21.Using the built-in functions on Numbers perform following operations

a) Round of the given floating point number. Example: n=0.543 then round it next decimal number, i.e n=1.0 Use round() function

b) Find out the square root of a given number. (Use sqrt(x) function)

c) Generate random number between 0, and 1 (use random() function)

d) Generate random number between 10 and 500. (Use uniform() function)

e) Explore all Math and Random module functions on a given number/ List.

```
import math

import random

def operationOnNumbers():

    try:

        n=0.543

        print(round(n))

        val1 = input("Enter the number to find squareroot")

        sqrtval = math.sqrt(val1)

        print sqrtval

        print random.random()

        print random.uniform(1,100)

        print math.ceil(12.534)

        print math.floor(12.534)

        print math.trunc(12.984)

        print pow(2,3)

        print random.randint(1,10)

        print random.randrange(20,50)

    except Exception as e:

        print e
```

```
if __name__ == '__main__':
    operationOnNumbers()
```

22. Read the value x and y from the user and apply all trigonometric functions on these numbers. Note: Refer the tutorial Trigonometric operation table.

```
import math
def trignometricOperations(a,b):
    print "cos(",a,")", math.cos(a)
    print "sin(",a,")", math.sin(a)
    print "tan(",a,")", math.tan(a)
    print "acos(",a,")", math.acos(a)
    print "asin(",a,")", math.asin(a)
    print "atan(",a,")", math.atan(a)
    print "atan2(",a,b,")", math.atan2(a,b)
    print "hypot(",a,b,")", math.hypot(a,b)


if __name__ == "__main__":
    try:
        a=int(input("Enter value1:"))
        b=int(input("Enter value2:"))
        trignometricOperations(a,b)
    except Exception as e:
        print e
```

23. Find the area of Circle given that radius of a circle. (Use pi value from Math module as mathematical constant)

```
import math

def calculateArea(r):

    return math.pi*r*r



if __name__ == "__main__":

    try:

        radius=float(input("Enter the radius:"))

        area = calculateArea(radius)

        print area

    except ValueError:

        print "Enter only numeric values"

    except Exception as e:

        print e
```

24. Special Characters: Write program to explore all Escape characters specified in Tutorial (Under String chapter)

```
def escapeCharacters():
    #single quote
    print '\'Hello\''
    #double quote
    print '\"Hello\"'
    #new line
    print 'Hello\nWorld'
    #backslash
    print 'Hello\\World'
    #backspace
    print 'Hello \bWorld'
    #carriage return
    print 'Hello\rWorld'
    #horizontal tab
    print 'Hello\tWorld'
    #vertical tab
    print 'Hello\vWorld'
    #octal
    print '\110\145\154\154\157'
    #hex
    print '\x48\x65\x6c\x6c\x6f'

if __name__ == "__main__":
    try:
        escapeCharacters()
    except Exception as e:
        print e
```

25. Write a program to print the different data types (Numbers, strings characters) using the Format symbols (Refer to different format symbols specified in Tutorial)

```
def printDifferentDataTypes(a,b,c,d,e):
    #int
    print "%d"%a
    #float
    print "%f"%b
    #float with 2 precision
    print "%.2f"%b
    #string
    print "%s"%c
    #hex representation
    print "%x" %a
    #char
    print "%c"%e
    #exponential notation
    print "%e"%a
    #octal
    print "%o"%a
    #unsigned int
    print "%u"%a
```

```
if __name__ == '__main__':
    try:
        a=5289
        b=5.678954
        c="ABCD"
        d = True
        e='a'
        printDifferentDataTypes(a,b,c,d,e)
    except Exception as e:
        print e
```

26. Receive the encoded string from your friend & decode it to check the original message. PS: You will receive Encoded string and the Algorithm used for encoding.

```
def decode(str1,encoding,errors):

    return str1.decode(encoding,errors)



if __name__ == '__main__':

    try:

        str1 = input("Enter the encode string")

        print "Enter the algorithm used for encoding"

        encoding = input() #base64

        errors = input() #strict

        decodedstr = decode(str1,encoding='base64',errors='strict')

        print "Encodedstring: ", str1

        print "Decodedstring: ",decodedstr

    except Exception as e:

        print e
```

