41. Using calendar module perform following operations.
a) Print the 2016 calendar with space between months as 10 characters.
b) How many leap days between the years 1980 to 2025.
c) Check given year is leap year or not.
d) print calendar of any specified month of the year 2016.

```
import calendar


class CalendarOperations:
    def displayOneYearCalendar(self,year):
        print calendar.calendar(year,c=10)

    def leapdaysCount(self,startyear,endyear):
        return calendar.leapdays(startyear,endyear)

    def isLeapYear(self,year):
        return calendar.isleap(year)

    def displayOneMonthCalendar(self,month,year=2016):
        print calendar.month(2016,month)

if __name__ == "__main__":
    try:
        c1 = CalendarOperations()
        #1year calendar
        c1.displayOneYearCalendar(2016)
        #leapdays count
        no_of_leapdays = c1.leapdaysCount(1980,2025)
        print "Leap days between 1980 to 2025 is",no_of_leapdays
        #check leap year or not
        year = input("Enter the year to check leap year or not")
        print c1.isLeapYear(year)
        #1month calendar
        month  = input("Enter the month which is needed to be displayed")
        c1.displayOneMonthCalendar(month)

    except Exception as e:
        print e
```

42. Write a program to generate a Fibonacci series using a function called fib(n), a) Where ‘n’ is user specified argument specifies number of elements in the series.

```
def fib(n):
    a,b=0,1
    while(n>0):
        print a,
        c = a+b
        a=b
        b=c
        n-=1

if __name__ == "__main__":
    try:
        n = int(input("Enter the number of elements in the series: "))
        fib(n)
    except ValueError:
        print "Enter only int values"
    except Exception as e:
        print e
```

43. Write a program to search given element from the list. Use your own function to search an element from list. Note: Function should receive variable length arguments and search each of these arguments present in the list.

```
def search(studentnamelist,name):
    for index in range(len(studentnamelist)):
        if studentnamelist[index] == name:
            return index
    return -1

def printFoundIndex(index):
    if index == -1:
        print "Element not found"
    else:
        print "Element found at index", index



if __name__ == "__main__":
    try:
        studentnamelist = 
['Abinesh','Aravindhan','Karthikeyan','ClintonAntony','Jofus','Balaji',"Nar
ayanan",'Bhuvanesh','Kannan','Logesh','Deepan']
        name = input("Enter the name to be searched: ")
        index = search(studentnamelist,name)
        printFoundIndex(index)
    except Exception as e:
        print e
```

44. Write a program with lambda function to perform following.
a) Perform all the operations of basic calculator (Add, Sub, Multiply, Divide, Modulus, Floor division)

```
import math

add = lambda *args : sum(args)
sub = lambda x,y : x-y
mul = lambda x,y : x*y
div = lambda x,y : x/y
mod = lambda x,y : x%y
floor = lambda x : math.floor(x)

if __name__ == "__main__":
    try:
        print add(6,4)
        print sub(6,4)
        print mul(6,4)
        print div(6,4)
        print mod(6,4)
        print floor(6.82134)
    except Exception as e:
        print e
```

45. Write a program to check given string is Palindrome or not. (Use function Concepts and Required keyword, Default parameter concepts) i.e Reverse the given string and check whether it is same as original string, if so then it is palindrome. Example: String "malayalam" when reversed will be "malayalam" hence palindrome.

```
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
```

46. Write a function to find the biggest of 4 numbers.
   a) All numbers are passed as arguments separately (Required argument)
   b) use default values for arguments (Default arguments)

```
class operationOnNumbers:
  def biggest(self,a=22,b=54,c=100,d=16):
    if a>b and a>c and a>d:
      return a
    elif b>c and b>d:
      return b
    elif c>d:
      return c
    return d

if __name__ == '__main__':
  try:
    num1 = int(input("Enter number1"))
    num2 = int(input("Enter number2"))
    num3 = int(input("Enter number3"))
    num4 = int(input("Enter number4"))

    obj1 = operationOnNumbers()
    #a - required arguments
    maxnum = obj1.biggest(num1,num2,num3,num4)
    print maxnum
    #b - default arguments
    maxnum = obj1.biggest()
    print maxnum
  except ValueError:
    print "Enter only int values"
  except Exception as e:
    print e
```

47. Write function to extend the tuple with elements of list. Pass list and Tuple as parameter to the function.

```
class TupleOperation:
    def extendTuple(self,temptuple1,templist1):
        return temptuple1  + tuple(templist1)

if __name__ == "__main__":
    try:
        tuple1 = (1,2,3,4,5)
        list1 = [6,7,8,9]
        tuple1 = TupleOperation().extendTuple(tuple1,list1)
        print tuple1
    except Exception as e:
        print e
```

