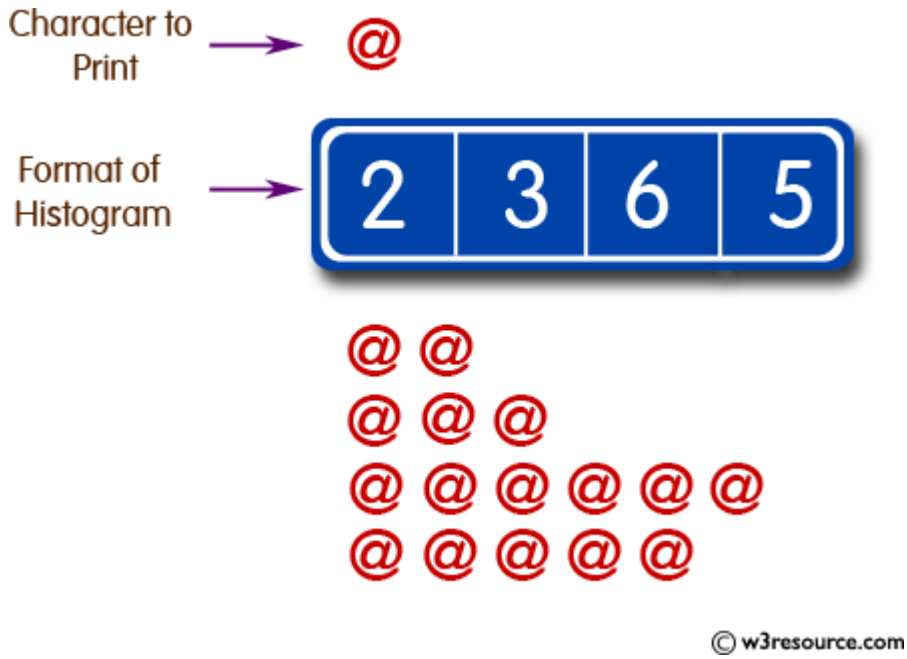
```
# Import pandas package
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Height': [5.1, 6.2, 5.1, 5.2],
        'Qualification': ['Msc', 'MA', 'Msc', 'Msc']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Declare a list that is to be converted into a column
address = ['Delhi', 'Bangalore', 'Chennai', 'Patna']

# Using 'Address' as the column name
# and equating it to the list
df['Address'] = address

# Observe the result
print(df)
```

OUTPUT:

EXPERIMENT NO: 14

AIM: Given a page of text. Count the number of occurrences of each latter (Assume case insensitivity and don't consider special characters). Draw a histogram to represent the same

```
import collections
import pprint
file_input = input('File Name: ')
with open(file_input, 'r') as info:
  count = collections.Counter(info.read().upper())
  value = pprint.pformat(count)
print(value)
```



OUTPUT:

EXPERIMENT NO: 15

AIM: Write program which performs the following operations on list's. Don't use built-in functions

a) Updating elements of a list
b) Concatenation of list's
c) Check for member in the list
d) Insert into the list
e) Sum the elements of the list
f) Push and pop element of list
g) Sorting of list
h) Finding biggest and smallest elements in the list
i) Finding common elements in the list

a)

```
#!/usr/bin/python
list = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 : "
print list[2]
list[2] = 2001;
print "New value available at index 2 : "
print list[2]
```

b)

```
# Python3 code to demonstrate list
# concatenation using naive method

# Initializing lists
test_list1 = [1, 4, 5, 6, 5]
test_list2 = [3, 5, 7, 2, 5]
# using naive method to concat
for i in test_list2 :
    test_list1.append(i)

# Printing concatenated list
print ("Concatenated list using naive method : "
                               + str(test_list1))
```

c)

```
Python code to demonstrate
checking of element existence
using List count() method
"""

# Initializing list
test_list = [10, 15, 20, 7, 46, 2808]
print("Checking if 15 exists in list")

# number of times element exists in list
exist_count = test_list.count(15)

# checking if it is more then 0
if exist_count > 0:
```

```
    print("Yes, 15 exists in list")
else:
    print("No, 15 does not exists in list")
```

d)

```
# create a list of prime numbers
prime_numbers = [2, 3, 5, 7]
# insert 11 at index 4
prime_numbers.insert(4, 11)
print('List:', prime_numbers)
```

e)
```
# Python program to find sum of elements in list
total = 0
# creating a list
list1 = [11, 5, 17, 18, 23]
# Iterate each element in list
# and add them in variable total
for ele in range(0, len(list1)):
    total = total + list1[ele]
# printing total value
print("Sum of all elements in given list: ", total)
```

f)

```
list=[]
list.append(1) # append 1
print("push:",list)
list.append(2) # append 2
print("push:",list)
list.append(3) # append 3
print("push:",list)
list.pop() # pop 3
print("pop:",list)
print("peek:",list[-1]) # get top most element
list.pop() # pop 2
print("pop:",list)
print("peek:",list[-1]) # get top most element
```

g)
```
# A function that returns the length of the value:
def myFunc(e):
  return len(e)
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
cars.sort(reverse=True, key=myFunc)
```

h)
```
lst = []
num = int(input('How many numbers: '))for n in range(num):
    numbers = int(input('Enter number '))
    lst.append(numbers)print("Maximum element in the list is :",
max(lst), "\nMinimum element in the list is :", min(lst))
```

i) list1 = [1, 2]
list2 = [1, 3]

```
list1_as_set = set(list1)
intersection = list1_as_set.intersection(list2)
Find common elements of set and list
intersection_as_list = list(intersection)
print(intersection_as_list)
```