27. Write a program to check given string is Palindrome or not.That is reverse the given string and check whether it is same as original string, if so then it is palindrome.
Example: String = "malayalam" reverse string = "malayalam" hence given string is palindrome. Use built functions to check given string is palindrome or not.

```
def isPalindrome(str1):
      if str1 == str1[::-1]:
         return True
      return False

if __name__ == '__main__':
   try:
      str1=input("Enter the string:")
      print(isPalindrome(str1))
   except TypeError:
      print "Enter only string input"
   except Exception as e:
      print e
```

28. Write a program to check how many ovals present in the given string. That is, count the number of " a e i o u" in the given string and print the numbers against each oval. Example:- "This is Python" Number of total ovals = 3, i = 2, o =1

```
def VowelsFilter(str1):
  vowels = "aeiou"
  count=0
  dict1 = {}
  for c in str1:
     if c.isupper():
        c = c.lower()
     if c in vowels:
        if(c in dict1):
           dict1[c] += 1
        else:
           dict1[c]=1
        count +=1
  print 'ovals = ', count ,
  for i in dict1:
     print ",", i," = ",dict1[i],
if __name__ == "__main__":
        try:
        str1=input("Enter the string:")
        VowelsFilter(str1)
        except Exception as e:
            print e
```

29. Apply all built in functions on Strings in your program. Note: There are 40 string functions. Use Tutorial for the help. Note: Each program should have 5 string built in functions(so write 8 programs to cover all string functions)

29.1

```
def stringFunctions():
    str1 = "welcome to the python programming course welcome"
    print "str.capitalize():",str1.capitalize()
    print "str.center(65,'*'):",str1.center(65,'*')
    print "str.count('welcome'):",str1.count('welcome')
    print "str.count('o',9,22):",str1.count('o',9,22)

    str1 = str1.encode('base64','strict')
    print "Encoded string is:",str1

    str1 = str1.decode('base64','strict')
    print "Decoded string is:",str1

if __name__ == "__main__":
        try:
           stringFunctions()
        except Exception as e:
            print e
```

29.2

```
def stringFunctions():
    str1 ="Welcome to python\tprogramming course"

    #endswith
    print str1.endswith('course')
    print str1.endswith("Welcome",0,7)
    #expandtabs
    print str1
    print str1.expandtabs() #default size=8
    print str1.expandtabs(16)

    #find
    print str1.find("co")
    print str1.find("co",10,len(str1))
    print str1.find("hello")

    #index
    print str1.index("co")
    print str1.index("co",10)
    #print str1.index("hello") #if not found, find method return -1 whereas
index throws value error

    #join
    list1 = ['1','2','3']
    str2 = "-".join(list1)
    print str2


if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.3

```
def stringFunctions():
    str1 ="welcome to python programming course"
    str2 = '2033'
    str3 = "123abcd4"
    str4 = "Python"

 #isalnum
    print str1.isalnum()
    print str2.isalnum()
    print str3.isalnum(),"\n"
    #isalpha
    print str1.isalpha() #only aplphabets no spaces or special char or
numbers
    print str2.isalpha()
    print str4.isalpha(),"\n"

    #isdigit
    print str1.isdigit()
    print str2.isdigit()
    print str3.isdigit(),"\n"

    #islower
    print str1.islower()
    print str2.islower()
    print str3.islower()
    print str4.islower(),"\n"

    #isnumeric --only for unicode objects
    str5 = u"1234"
    str6 = u"a123b"
    str7 = u"abcd"
    print str5.isnumeric()
    print str6.isnumeric()
    print str7.isnumeric(),"\n"

if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.4

```
def stringFunctions():
    str1 = "        "
    str2 = "     A   "
    str3 = "Welcome To Python Programming Course"
    str4 = "Welcome to python programming course"
    str5 = "WELCOME TO PYTHON PROGRAMMING COURSE"

    #isspace
    print str1.isspace()
    print str2.isspace()
    print str3.isspace(),"\n"

    #istitle
    print str3.istitle()
    print str4.istitle()
    print str5.istitle(),"\n"

    #isupper
    print str3.isupper()
    print str4.isupper()
    print str5.isupper(),"\n"

    #ljust
    print str5.ljust(50,"#") #only char allowed not string
    print str5.ljust(20,'#'),"\n" #value less than length

    #rjust
    print str5.rjust(50,"#") #only char allowed not string

if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.5

```
def stringFunctions():
    str1 = "    ***%%**Welcome To Python Programming
Course**%%%***    "
    str2 = "Welcome-To-Python-Programming-Course"
    str3 = "843921"

    #strip
    print str1.strip(),"\n"

