

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PYTHON PROGRAMMING LAB**



**MALLA REDDY ENGINEERING COLLEGE**

(AUTONOMOUS)

(An Autonomous institution, Autonomy granted by UGC and affiliated to JNTUH, Accredited by NAAC with 'A' Grade, Accredited by NBA (2008-11) & Recipient of World Bank Assistance under TEQIP phase – II S.C.1.1 for the period (2011-14))

Maisammaguda, Dhulapally (Post. Via. Kompally), Secunderabad  
500100.

Website: [www.mrec.ac.in](http://www.mrec.ac.in)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING****PYTHON PROGRAMMING LAB****OBJECTIVES:**

After completion of this course, the students would be able to

1. Understand how to design and program python application
2. Learn how to use list, tuples and dictionaries
3. Understand how to write functions and pass arguments

**LIST OF EXPERIMENTS**

- 1 a) Write a program to purposefully raise Indentation Error and correct it.  
b) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem).  
c) Write a program add.py that takes 2 numbers as command line arguments and prints its sum.
- 2 a) Write a Program for checking whether the given number is a even number or not.  
b) Using for loop, write a program that prints out the decimal equivalents of  $1/2$ ,  $1/3$ ,  $1/4$ , . . . ,  $1/10$ .  
c) Write a program using for loop that loops over a sequence. What is sequence?  
d) Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.
- 3 a) Find the sum of all the primes below two million.  
b) Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89  
By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.  
c) Write a program to count the numbers of characters in the given string and store them in a dictionary data structure

- d) Write a program to use split and join methods in the given string and trace a birthday with a dictionary data structure.
- 4 a) Write a program to combine two lists into a dictionary.
- b) 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?
- 5 a) Write a program to print each line of a file in reverse order.
- b) Write a program to compute the number of characters, words and lines in a file.
- 6 a) Write a function `ball_collide` that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding.  
Hint: Represent a ball on a plane as a tuple of  $(x, y, r)$ ,  $r$  being the radius. If  $(\text{distance between two balls centers}) \leq (\text{sum of their radii})$  then (they are colliding)
- b) Find mean, median, mode for the given set of numbers in a list.
- 7 a) 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  $b$ .
- b) Write a function `dups` to find all duplicates in the list.
- c) Write a function `unique` to find all the unique elements of a list.
- 8 a) Write a function `cumulative_product` to compute cumulative product of a list of numbers.
- b) Write a function `reverse` to reverse a list. Without using the reverse function.
- 9 Create a Regular Expression and implement the following
  - a) Recognize the following strings: "bat," "bit," "but," "hat," "hit," or "hut."
  - b) Match any pair of words separated by a single space, i.e., first and last names.
  - c) Match any word and single letter separated by a comma and single space, as in last name, first initial.
- 10 Write a python program to implement multithreading scenarios.
- 11 Write a python program to simulate the banking operations using Class.
- 12 Write a python program to demonstrate the Queue / Stack operations using Class.

## INDEX

	Program Name	Page No
1	a) Write a program to purposefully raise Indentation Error and correct it.	6
	b) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem).	7
	c) Write a program add.py that takes 2 numbers as command line arguments and prints its sum.	8
2	a) Write a Program for checking whether the given number is a even number or not.	8
	b) Using for loop, write a program that prints out the decimal equivalent of 1/2, 1/3, 1/4, ..1/10.	9
	c) Write a program using for loop that loops over a sequence. What is sequence?	9
	d) Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero.	11
3	a) Find the sum of all the primes below two million.	12
	b) Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.	12
	c) Write a program to count the numbers of characters in the given string and store them in a dictionary data structure	13
	d) Write a program to use split and join methods in the given string and trace a birthday with a dictionary data structure.	14
4	a) Write a program to combine two lists into a dictionary.	15
	b) 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?	16
5	a) Write a program to print each line of a file in reverse order.	17
	b) Write a program to compute the number of characters, words	18

	and lines in a file.	
6	a) Write a function <code>ball_collide</code> that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding. Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius. If (distance between two balls centers) $\leq$ (sum of their radii) then (they are colliding)	19
	b) Find mean, median, mode for the given set of numbers in a list	20
7	a) Write a function <code>nearly_equal</code> 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.	22
	b) Write a function <code>dups</code> to find all duplicates in the list.	23
	c) Write a function <code>unique</code> to find all the unique elements of a list.	24
8	a) Write a function <code>cumulative_product</code> to compute cumulative product of a list of numbers.	25
	b) Write a function <code>reverse</code> to reverse a list. Without using the reverse function.	26
9	Create a Regular Expression and implement the following	
	a) Recognize the following strings: "bat," "bit," "but," "hat," "hit," or "hut."	26
	b) Match any pair of words separated by a single space, i.e., first and last names.	27
	c) Match any word and single letter separated by a comma and single space, as in last name, first initial.	28
10	Write a python program to implement multithreading scenarios.	30
11	Write a python program to simulate the banking operations using Class.	31
12	Write a python program to demonstrate the Queue / Stack operations using Class.	33

1.

- a) Write a program to purposefully raise Indentation Error and correct it.

**Description:** Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code. The number of spaces is up to you as a programmer, but it has to be at least one. You have to use the same number of spaces in the same block of code, otherwise Python will give you an error

Example1-

**Program:**

```
if5>2:
```

```
print("Five is greater than two!")
```

**Output:**

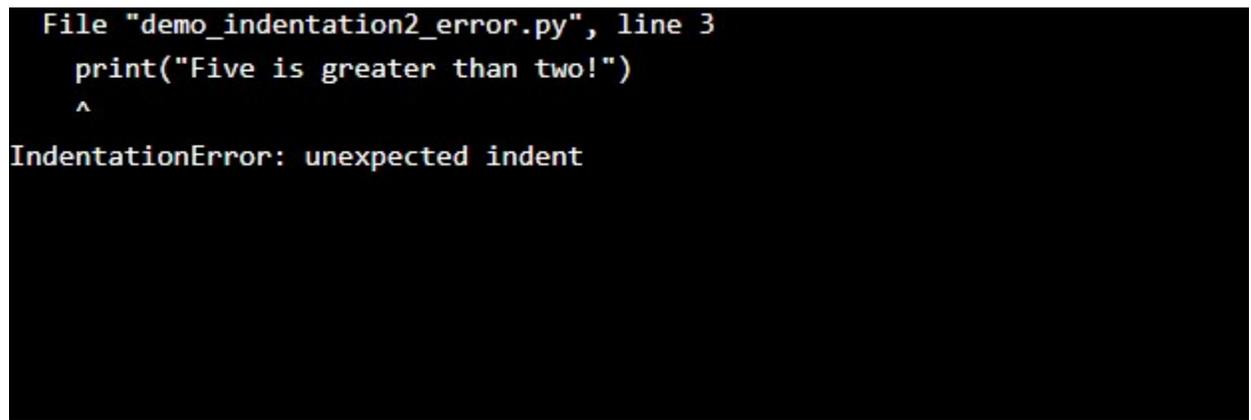
```
File "demo_indentation_test.py", line 2
  print("Five is greater than two!")
    ^
IndentationError: expected an indented block
```

Example 2-

**Program:**

```
if 5 > 2:  
print("Five is greater than two!")  
print("Five is greater than two!")
```

**Output:**



```
File "demo_indentation2_error.py", line 3  
    print("Five is greater than two!")  
    ^  
IndentationError: unexpected indent
```

1 b) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem).

**Program:**

```
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(result)
```

**Output:**

```
enter x1 : 3
enter x2 : 4
enter y1 : 5
enter y2 : 6
1.4142135623730951
```

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

**#addition of two numbers using commandline. Let the filename be add.py**

**Program:**

```
import sys

x=int(sys.argv[1])
y=int(sys.argv[2])

sum=x+y

print("The result is :",sum)
```

**Output:**

**python add.py 5 7**  
**The result is: 12**

2.a) Write a Program for checking whether the given number is a even number or not.

**Program:**

```
n=int(input("Number"))
if n%2==0:
print("Number is even")
else:
print("Number is odd")
```

**Output:**

```
Number 6
Number is even
```

---

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

**Program:**

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

**Output:**

```
0.5
0.3333333333333333
0.25
0.2
0.16666666666666666
0.14285714285714285
0.125
0.11111111111111111
0.1
```

2.c) Write a program using for loop that loops over a sequence. What is sequence?

**Description:** Sequences are one of the principal built-in data types besides numerics, mappings, files, instances and exceptions. Python provides for six sequence (or sequential) data types:

Strings

byteSequences

byte arrays

lists

tuples

range objects

**Program:**

```
thislist = ["apple", "banana", "cherry"]
```

```
for x in thislist:
```

```
    print(x)
```

**Output:**



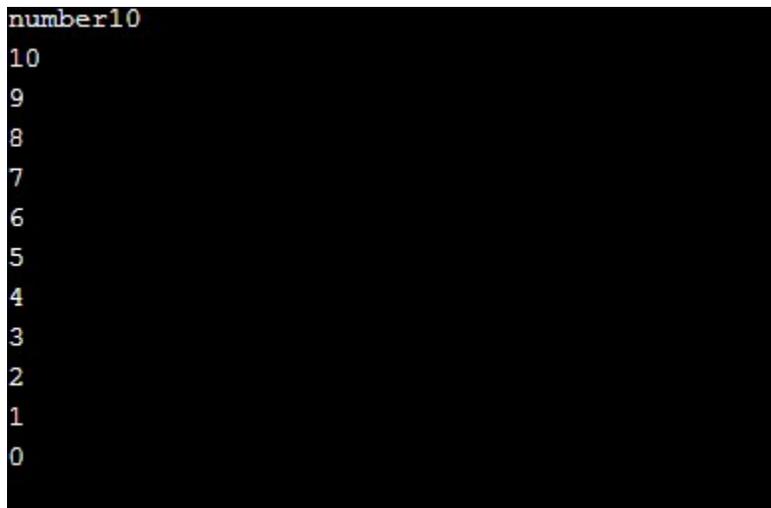
```
apple  
banana  
cherry
```

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

**Program:**

```
n=int(input("number"))
while n>=0:
    print(n)
    n=n-1
```

**Output:**



```
number10
10
9
8
7
6
5
4
3
2
1
0
```

**3 a)** Find the sum of all the primes below two million.

**Program:**

```
s=0
for i in range(2,2000000):
    f=0
    for j in range(2,int(i**0.5)+1):
        if i%j==0:
            f=f+1
    break
    if f==0:
        s=s+i
print("sum is",s)
```

**Output:**

Sum is 142,913,828,922

**3 .b)** Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

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

**Program:**

```
n=int(input("enter the limit:"))
a=1
b=2
```

```
s=0
```

```
while(i<n-2):
```

```
    c=a+b
```

```
if(c%2==0):
```

```
    s=s+c
```

```
    a=b
```

```
    b=c
```

```
print("sum of even numbers",s)
```

**Output:**

```
enter the limit: 5
```

```
sum of even numbers:10
```

**3 .c)**Write a program to count the numbers of characters in the given string and store them in a dictionary data structure

**Program:**

```
x=input("enter string")
```

```
r={}
```

```
for i in x:
```

```
    if i in r:
```

```
        r[i]+=1
```

```
    else:
```

```
r[i]=1  
print(r)
```

**Output:**

```
enter stringhello  
{'e': 1, 'o': 1, 'h': 1, 'l': 2}
```

**3.d)** Write a program to use split and join methods in the given string and trace a birthday with a dictionary data structure.

**Program:**

```
str1=input("Enter date of birth:")  
sp=str1.split(".")  
bday='/'.join(sp)  
d1={"birthday":bday  
    }  
if "birthday" in d1:  
print(d1["birthday"])
```

**Output:**

```
Enter date of birth:4.5.1987  
4/5/1987
```

4.a) Write a program to combine two lists into a dictionary.

**Program:**

```
t1=["name","clg","age"]
```

```
t2=["abc","mrec",34,]
```

```
res = {}
```

```
for key in t1:
```

```
for value in t2:
```

```
res[key] = value
```

```
t2.remove(value)
```

```
break
```

```
print(res)
```

**Output:**

```
['clg': 'mrec', 'age': 34, 'name': 'abc']
```

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

**Program:**

```
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:**

```
File Name:  abc.txt
Counter({' ': 93,
        'E': 64,
        'N': 45,
        'A': 42,
        'T': 40,
        'I': 36,
        'O': 31,
        'R': 29,
        'H': 25,
        'D': 19,
        'M': 17,
        'Y': 17,
        'L': 15,
        'F': 15,
        'U': 14,
        'C': 13,
```

```
'G': 13,  
'S': 12,  
' ': 7,  
'B': 6,  
'W': 5,  
'9': 5,  
'.': 4,  
'P': 4,  
'1': 3,  
'\n': 2,  
'0': 2,  
'3': 2,  
' ': 1,  
'-': 1,  
'K': 1,  
'(': 1,  
')': 1,  
'V': 1})
```

**5.a)** Write a program to print each line of a file in reverse order.

**Description:**

reversed() function produces a reverse iterator.

**Program:**

```
textfile=open("a_file.txt")
```

```
lines=textfile.readlines()
```

```
for line in reversed(lines):
```

```
print(line)
```

**Output:**

ghi

def

abc

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

**Program:**

```
file=open("sample.txt","r")

number_of_lines=0
number_of_words=0
number_of_characters=0
for line in file:
    line=line.strip("\n")#won'tcount \n as character
    words=line.split()
    number_of_lines+=1
    number_of_words+=len(words)
    number_of_characters+=len(line)

file.close()
print("lines:", number_of_lines, "words:", number_of_words, "characters:",
number_of_characters)
```

**OUTPUT:**

lines: 3 words: 5 characters: 29

**6.a)** Write a function `ball_collide` that takes two balls as parameters and computes if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding.

Hint: Represent a ball on a plane as a tuple of  $(x, y, r)$ ,  $r$  being the radius. If (distance between two balls centers)  $\leq$  (sum of their radii) then (they are colliding)

**Program:**

```
def ball_collide(b1,b2):
    if int(b1[2])<=int(b2[2]):
        return True
    else:
        return False

a=int(input("enter a number"))
b=int(input("enter a number"))
c=int(input("enter a number"))
d=int(input("enter a number"))
e=int(input("enter a number"))
f=int(input("enter a number"))
ball1=(a,b,c)
ball2=(d,e,f)
status=ball_collide(ball1,ball2)
if status==True:
    print("balls are collided")
else:
    print("balls are not collided");
```

**OUTPUT:**

```
enter a number3
enter a number4
enter a number5
enter a number6
enter a number7
enter a number5
balls are collided
```

**6.b)** Find mean, median, mode for the given set of numbers in a list.

**Program:**

```
n_num = [1, 2, 3, 4, 5]
```

```
n = len(n_num)
```

```
get_sum = sum(n_num)
```

```
mean = get_sum / n
```

```
print("Mean / Average is: " + str(mean))
```

```
n_num.sort()
```

```
if n % 2 == 0:
```

```
    median1 = n_num[n//2]
```

```
    median2 = n_num[n//2 - 1]
```

```
    median = (median1 + median2)/2
```

```
else:
    median = n_num[n//2]
print("Median is: " + str(median))
data = Counter(n_num)
get_mode = dict(data)
mode = [k for k, v in get_mode.items() if v ==
max(list(data.values()))]

if len(mode) == n:
    get_mode = "No mode found"
else:
    get_mode = "Mode is / are: " + ', '.join(map(str,
mode))

print(get_mode)
```

**OUTPUT:**

Mean / Average is: 3.0

Median is: 3

Mode is / are: 5

7(a) 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:**

**Description:** Two strings are said to be nearly equal if a single mutation applied to one string will result in another string. That means, Given two strings s1 and s2, find if s1 can be converted to s2 with exactly one edit (mutation). If yes, then the function should return a True value as the result. Otherwise, it must return a False value as the result.

```
defnearly_equal(str1,str2):
count=0
i=j=0
while(i<len(str1) and j<len(str2)):
if(str1[i]!=str2[j]):
count=count+1
if(len(str1)>len(str2)):
i=i+1
elif(len(str1)==len(str2)):
pass
else:
i=i-1
if(count>1):
return False
i=i+1
j=j+1
if(count<2):
return True

str1=input("Enter first string:\n")
str2=input("Enter second string:\n")
boolean=nearly_equal(str1,str2)
if(boolean):
print("Strings are nearly equal.")
else:
print("Strings are not equal.")
```

**OUTPUT:**

```
Enter first string::  
RISE  
Enter second string::  
RISEE  
Strings are nearly equal.
```

7 b) Write a function dups to find all duplicates in the list.

**Description:** We need to write a function dups to find all duplicate elements in the list. If an element is repeated more than once in the list, then add that repeated element to the resultant list.

**Program:**

```
def dups(numlist):  
    duplicates={}  
    forele in numlist:  
        c=numlist.count(ele)  
        if(c>=2):  
            duplicates[ele]=c  
    print("Duplicate elements are:\n",duplicates)  
    return  
numlist=[]  
n=int(input("Enter number of elements to be insert:\n"))  
fori in range(n):  
    ele=int(input("Enter element"))  
    numlist.append(ele)  
    dups(numlist)
```

**OUTPUT:**

Enter number of elements to be insert:

5

Enter element2

Enter element4

Enter element2

Enter element3

Enter element3

Duplicate elements are:

{2: 2, 3: 2}

7 c) Write a function unique to find all the unique elements of a list.

**Description:** We need to write a function unique to find all unique elements in the list. If an element is found only once in the list, then add that element to the resultant list.

**Program:**

```
def unique(numlist):
    uniqueele=[]
    forele in numlist:
        c=numlist.count(ele)
    if(c==1):
        uniqueele.append(ele)
    print("Unique elements are:\n",uniqueele)
    return

numlist=[]
n=int(input("Enter number of elements to be insert:\n"))
fori in range(n):
    ele=int(input("Enter element"))
    numlist.append(ele)
    unique(numlist)
```

**OUTPUT:**

Enter number of elements to be insert:

5

Enter element2

Enter element3

Enter element4

Enter element3

Enter element2

Unique elements are:

[4]

**8 a)** Write a function `cumulative_product` to compute cumulative product of a list of numbers.

**Description:** We need to write a function `cumulative_product` to find cumulative product of numbers in the list. A Cumulative product is a sequence of partial products of a given sequence. For example, The cumulative product of sequence `[a,b,c,.....]` are `a,ab,abc,.....`

**Program:**

```
fromfunctools import reduce
nums = [10, 20, 30,]
nums_product = reduce( (lambda x, y: x * y), nums)
print("Product of the numbers : ",nums_product)
```

**OUTPUT:**

```
$python main.py
```

```
(Product of the numbers : ', 6000)
```

8 b) Write a function reverse to reverse a list. Without using the reverse function.

**Description:** We need to write a function reverse to reverse the given elements in a list. Reversing of a list is done by using the feature called slicing. Python's list objects have an interesting feature called slicing

**Program:**

```
defmyfunc(a):  
    rev_str = []  
    for x in a:  
        rev_str.insert(0, x)  
    returnrev_str  
print(myfunc([1,2,3,4,5]))
```

**OUTPUT:**

```
$python main.py
```

```
[5, 4, 3, 2, 1]
```

9. Create a Regular Expression and implement the following

a) Recognize the following strings: "bat," "bit," "but," "hat," "hit," or "hut."

**Program:**

```
import re  
String1="hi hitman had you gone for batting"  
x = re.findall("&quot;bat&quot;", String1)  
print(x)
```

```
String2="he is bitten by a dog"  
x= re.findall("&quot;bit&quot;", String2)  
print(x)
```

```
String3="butterfly is colorfull"
```

```
x= re.findall("&quot;but&quot;, String3)
```

```
print(x)
```

```
String4="hat is oval in shape"
```

```
x= re.findall("&quot;hat&quot;, String4)
```

```
print(x)
```

```
String5="ramana is living in the hut,hits mosquitoes in the evening"
```

```
x= re.findall("&quot;hut|hit&quot;, String5)
```

```
print(x)
```

**OUTPUT:**

```
["batting"]
```

```
["bitten"]
```

```
["butterfly"]
```

```
["hat"]
```

```
["hut"]
```

**9.b)**Match any pair of words separated by a single space, i.e., first and last names.

**Program:**

```
import re
```

```
s1="vamshi krishna is learning java,vamshi krishna is eagerly waiting for learning python"
```

```
s2=re.findall("vamshi krishna",s1)
```

```
print(s2)
```

**OUTPUT:**

```
['vamshikrishna', 'vamshikrishna']
```

**9.c)** Match any word and single letter separated by a comma and single space, as in last name, first initial.

**Program:**

```
def printInitials(string: str):
    length = len(string)
    # to remove any leading or trailing spaces
    string.strip()
    # to store extracted words
    t = ""
    for i in range(length):
        ch = string[i]
        if ch != ' ':
            # forming the word
            t += ch
        # when space is encountered
        # it means the name is completed
        # and thereby extracted
    else:
        # printing the first letter
        # of the name in capital letters
        print(t[0].upper() + ". ", end="")
        t = ""
```

```
temp = ""
# for the surname, we have to print the entire
# surname and not just the initial
# string "t" has the surname now
for j in range(len(t)):
    # first letter of surname in capital letter
    if j == 0:
        temp += t[0].upper()
    # rest of the letters in small
    else:
        temp += t[j].lower()
# printing surname
print(temp)
# Driver Code
if __name__ == "__main__":
    string = "ishitabhuiya"
    printInitials(string)
```

**OUTPUT:**

I. Bhuiya

10. Write a python program to implement multithreading scenario.

**Description:** The below program shows the simultaneous execution of multiple number of threads.

**Program:**

```
import thread
import time

# Define a function for the thread
def print_time( threadName, delay):
    count = 0
    while count < 5:
        time.sleep(delay)
        count += 1
    print ("%s: %s" % ( threadName, time.ctime(time.time()) )

# Create two threads as follows
try:
    thread.start_new_thread(print_time, ("Thread-1", 2, ) )
    thread.start_new_thread(print_time, ("Thread-2", 4, ) )
except:
    print "Error: unable to start thread"

while 1:
    pass
```

**OUTPUT:**

Thread-1: Thu Jan 22 15:42:17 2009

```
Thread-1: Thu Jan 22 15:42:19 2009
Thread-2: Thu Jan 22 15:42:19 2009
Thread-1: Thu Jan 22 15:42:21 2009
Thread-2: Thu Jan 22 15:42:23 2009
Thread-1: Thu Jan 22 15:42:23 2009
Thread-1: Thu Jan 22 15:42:25 2009
Thread-2: Thu Jan 22 15:42:27 2009
Thread-2: Thu Jan 22 15:42:31 2009
Thread-2: Thu Jan 22 15:42:35 2009
```

**11.** Write a python program to simulate the banking operations using class.

**Description:** The following program provides simulation of Banking operations.

**Program:**

```
class Account:
    def __init__(self):
        self.balance=0
        print('Your Account is Created.')
    def deposit(self):
        amount=int(input('Enter the amount to deposit:'))
        self.balance+=amount
        print('Your New Balance =%d' %self.balance)
    def withdraw(self):
        amount=int(input('Enter the amount to withdraw:'))
        if(amount>self.balance):
            print('Insufficient Balance!')
        else:
```

```
self.balance-=amount  
print('Your Remaining Balance =%d' %self.balance)
```

```
def enquiry(self):  
    print('Your Balance =%d' %self.balance)  
  
account= Account()  
account.deposit()  
account.withdraw()  
account.enquiry()
```

**OUTPUT:**

Your Account is Created.

Enter the amount to deposit: 5000

Your New Balance = 5000

Enter the amount to withdraw: 2000

Your Remaining Balance = 3000

Your Balance = 3000

12. Write a python program demonstrate the Queue/Stack operations using class.

**Description:**The following program demonstrates the operations of stack.

```
class Stack:
def __init__(self):
self.items=[]
def is_empty(self):
return self.items==[]
def push(self, data):
self.items.append(data)
def pop(self):
return self.items.pop()
def display(self):
return self.items

s= Stack()
while True:
print("push value ")
print("pop")
print("quit")
print("display")
do= input("What would you like to do?").split()
operation= do[0].strip().lower()
if operation== "push":
```

```
s.push(int(do[1]))
elif operation=="pop":
    if s.is_empty():
        print("Stack is empty")
    else:
        print("Popped value:", s.pop())
elif operation=="display":
    print(s.display())
elif operation=="quit":
    break
```

**OUTPUT:**

```
push value
pop
quit
display
What would you like to do?push 10
push value
pop
quit
display
What would you like to do?display
[10]
push value
pop
quit
display
What would you like to do?pop
Popped value: 10
push value
pop
quit
display
What would you like to do?█
```