OUTPUTS:

A)

B)

C)

D)

E)

F)

EXPERIMENT NO: 16

AIM: Write a program that reads a file, breaks each line into words, strips whitespace and punctuation from the words, and converts them to lowercase.

```
from string
import
punctuation,
whitespace
               book = 'origin.txt'
               with open(book, 'r') as fd:
                   words = fd.read().split()
               #remove punctuation, whitespace, uppercase
               def clean(word):
                   cleansed = ''
                   for char in word:
                       if ((char in punctuation) or (char in whitespace)):
                           pass
                       else:
                           cleansed += char.lower()
                   return cleansed

               print "{} has {} 'words'".format(book, len([clean(word) for word in words]))
```

OUTPUT:

EXPERIMENT NO: 17(A)

AIM: Write a program illustrating the object oriented features supported by Python

```
class car:
    def_init_(self,modelname, year):
        self.modelname = modelname
        self.year = year
    def display(self):
        print(self.modelname,self.year)
c1 = car("Toyota", 2016)
c1.display()
```

EXPERIMENT NO: 17(B)

AIM: Write a program illustrating the object oriented features supported by Python

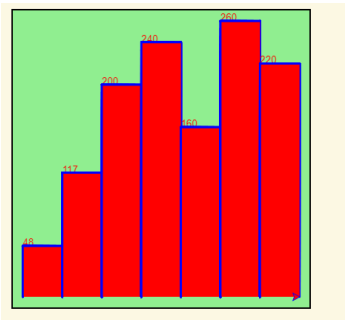
```
class Employee:
    id = 10
    name = "John"

    def display(self):
        print("ID: %d \nName: %s" % (self.id, self.name))
    # Creating a emp instance of Employee class
  emp = Employee()
  # Deleting the property of object
del emp.id
# Deleting the object itself
del emp
emp.display()
```

OUTPUT:

EXPERIMENT NO: 18

AIM: Design a Python script using the Turtle graphics library to construct a turtle bar chart representing the grades obtained by N students read from a file categorising them into distinction, first class, second class, third class and failed.

```
import turtle

def drawBar(t, height):
   """ Get turtle t to draw one bar, of height. """
   t.begin_fill()             # start filling this shape
   t.left(90)
   t.forward(height)
   t.write(str(height))
   t.right(90)
   t.forward(40)
   t.right(90)
   t.forward(height)
   t.left(90)
   t.end_fill()               # stop filling this shape

xs = [48, 117, 200, 240, 160, 260, 220]  # here is the data
maxheight = max(xs)
numbars = len(xs)
border = 10

wn = turtle.Screen()           # Set up the window and its attributes
wn.setworldcoordinates(0-border, 0-border, 40*numbars+border, maxheight+border)
wn.bgcolor("lightgreen")

tess = turtle.Turtle()         # create tess and set some attributes
tess.color("blue")
tess.fillcolor("red")
tess.pensize(3)

for a in xs:
   drawBar(tess, a)
wn.exitonclick()
```



OUTPUT:

EXPERIMENT NO: 19
AIM: Design a Python script to determine the difference in date for given two dates in YYYY:MM:DD format(0 <= YYYY <= 9999, 1 <= MM <= 12, 1 <= DD <= 31) following the leap year rules.

Ex-1:

```
from datetime import date
a = date(2000,2,28)
b = date(2001,2,28)
print(b-a)
```

Ex-2:

```
import datetime
from datetime import date
def differ_days(date1, date2):

    a = date1
    b = date2
    return (a-b).days
print()
print(differ_days((date(2016,10,12)), date(2015,12,10)))
print(differ_days((date(2016,3,23)), date(2017,12,10)))
print()
```

OUTPUTS:

EXPERIMENT NO: 20

AIM: Design a Python Script to determine the time difference between two given times in HH:MM:SS format.(0 <= HH <= 23, 0 <= MM <= 59, 0 <= SS <= 59)

```
# Python 3 program to find difference between two
# given times.
# Remove ':' and convert it into an integer
def removeColon(s):

    if (len(s) == 4):
        s = s[:1] + s[2:]

    if (len(s) == 5):
        s = s[:2] + s[3:]

    return int(s)

# Main function which finds difference
def diff(s1, s2):

    # Change string
    # (eg. 2:21 --> 221, 00:23 --> 23)
    time1 = removeColon(s1)
    time2 = removeColon(s2)

    # Difference between hours
    hourDiff = time2 // 100 - time1 // 100 - 1;

    # Difference between minutes
    minDiff = time2 % 100 + (60 - time1 % 100)

    if (minDiff >= 60):
        hourDiff += 1
        minDiff = minDiff - 60

    # Convert answer again in string with ':'
    res = str(hourDiff) + ':' + str(minDiff)

    return res

# Driver code
s1 = "14:00"
s2 = "16:45"

print(diff(s1, s2))
```

OUTPUT: