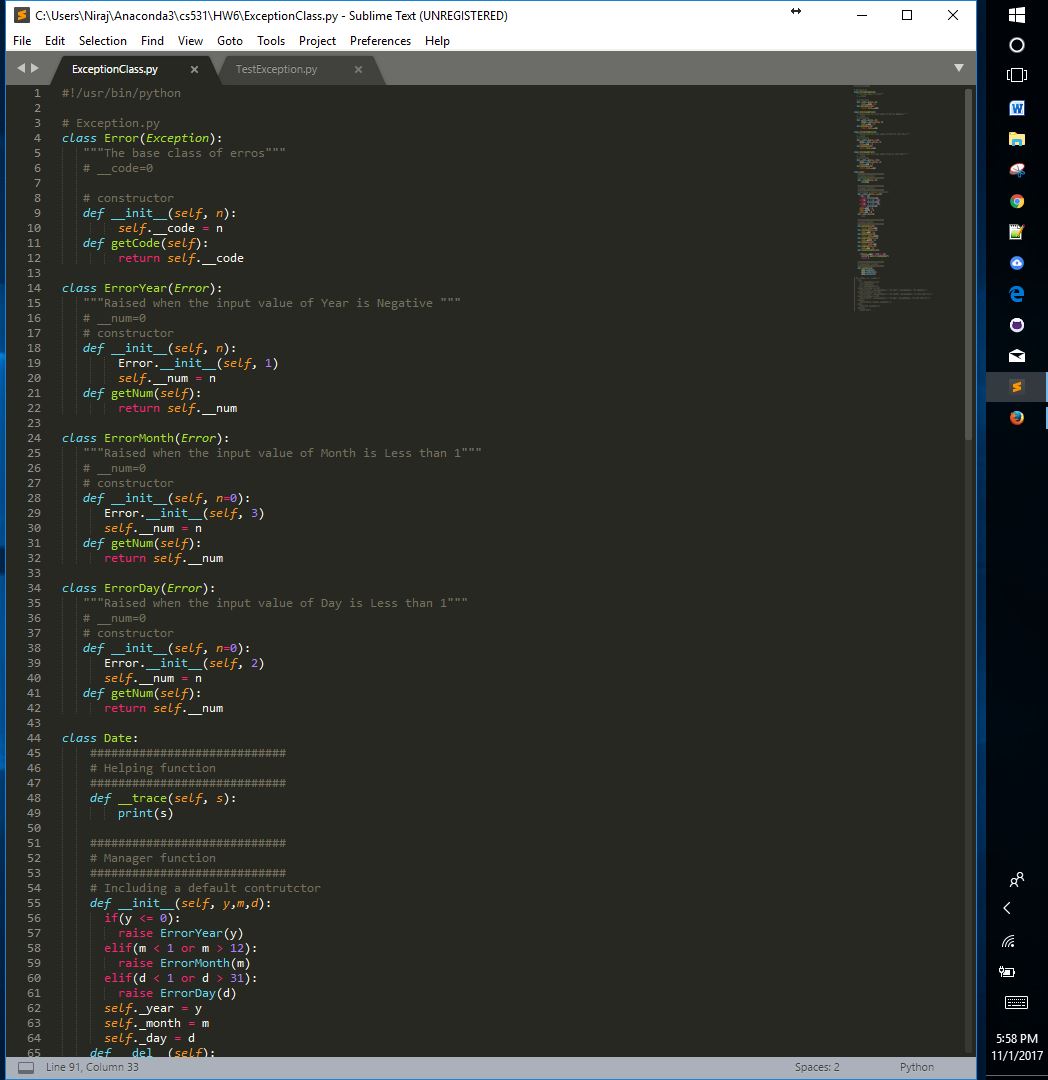
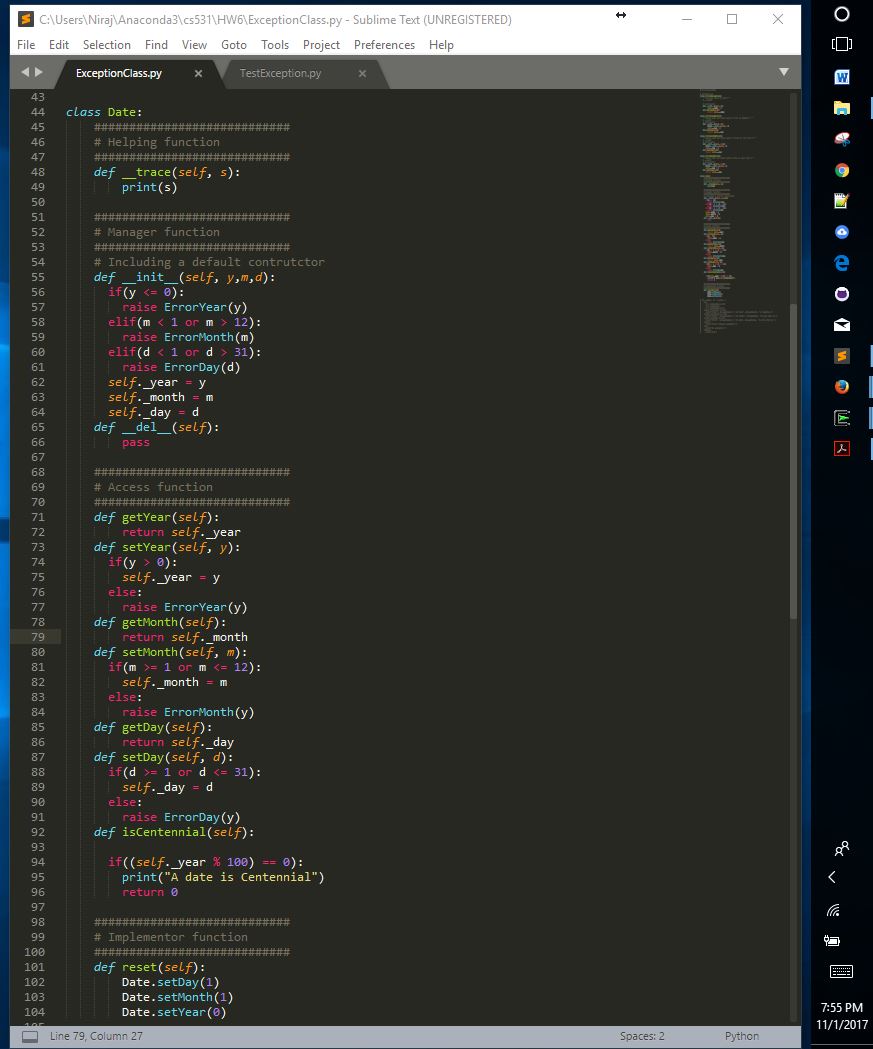
**Name : Niraj Thanki SID : 19376 CLASS : CS531**

