Python 3 - Functions and OOPs_FP

1. hands on python string methods

```
#
# Complete the 'strmethod' function below.
#
# The function accepts following parameters:
#  1. STRING para
#  2. STRING spch1
#  3. STRING spch2
#  4. LIST li1
#  5. STRING strf
#

def stringmethod(para, special1, special2, list1, strfind):
    for myChar in special1:
        para = para.replace(myChar,"")
    word1=para

    rword2=word1[69::-1]
    print(rword2)

    myspaceremove=rword2.replace(" ", "")
    myspaceremove=myspaceremove.replace("  ", "")
    myword= special2.join(myspaceremove)
    print(myword)

    if all(SearchStr in para for SearchStr in list1):
        print("Every string in  %s were present" %list1)
    else:
        print("Every string in  %s were not present" %list1)

    number=word1
    splitAll = number.split()
    print(splitAll[0:20])

    mylist=[]
    myDict = dict()
    for t in splitAll:
        myDict[t]=myDict.get(t,0)+1
    for x,y in myDict.items():
        if(y<3):
            mylist.append(x)
    print(mylist[-20:])

    print(word1.rfind(strfind))
```

1 magic constant

```
#
# Complete the 'Magic_const' function below.
#
#
#
# The function accepts INTEGER n1 as parameter.
#

def generator_Magic(n1):
    for i in range(3, n1+1):
        gen1 = i * ((i**2) + 1) // 2
        yield gen1
```

Python 3 Functions and OOPs - Prime Number Generator

```
#
# Complete the 'primegenerator' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER num
#  2. INTEGER val
#

def primegenerator(num, val):
    # Write your code here
    x = 0
    for i in range(2,num):
        if(len([j for j in range(2,i-1) if i%j==0]) == 0):
            x = x + 1
            if(int(val) == 0):
                if (x % 2) == 0:
                    yield i
            if(int(val) == 1):
                if (x % 2) != 0:
                    yield i
```

Python 3 - Functions and OOPs - Classes and Objects 1

Define a class 'Movie' that represents

```
# Write your code here
class Movie:
    def __init__(self, val1, val2, val3):
        self.movieName= val1
        self.numberTickets = val2
        self.totalCost = val3

    def __str__(self):
        #return self.movieName
        #return self.numberTickets
        #return self.totalCost
        return "Movie : {}\nNumber of Tickets : {}\nTotal Cost :
{}".format(self.movieName,self.numberTickets,self.totalCost)
```

task 2

Define the class 'comp' that represents the Real and Imaginary

```
#Write your code here
class comp(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary

    def add(self, other):
        sum = complex(self.real + other.real, self.imaginary + other.imaginary)
        sum = str(sum).replace('j','i').replace('(','').replace(')','')
        print ("Sum of the two Complex numbers :" +sum)

    def sub(self, other):
        diff = complex(self.real - other.real, self.imaginary - other.imaginary)
        diff = str(diff).replace('j','i').replace('(','').replace(')','')
        if diff == "0i":
            diff = "0+0i"
        print ("Subtraction of the two Complex numbers :" +diff)
```

Addition and Subtraction of Complex Numbers

2. Classes and Objects

Polymorphism

```
class rectangle:
    def display(self):
        print("This is a Rectangle")

    def area(self, Length, Breadth):
        area = Length * Breadth
        print("Area of Rectangle is ", area)

class square:
    def display(self):
        print("This is a Square")

    def area(self, Side):
        area = Side * Side
        print("Area of square is ", area)
```

1. Handling Exceptions - 1

Exception handling 1.

```
# Complete the 'Handle_Exc1' function below.

def Handle_Exc1():
    a=int(input())
    b=int(input())
    c=a+b
    if a>150 or b<100:
        raise ValueError("Input integers value out of range.")
    elif c>400:
        raise ValueError("Their sum is out of range")
    else:
        print("All in range")
```