48. Create a Calculator with the following functions.
a) Addition/subtraction/multiplication and division of two numbers (Note: Create separate function for each operation)
b) Find square root of a given number. (Use keyword arguments in your function)
c) Create a list of sub strings from a given string, such that sub strings are created with given character. i.e. String = "Pack: My: Box: With: Good: Food"
Create sub strings with the delimiter character ":" such that the following sub strings are created. substrlist=[Pack, My, Box, With, Good, Food] Note : Function should take at least 2 parameters (Main string and delimiter character) return value from function will be list of substring.

```
import math
class Calculator:
    def add(self,num1,num2):
        return num1+num2
    def sub(self,num1,num2):
        return num1-num2
    def mul(self,num1,num2):
        return num1*num2
    def div(self,num1,num2):
        return num1/num2
    def sqrt(self,num1):
        return math.sqrt(num1)

class StringOperation:
    def listofSubstr(self,tempstr1,delimiter):
        return tempstr1.split(delimiter)

if __name__ == '__main__':
    try:
        num1 = int(input("Enter number1"))
        num2 = int(input("Enter number2"))

        obj1 = Calculator()
        sumvalue =  obj1.add(num1,num2)
        print "add of two numbers is",sumvalue
        difference = obj1.sub(num1,num2)
        print "sub of two numbers is",difference
        product = obj1.mul(num1,num2)
        print "mul of two numbers is",product
        div = obj1.div(num1,num2)
        print "div of two numbers is",div
        squareroot = obj1.sqrt(num1)
```

```
            print "Square root of",num1,"is",squareroot


            str1 = "Pack: My: Box: With: Good: Food"
            delimiter = ':'
            list1 = StringOperation().listofSubstr(str1,delimiter)
            print list1
        except ValueError:
            print "Enter only int values"
        except Exception as e:
            print e
```

49. Write a program to perform following file operations
a) Open the file in read mode and read all its contents on to STDOUT.
b) Open the file in write mode and enter 5 new lines of strings in to the new file.
c) Open file in Append mode and add 5 lines of text into it.

```
class FileOperations:
    def fileread(self,path):
        f = open(path,'r')
        print f.read()
        f.close()


    def filewrite(self,path,content,AppendOrWrite='w'):
        f = open(path,AppendOrWrite)
        f.write(content)
        f.close()


if __name__ =="__main__":
```

```
try:
    samplefile1_path = "C:\Users\priya\Desktop\samplefile.txt"
    obj1 = FileOperations()

    obj1.fileread(samplefile1_path)

    quotes = "aaaaa\nbbbbb\ncccccc\nddddd\neeeee"
    obj1.filewrite(samplefile1_path,quotes,"w")

    additionalquotes = "\nffffff\nggggggg\nhhhhhh\niiiiii"
    obj1.filewrite(samplefile1_path,additionalquotes,"a")

    obj1.fileread(samplefile1_path)
except IOError:
    print "Please check the file path"
except Exception as e:
    print e
```

50. Write a program to open the existing file in read mode and perform following tasks,
a) Read 10 character at a time and then print its current position of file object. Repeat this operation till the EOF.
b) Reset the file pointer after reading 100 Character from file (Use Seek function to reset)
c) Open the file in read mode and start printing the contents from 5th line onwards.

```
class FileOperations:
    def fileread(self,path):
        f = open(path,'r')
        while True:
            c = f.read(10)
            if not c:
                break
            print c
            print "Current Position=",f.tell(),"\n"

        f.close()

    def read_N_charsOnly(self,path,n):
        f = open(path,'r')
        print f.read(100)
        f.seek(0,0)

    def printFromNthLine(self,path,lineno):
        f = open(path,'r')
        for line in range(lineno-1):
            f.readline()
        print f.read()



if __name__ =="__main__":
    try:
        obj1 = FileOperations()

        obj1.fileread("C:\Users\priya\Desktop\samplefile.txt")

obj1.read_N_charsOnly("C:\Users\priya\Desktop\samplefile.txt",100)
```

```
            obj1.printFromNthLine("C:\Users\priya\Desktop\samplefile.txt",5)

        except IOError:
            print "Please check the file path"
        except Exception as e:
            print e
```

51. In a given directory search all text files for the pattern "Treasure".
a) Find how many text files has the pattern.
b) Count how many times pattern repeats in each file Note: Create at least 4 text files in a directory and keep the pattern in at least 2 files. Repeat the pattern in the file many times.

```
class FileOperations:
    def fileread(self,path,name):
        f = open(path,'r')
        counter=0
        while True:
            content = f.readline()
            if not content:
                break
            counter += content.count("Treasure")
        print name , "has" ,counter,"times the word Treasure"
        f.close()

    def checkFiles(self,files_path,files_name):
        for filepath,filename in zip(files_path,files_name):
            self.fileread(filepath,filename)
```

```
if __name__ =="__main__":
    try:
        obj1 = FileOperations()


        files_path = 
["C:\Users\Abcd\Desktop\samplefile.txt","C:\Users\Abcd\Desktop\GK\Wipro\Assignment3\samplefile1.txt","C:\Users\Abcd\Desktop\GK\Wipro\Assignment3\samplefile2.txt","C:\Users\Abcd\Desktop\GK\Wipro\Assignment3\samplefile3.txt"]
        files_name = 
["SampleFile1","SampleFile2","SampleFile3","SampleFile4"]
        obj1.checkFiles(files_path,files_name)


    except IOError:
        print "Please check the file path"
    except Exception as e:
        print e
```

52. Open existing text file and reverse its contents. i.e
a) print the last line as first line and first line as last line (Reverse the lines of the file)
b) print characters of file from last character of file till the first character of the file.(Reverse entire contents of file)

```
class FileOperations:
    def fileReverseLines(self,path):
        f = open(path,'r+')
        k = reversed(f.readlines())
        f.seek(0,0)
        for i in k:
            f.write(i)
    def fileReverseContent(self,path):
        f = open(path,'r+')
        k = reversed(f.readlines())
        f.seek(0,0)
        for i in k:
            f.write(i[::-1])
    def fileread(self,path):
        f = open(path,'r')
        print "\n", f.read()

if __name__ =="__main__":
    try:
        obj1 = FileOperations()
        samplefile1_path = "C:\Users\Abcd\Desktop\samplefile1.txt"
        samplefile2_path = "C:\Users\Abcd\Desktop\samplefile2.txt"

        #reversing lines
        obj1.fileReverseLines(samplefile1_path)
        #printing the file content after reversing
        obj1.fileread(samplefile1_path)

        #reversing entire content
        obj1.fileReverseContent(samplefile2_path)
        #printing the file content after reversing
        obj1.fileread(samplefile2_path)
    except IOError:
        print "Please check the file path"
    except Exception as e:
        print e
```

53. Open the file is read & write mode and apply following functions
All 13 functions mentioned in Tutorial File object table.

```
def fileFunctions(samplefile1_path):
    f = open(samplefile1_path, "r+")
    print "Name of the file: ", f.name  #name
    f.flush() #flush
    fid = f.fileno() #fileno
    print "File Descriptor: ", fid
    ret = f.isatty() #isatty
    print "Return value : ", ret
    f.seek(0,0) #seek
    line = f.read(10)  #read
    print "Read Line: %s" % (line)
    line = f.readline() #readline
    print "Read Line: %s" % (line)
    content = f.readlines(5) #readlines
    print "Read Lines: %s" % (content)
    pos = f.tell() #tell
    print "Current Position: %d" % (pos)
    f.truncate() #truncate
    line = f.readline()
    print "Read Line after truncate: %s" % (line)
```

```
        f.seek(0, 2)
        line = f.write("Hello") #write
        f.seek(0, 3)
        line = f.writelines("Welcome to python programming") #writelines
        f.seek(0,0)
        for index in range(3):
          line = f.next()   #next
          print "Line No %d - %s" % (index, line)
        f.close()
if __name__ =="__main__":
    try:
        samplefile1_path = "C:\Users\Abcd\Desktop\samplefile.txt"
        fileFunctions(samplefile1_path)
    except IOError as e:
        print e
    except Exception as e:
        print e
```

54. Write a program to handle the following exceptions in you program.
a) KeyboardInterrupt
b) NameError
c) ArithmeticError Note: Make use of try, except, else: block statements.

```
class ArithmeticOperations:
    def div(self,num1,num2):
        try:
            return num1/num2
        except ArithmeticError:
            print "Arithemtic Exception occurred"


if __name__ =="__main__":
    try:
        ArithmeticOperations().div(12,0)
        result += 10
    except NameError:
        print "Name does not exists"
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
    except Exception as e:
        print e
    else:
        print "Executed error free"
    finally:
        print "End of the program"
```

55. Write a program for converting weight from Pound to Kilo grams.
a) Use assertion for the negative weight.
b) Use assertion to weight more than 100 KG

```
class Conversion:
    def kilosToPounds(self,kilo):
        assert(kilo<100), "Weight more than 100 kilograms"
        assert(kilo>0), "Negative value"
        return kilo*2.2
if __name__ =="__main__":
    obj1 = Conversion()
    val1 = obj1.kilosToPounds(10); print val1
    val2 = obj1.kilosToPounds(-3); print val2
    val3 = obj1.kilosToPounds(123); print val3
```