    #lstring
    print str1.lstrip(" *%"),"\n"

    #rstrip
    print str1.rstrip(" *%"),"\n"
    #max
    print max(str2)
    print max(str3),"\n"

    #min
    print min(str2)
    print min(str3),"\n"

if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.6

```
def stringFunctions():
    str1 = "Welcome-To-Python-Programming-Course"
    str2 = "WELCOME TO THE CLASS"

    #upper
    print str1.upper(),"\n"

    #lower
    print str2.lower(),"\n"

    #replace
    print str1.replace('o','$')
    print str1.replace('o','$',3) #last argument is max replace

    #find
    print str1.find('o')
    print str1.find('o',8,16)

    #rfind
    print str1.rfind('o')
    print str1.rfind('o',8,16)


if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.7

```
from string import maketrans

def stringFunctions():
    str1 = "Welcome-To-Python-Programming-Course"
    str2 = "WELCOME TO THE CLASS"

    #len
    print len(str1)
    print len(str2),"\n"

    #rindex
    print str1.rindex('co'),"\n"

    #maketrans
    intab = "aeiou"
    outtab = "AEIOU"
    trantab = maketrans(intab, outtab)
    print str1.translate(trantab),"\n"

    #split
    print str2.split()
    print str1.split("-",2),"\n"

    #swapcase
    print str1.swapcase()

if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

29.8

```
from string import maketrans

def stringFunctions():
    str1 = "welcome-to-python-programming-course"
    #startswith
    print str1.startswith("wel")
    print str1.startswith("python",11,60),"\n"

    #title()
    print str1.title(),"\n"

    #translate
    intab = "aeiou"
    outtab = "12345"
    trantab = maketrans(intab, outtab)
    print str1.translate(trantab),"\n"

    #zfill
    print str1.zfill(50),"\n"

    #isdecimal -- only for unicode
    str2 = u"1234"
    str3 = u"12ab3"
    print str2.isdecimal()
    print str3.isdecimal(),"\n"

    #splitline
    str4 = "WELCOME\nTO\nTHE\nCLASS"
    print str4.splitlines()


if __name__ == "__main__":
    try:
        stringFunctions()
    except Exception as e:
        print e
```

30. Write a program to Sort given N numbers (Use only loop structures and Conditions to sort the elements. Use Bubble sort / Selection sort technique to sort the elements of List) Note: Don't use built in functions to sort.

```
def selectionsort(list1):

    for i in range(len(list1)):
        min=i
        for j in range(i+1,len(list1)):
            if(list1[min]>list1[j]):
                min=j
        list1[i],list1[min] = list1[min],list1[i]
    return list1

if __name__ == "__main__":
    try:
        list1 = [2,3,12,34,1,234,23,455,6778,233,22,12,11]
        sortedlist = selectionsort(list1)
        print sortedlist
    except Exception as e:
        print e
```

31. Write a program to search an element in the list. i.e. Perform the binary search on the sorted list having integers as elements. If the search is successful print "Success" else print "un successful search".

```
def binarysearch(list1,x):
    start = 0
    end = len(list1)-1
    while(start<=end):
        mid = start + (end-start)/2
        if list1[mid] == x:
            return mid
        elif x > list1[mid]:
            start = mid+1
        else:
            end = mid-1
    return -1

if __name__ == "__main__":
    try:
        sortedlist = [2,3,12,34,45,67,73,76,79,81,85,90,98,103,111]
        x = int(input("Enter the value to be found: "))
        position = binarysearch(sortedlist,x)
        if(position==-1):
            print "Element not found"
        else:
            print "Element found at position", position
    except ValueError:
            print "Enter only int value"
```

```
          except Exception as e:
             print e
```