**Exception.py**

****

****

**Source Code:**

#!/usr/bin/python

# Exception.py

class Error(Exception):

"""The base class of erros"""

# \_\_code=0

# constructor

def \_\_init\_\_(self, n):

self.\_\_code = n

def getCode(self):

return self.\_\_code

class ErrorYear(Error):

"""Raised when the input value of Year is Negative """

# \_\_num=0

# constructor

def \_\_init\_\_(self, n):

Error.\_\_init\_\_(self, 1)

self.\_\_num = n

def getNum(self):

return self.\_\_num

class ErrorMonth(Error):

"""Raised when the input value of Month is Less than 1"""

# \_\_num=0

# constructor

def \_\_init\_\_(self, n=0):

Error.\_\_init\_\_(self, 3)

self.\_\_num = n

def getNum(self):

return self.\_\_num

class ErrorDay(Error):

"""Raised when the input value of Day is Less than 1"""

# \_\_num=0

# constructor

def \_\_init\_\_(self, n=0):

Error.\_\_init\_\_(self, 2)

self.\_\_num = n

def getNum(self):

return self.\_\_num

class Date:

############################

# Helping function

############################

def \_\_trace(self, s):

print(s)

############################

# Manager function

############################

# Including a default contrutctor

def \_\_init\_\_(self, y,m,d):

if(y <= 0):

raise ErrorYear(y)

elif(m < 1 or m > 12):

raise ErrorMonth(m)

elif(d < 1 or d > 31):

raise ErrorDay(d)

self.\_year = y

self.\_month = m

self.\_day = d

def \_\_del\_\_(self):

pass

############################

# Access function

############################

def getYear(self):

return self.\_year

def setYear(self, y):

if(y > 0):

self.\_year = y

else:

raise ErrorYear(y)

def getMonth(self):

return self.\_month

def setMonth(self, m):

if(m >= 1 or m <= 12):

self.\_month = m

else:

raise ErrorMonth(y)

def getDay(self):

return self.\_day

def setDay(self, d):

if(d >= 1 or d <= 31):

self.\_day = d

else:

raise ErrorDay(y)

def isCentennial(self):

if((self.\_year % 100) == 0):

print("A date is Centennial")