56. Write a program to handle following exceptions using Try block.
a) IO Error while you try writing contents into the file that is opened in read mode only.
b) ValueError

```
class FileOperations:
    def readFile(self,path):
        try:
            f = open(path,"r")
            f.write("Hello") #IOError
            try:
                print int(f.read(5)) #value error
            except ValueError:
                print "Can convert values to expected datatype"
        except IOError:
            print "InputOutputException Occurred"
        finally:
            f.close()

if __name__ =="__main__":
```

```
try:
  path = "C:\Users\priya\Desktop\samplefile.txt"
  FileOperations().readFile(path)
except Exception as e:
      print e
```

57. Try implementing atleast any 5 exceptions in you program.

```
class FileOperations:
   def fileread(self,path):
      f = open(path,'r')
      print f.read()
      f.close()

if __name__ =="__main__":
   try:
      obj1 = FileOperations()
      obj1.fileread("C:\Users\Abcd\Desktop\samplefile.txt") #IOError
      a = int('abcd') #ValueError
      b = "df"+34.56 #TypeError
      c += 10 #NameError
      d = 54%0 #ArithmeticError
   except IOError:
      print "Please check the file path"
   except TypeError:
      print "Inappropriate type"
   except ValueError:
        print "Can't convert values to expected datatype"
    except NameError:
        print "Name you specified doesn't exist"
    except ArithmeticError:
        print "Arithmetic Exception occurred"
    except Exception as e:
        print e
```

58. Create file called "calc.py" which has following functions
i) functions to add 2 numbers
ii) function to find diff of 2 numbers
iii) function to multiply 2 numbers
iv) all maths operations (sqrt, div, floor div, modulus, primenumber)
v) Fibonacci series

a) Write a new program in file "maths.py" such that you import functions of file "calc.py" to your new program
b) Use From <module> import <function> statement to import only few function from calc module.

calc.py

```
def add(x,y):
  return x + y
def subtract(x,y):
  return x - y
def multiply(x,y):
  return x * y
def divide(x,y):
  return x / y
def floordiv(x,y):
    return x // y
```

```
def squareroot(x):
  return x ** 0.5
def modulus(x,y):
    return x % y
def isPrime(num):
    i=2
    while((i*i)<=num):
        if(num%i==0):
            return False
        i += 1
    return True

def fib(x):
    a,b=0,1
    while(x>0):
        print (a),
        c = a+b
        a=b
        b=c
        x-=1
```

maths.py

```
from calc import add,subtract,floordiv,modulus,squareroot

if __name__ =="__main__":
    try:
        val1 = input('Enter Value1: ')
        val2 = input('Enter Value2: ')


        sumVal = add(val1,val2)
        subVal = subtract(val1,val2)
        floordivVal = floordiv(val1,val2)
        sqrtval = squareroot(val1)
        modVal = modulus(val1,val2)
        print(sumVal,subVal,floordivVal,sqrtval,modVal),


    except ArithmeticError:
        print "Arithmetic Exception occurred"
    except Exception as e:
        print e
```

59. Create file called "stringop.py" which has following functions
i) functions to sort numbers (Use loops for sorting, do not use built in function)
ii) function to search given element through binary search method.(Refer to net for the Binary search algorithm)
iii) function to reverse the given string

Write new program in file strpackage.py such that you import functions of file "stringop.py" to your new program

stringop

```
def selectionsort(templist1):
    for i in range(len(templist1)):
        min=i
        for j in range(i+1,len(templist1)):
            if(templist1[min]>templist1[j]):
                min=j
        templist1[i],templist1[min] = templist1[min],templist1[i]
    return templist1

def binarysearch(templist1,x):
    start = 0
    end = len(templist1)-1
    while(start<=end):
        mid = start + (end-start)/2
        if templist1[mid] == x:
            return mid
        elif x > templist1[mid]:
            start = mid+1
        else:
            end = mid-1
    return -1

def reversestring(tempstr):
    return tempstr[::-1]
```

strpackage

```
from stringop import *

if __name__ == "__main__":
    try:
        list1 = [2,3,12,34,1,234,23,455,6778,233,22,12,11]
        sortedlist = selectionsort(list1) #sorting
        x = int(input("Enter the value to be found: "))
        position = binarysearch(sortedlist,x) #binary search
        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
```

60. Create a package of all programs you have done in earlier.

a. All programs related to strings - Stringpackage
b. All programs related to Lists -ListPackage
c. All programs related to Tuple - TuplePackage
d. All programs related to Dictionary -DictionaryPackage
e. All programs related to Functions - FunctionPackage
f. All programs related to Files -- FilePackage

Uploaded as separate folder