32. Write a program to perform following operations on List. Create three lists as List1, List2 and List3 having 5 city names each list.
a. Find the length city of each list element and print city and length
b. Find the maximum and minimum string length element of each list
c. Compare each list and determine which city is biggest & smallest with length.
d. Delete first and last element of each list and print list contents.

```
def printCityWithLenAndFindBiggestAndSmallestCity(WholeList):
   biggestcity=WholeList[0][0]
   smallestcity=WholeList[0][0]
   for list in WholeList:
      minlen=len(list[0])
      maxlen=len(list[0])
      listbcity=list[0]
      listscity=list[0]
      for city in list:
         citylen = len(city)
         print city,citylen
         if minlen>citylen:
            minlen = citylen
            listscity = city
            if(len(smallestcity)>minlen):
               smallestcity = city
         elif maxlen<citylen:
            maxlen = citylen
            listbcity = city
            if(len(biggestcity)<maxlen):
               biggestcity = city
      print 'Maximum length element of ', list,'is',listbcity,'with
length',maxlen
      print 'Minimum legth of ',list,'is',minlen,'is',listscity,'with
length',minlen
   print smallestcity,"is the smallest city with length of ",len(smallestcity)
   print biggestcity,"is the biggest city with length of",len(biggestcity)

def deleteFirstAndLastElement(WholeList):
   for list in WholeList:
      del(list[0])
```

```
        del(list[-1])
        print list


if __name__ == "__main__":
    try:
        list1 = ['Banglore', 'Delhi', 'Mumbai', 'Pune', 'Chennai']
        list2 = ['Paris', 'Berlin', 'Moscow', 'Lyon', 'Mersaille']
        list3 = ['Washington', 'Miami', 'New Orleans', 'New york',
'Sacremento']
        Wholelist=[list1,list2,list3]
        printCityWithLenAndFindBiggestAndSmallestCity(Wholelist)
        deleteFirstAndLastElement(Wholelist)
    except Exception as e:
        print e
```

33. Create a list with 7 elements and perform following operations. Let the list be initialized with List1 any 5 city names;
a) Append a new city name to the List
b) Insert a city name at 4th index position
c) Sort the list and print all elements
d) Sort the elements of the list in descending order.
e) delete last three elements using pop operation

```
def appendToList(list1):
    list1.append('Hyderabad')
    print(list1)
def insertAt4thIndex(list1):
    list1.insert(4,"Kolkata")
    print(list1)
def sortingList(list1):
    list1.sort()
    print(list1)
def descendingSortList(list1):
    list1.sort(reverse=True)
    print(list1)
def popElement(list1):
    print "Deleted element is",list1.pop()
    print(list1)


if __name__ == "__main__":
    try:
        list1 = ['Banglore', 'Delhi', 'Mumbai', 'Pune', 'Chennai']
        print(list1)
        appendToList(list1)
        insertAt4thIndex(list1)
        sortingList(list1)
        descendingSortList(list1)
        popElement(list1)
        popElement(list1)
        popElement(list1)
    except Exception as e:
        print e
```

34. Create 3 Lists (list1, list2, list3) with integer and perform following operations
a) Create Maxlist by taking 2 maximum elements from each list.
b) Find average value from all the elements of Maxlist.
c) Create a MinlIst by taking 2 minimum elements from each list
d) Find the average value from all the elements of Minlist

```
from statistics import mean


def createMaxList(WholeList):
    global maxlist
    maxlist = []
    for list in WholeList:
        list.sort()
        maxlist.extend(list[-2:])
    print("maxlist =",maxlist)

def createMinList(WholeList):
    global minlist
    minlist = []
    for list in WholeList:
        list.sort()
        minlist.extend(list[:2])
    print("minlist =",minlist)

def avgOfMaxList(maxlist):
    maxlistAvg = mean(maxlist)
    print("Average of maxlist is", maxlistAvg)
```

```
def avgOfMinList(minlist):
    minlistAvg = mean(minlist)
    print("Average of minlist is", minlistAvg)

if __name__ == "__main__":
    try:
        list1 = [34,23,45,12,25,67,89,33,77]
        list2 = [55,53,65,45,36,67,76,99,10]
        list3 = [66,55,1,23,9,98,76,45,34,33]
        WholeList = [list1,list2,list3]
        createMaxList(WholeList)
        createMinList(WholeList)
        avgOfMaxList(maxlist)
        avgOfMinList(minlist)
    except Exception as e:
        print(e)
```