return 0

############################

# Implementor function

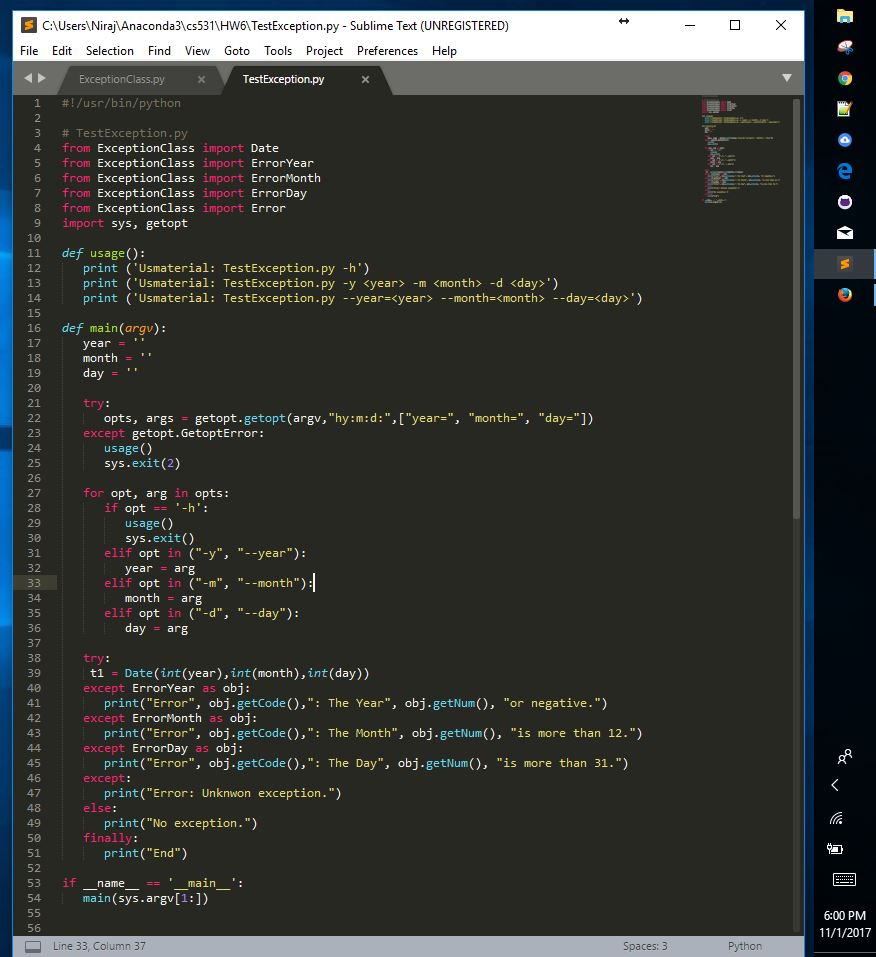
############################

def reset(self):

Date.setDay(1)

Date.setMonth(1)

Date.setYear(0)

**TestException.py**

**Source Code:**

#!/usr/bin/python

# TestException.py

from ExceptionClass import Date

from ExceptionClass import ErrorYear

from ExceptionClass import ErrorMonth

from ExceptionClass import ErrorDay

from ExceptionClass import Error

import sys, getopt

def usage():

print ('Usmaterial: TestException.py -h')

print ('Usmaterial: TestException.py -y <year> -m <month> -d <day>')

print ('Usmaterial: TestException.py --year=<year> --month=<month> --day=<day>')

def main(argv):

year = ''

month = ''

day = ''

try:

opts, args = getopt.getopt(argv,"hy:m:d:",["year=", "month=", "day="])

except getopt.GetoptError:

usage()

sys.exit(2)

for opt, arg in opts:

if opt == '-h':

usage()

sys.exit()

elif opt in ("-y", "--year"):

year = arg

elif opt in ("-m", "--month"):

month = arg

elif opt in ("-d", "--day"):

day = arg

try:

t1 = Date(int(year),int(month),int(day))

except ErrorYear as obj:

print("Error", obj.getCode(),": The Year", obj.getNum(), "or negative.")

except ErrorMonth as obj:

print("Error", obj.getCode(),": The Month", obj.getNum(), "is more than 12.")

except ErrorDay as obj:

print("Error", obj.getCode(),": The Day", obj.getNum(), "is more than 31.")

except:

print("Error: Unknwon exception.")

else:

print("No exception.")

finally:

print("End")

if \_\_name\_\_ == '\_\_main\_\_':

main(sys.argv[1:])

**Output :**