35. Create Tuple as specified below;

a) Create a Tuple tup1 with days in a week & print the tuple elements
b) Create a Tuple tup2 with months in a year and concatenate it with tup1
c) Create 3 tuples(tup1,tup2,tup3) with numbers and determine which is greater.
d) Try to delete an individual element in tup1 and try deleting complete Tuple -tup1 notice the error type you get.
e) Insert new element in to tuple by typecasting it to List

```
def printTupleElements(tuple1):
    for element in tuple1:
        print element,
    print "\v"

def concatenateTuples(tuple1,tuple2):
    tuple1 += tuple2
    return tuple1

def findingGreaterTuple(tup1,tup2,tup3):
    greatertuple=tup2
    if cmp(tup1,tup2)==1:
        greatertuple=tup1
    if cmp(tup3,greatertuple)==1:
        greatertuple=tup3
    print "Greater tuple is", greatertuple

def delTuple(tup1,tup2):
    try:
        #deleting entire tuple is possible
        del(tup2)
        #trying to delete tuple element will throw error
        del(tup1[0])
    except TypeError:
        print "Tuple doesn't support deletion"

def convertToList(tup1):
    return list(tup1)
    print list1
    list1.append(123)
    print list

def appendValueToList(list1,value):
    list1.append(value)

if __name__ == "__main__":
    try:
        #printing elements in tuple
        tuple1 =
("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Sat
urday")
        printTupleElements(tuple1)

        tuple2 =
("January","February","March","April","May","June","July","August","S
eptember","October","November","December")
        printTupleElements(tuple2)

        #concatenating tuples
        tuple1 = concatenateTuples(tuple1,tuple2)
        printTupleElements(tuple1)

        #finding greater tuple
        tup1 = (34,23,45,12,25,67,89,33,77)
        tup2 = (5,53,65,45,36,67,76,99,10)
        tup3 = (66,55,1,23,9,98,76,45,34,33)
        findingGreaterTuple(tup1,tup2,tup3)
        #trying to delete tuple element and entire tuple
        delTuple(tup1,tup2)

        #typecating to list
        list1 = convertToList(tup1)

        #inserting value into list
        appendValueToList(list1,123)
        print list1

    except Exception as e:
        print(e)
```

36. Create two tuples tup1 & tup2 and apply all built in functions on tuples.(Refer Tutorial for the Built in functions list)

```
def compareTuple(tup1,tup2):
    greatertuple=tup2
    if cmp(tup1,tup2)==1:
        greatertuple=tup1
    return greatertuple

def findTupleLength(tup1):
    return len(tup1)

def maximumElement(tup1):
    return max(tup1)

def minimumElement(tup1):
    return min(tup1)

def convertToTuple(list1):
    return tuple(list1)

def deleteTuple(tup1):
    print "Tuple deleted is ", tup1
    del tup1

def countElement(tup1,element):
    return tup1.count(element)
```

```
def findIndex(tup1,element):
    return tup1.index(element)


if __name__ == "__main__":
    try:
        tup1 = (34,23,45,12,25,67,89,33,77,12)
        tup2 = (5,53,65,45,36,67,76,99,10)
        tup3 = ("Eng","Math","Phy","Bot","Zoo","Comp")

        #cmp
        greatertuple = compareTuple(tup1,tup2)
        print "Greater tuple is", greatertuple

        #len
        length = findTupleLength(tup3)
        print "length of tuple",tup3, "is",length

        #max
        print "Max of int tuple is",maximumElement(tup1)
        print "Max of string tuple is",maximumElement(tup3)

        #min
        print "Min of int tuple is",minimumElement(tup1)
        print "Min of string tuple is",minimumElement(tup3)

        #Typecasting
        list1 = ['abcd','efgh','ijkl']
        tup4 =convertToTuple(list1)
        print tup4

        #delete tuple
        deleteTuple(tup2)


        #find count of specific element
        count = countElement(tup1,12)
        print "Count of element 12 in tuple",tup1,"is",count

        #finding index
        index = findIndex(tup3,"Bot")
        print "Index of element 'Bot' in tuple",tup3,"is",index
    except Exception as e:
        print(e)
```

37. Create 3 dictionaries (dict1, dict2, dict3) with 3 elements each. Perform following operations
a) Compare dictionaries to determine the biggest.
b) Add new elements in to the dictionaries dict1, dict2
c) print the length of dict1, dict2, dict3.
d) Convert dict1, dict2, and dict3 dictionaries as string and concatenate all strings together.

```
import json

def biggestDict(dict1,dict2,dict3):
    biggestDict = dict2
    if cmp(dict1,dict2):
        biggestDict = dict1
    if cmp(dict3,biggestDict):
        biggestDict = dict3
    return biggestDict

def addToDict(tempdict,key,val):
    tempdict[key] = val #dict2['l'] = 12

def addDictToAnotherDict(tempdict1,tempdict2):
    tempdict1.update(tempdict2) #dict1.update({'j':10,'k':11})

def findLength(tempdict):
    return len(tempdict)

def convertToStringAndConcatenate(dict1,dict2,dict3):
    return str(dict1) + str(dict2) + json.dumps(dict3) #str(dict3)


if __name__ == "__main__":
    try:
        dict1 = {'a':1,'b':2,'c':3}
        dict2 = {'d':4,'e':5,'f':6}
        dict3 = {'g':7,'h':8,'i':9}

        #finding biggest dict
        biggest = biggestDict(dict1,dict2,dict3)
        print "biggest dict is ", biggest

        #adding new elements to dict
        addDictToAnotherDict(dict1,{'j':10,'k':11})
        print dict1
        addToDict(dict2,'l',12)
        print dict2

        #printing length
        dict1_len = findLength(dict1)
        print "length of",dict1,"is",dict1_len
        dict2_len = findLength(dict2)
        print "length of",dict2,"is",dict2_len
        dict3_len = findLength(dict3)
        print "length of",dict3,"is",dict3_len

        #convert to string and concatenate
        str = convertToStringAndConcatenate(dict1,dict2,dict3)
        print "Final string is",str

    except Exception as e:
        print(e)
```

38.Create 2 dictionaries as follows;
dict1 ={'Name':'Ramakrishna','Age':25}
dict2={'EmpId':1234,'Salary':5000}
Perform following operations
a) Create single dictionary by merging dict1 and dict2
b) Update the salary to 10%
c) Update the age to 26
d) Insert the new element with key "grade" and assign value as "B1"
e) Extract and print all values and keys separately.
f) delete the element with key 'Age' and print dictionary elements.

```
def concatenateDict(tempdict1,tempdict2):
    tempdict1.update(tempdict2)

def updateDictSalary(tempdict1,Salary,percentageIncrease):
    tempdict1['Salary'] =
int(tempdict1['Salary']*(1+(percentageIncrease/100.0)))
```

```
def updateDict(tempdict1,key,value):
    tempdict1[key] = value

def printDict(tempdict1):
    for key,values in tempdict1.items():
        print key,"=",values
    #delete 'Age'

def deleteElement(tempdict1,key):
    del(tempdict1[key])


if __name__ == "__main__":
    try:
        dict1 ={'Name':'Ramakrishna','Age':25}
        dict2={'EmpId':1234,'Salary':5000}

        #concatenate
        concatenateDict(dict1,dict2)
        print dict1

        #updateSalaryBy10%percent
        updateDictSalary(dict1,'Salary',10)
        print dict1

        #updateage to 26
        updateDict(dict1,'Age',26)
        print dict1

        #insert into dict
        updateDict(dict1,'Grade','B1')
        print dict1

        #print elements in dict
        printDict(dict1)

        #delete elements in dict
        deleteElement(dict1,'Age')
        print dict1

    except Exception as e:
        print(e)
```

39. Using Time and Calendar module, Print current date and time. Print current month calendar.

```
import datetime
import calendar

def printCurrentDateTime(today):
    formattedDateTime = today.strftime("%d/%m/%Y %H:%M:%S")
    print formattedDateTime

def printCurrentMonthCalendar(today):
    print calendar.month(today.year,today.month)

if __name__ == "__main__":
    try:
      today = datetime.datetime.now()
      printCurrentDateTime(today)
      printCurrentMonthCalendar(today)
    except Exception as e:
       print(e)
```

40. Using time module perform following operations.
a) Print current time for every 5 seconds up to 1 minute time interval.
b) Write a program to find out how much CPU time is taken for the execution of above(32.a) program.

a)

```
import time
import threading

def currentTime(count,WaitSeconds):
    print(time.ctime())
    count -= 1
    if count!=0:
       threading.Timer(WaitSeconds,
currentTime,[count,WaitSeconds]).start()

def printCurrentTime(totalmins=1,WaitSeconds=2):
    count = (totalmins*60)/WaitSeconds + 1
    currentTime(count,WaitSeconds)

if __name__ == "__main__":
    try:
        printCurrentTime(1,5)
    except Exception as e:
        print(e)
```

b)

```
from subprocess import call
import time

def executionTime(path):
    starttime = time.time()
    call(["Python","{}".format(path)])
    endtime = time.time()
    return(endtime - starttime)

if __name__ == "__main__":
    try:
        program32_path = "C:\\Users\\Abcd\\Desktop\\32nd.py"
        timetaken = executionTime(program32_path)
        print timetaken
    except Exception as e:
        print(e)
```