1. Handling Exceptions - 2

Write the function definition for the function 'FORLoop'

```
# Complete the 'FORLoop' function below.

def FORLoop():
    n = int(input())
    l1 = [0]*n
    for i in range(len(l1)):
        l1[i]=int(input())

    print(l1[0:n])

    iter1 = iter(l1)
    for i in range(len(l1)):
        print(next(iter1))
    return iter1

Handling Exceptions 3

Bank_ATM

# Complete the 'Bank_ATM' function below.
#
# Define the Class for user-defined exceptions "MinimumDepositError" and
"MinimumBalanceError" here
class MinimumDepositError(Exception):
    pass
class MinimumBalanceError(Exception):
    pass

def Bank_ATM(balance,choice,amount):
    #print(balance)
    #print(choice)
    #print(amount)
    if balance < 500:
        raise(ValueError("As per the Minimum Balance Policy, Balance must be at
least 500"))
    if choice == 1:
        if amount < 2000:
            raise(MinimumDepositError("The Minimum amount of Deposit should be
2000."))
        else:
            balance+=amount
            print("Updated Balance Amount: ",balance)

    if choice == 2:
        if balance-amount < 500:
            raise(MinimumBalanceError("You cannot withdraw this amount due to
Minimum Balance Policy"))
        else:
            balance-=amount
            print("Updated Balance Amount: ",balance)


Classes and Objects 2 - Task 1

Inheritance - Parent and Children Shares

# It is expected to create two child classes 'son' & 'daughter' for the above
class 'parent'
#
#Write your code here

# Parent class created
class parent:
  def __init__(self,total_asset):
    self.total_asset = total_asset

  def display(self):
    print("Total Asset Worth is "+str(self.total_asset)+" Million.")
    print("Share of Parents is "+str(round(self.total_asset/2,2))+" Million.")

class son(parent):
    def __init__(self, total, val):
        parent.__init__(self, total)
        self.Percentage_for_son = val
        self.total = total
    def son_display(self):
        x = self.total * (self.Percentage_for_son / 100)
        x = round(x,2)
        print("Share of Son is {} Million.".format(x))

class daughter(parent):
    def __init__(self, total, val):
        parent.__init__(self, total)
        self.Percentage_for_daughter = val
        self.total = total

    def daughter_display(self):
        x = self.total * (self.Percentage_for_daughter / 100)
        x = round(x,2)
        print("Share of Daughter is {} Million.".format(x))
```

1. Handling Exceptions 4

Library Harry Potter

```
# Complete the 'Library' function below.
#


def Library(memberfee,installment,book):
    # Write your code here
    #print(memberfee)
    #print(installment)
    #print(book)

    if installment > 3:
        raise(ValueError("Maximum Permitted Number of Installments is 3"))

    if installment == 0:
        raise(ZeroDivisionError("Number of Installments cannot be Zero."))
    else:
        print ("Amount per Installment is ", memberfee / installment)

    if book == 'Philosophers stone' or book == 'Chamber of Secrets' or book ==
'prisoner of azkaban' or book == 'Goblet of Fire' or book == 'order of phoenix'
or book == 'Half Blood Prince' or book == 'Deathly Hallows 1' or  book ==
'deathly hallows 2':
        print ("It is available in this section")
    else:
        raise(NameError("No such book exists in this section"))
```

Python DateTime

```
# Complete the 'dateandtime' function below.
#
# The function accepts INTEGER val as parameter.
# The return type must be LIST.
#
import datetime
def dateandtime(val,tup):
    # Write your code here

    # tup = 1 and 4 : 3 values (year, month, date)
    # tup = 2 : single value (timestamp)
    # tup = 3 : 3 values (hours, mins, secs)
    # tup = 5: 5 values (year, month , date, hours, mins, secs)

    list = []
    if val == 1:
        d = datetime.date(tup[0],tup[1],tup[2])
        list.append(d)
        dd = d.strftime('%d/%m/%Y')
        list.append(dd)
    if val == 2:
        d = datetime.datetime.fromtimestamp(int(tup[0]))
        d = d.date()
        list.append(d)
    if val == 3:
        d = datetime.time(tup[0],tup[1],tup[2])
        list.append(d)
        h = d.strftime("%I")
        list.append(h)
    if val == 4:
        d = datetime.date(tup[0],tup[1],tup[2])
        #list.append(d)
        weekday = d.strftime('%A')
        list.append(weekday)
        fullmonth = d.strftime('%B')
        list.append(fullmonth)
        day = d.strftime('%j')
        list.append(day)
    if val == 5:
        d = datetime.datetime(tup[0],tup[1],tup[2],tup[3],tup[4],tup[5])
        list.append(d)
    return list
```

Python - Itertools (NOT YET COMPLETED)

```
#
# Complete the 'usingiter' function below.
#
# The function is expected to return a TUPLE.
# The function accepts following parameters:
#  1. TUPLE tupb
#
import itertools
import operator
from itertools import chain

def performIterator(tuplevalues):
    # Write your code here
    mainlist = list()
    list1 = list()

    for var in range(len(tuplevalues[0])):
        list1.append(tuplevalues[0][var])

    mainlist.append(list1[0:4])

    list2 = list()

    for var in range(len(tuplevalues[1])):
        list2.append(tuplevalues[1][var])
    num = int(list2[0])

    tupcount = len(tuplevalues[1])
    rep = list(itertools.repeat(num,tupcount))
    mainlist.append(rep)

    tup3 = tuplevalues[2]

    result = itertools.accumulate(tup3,operator.add)
    list3 = list()

    for each in result:
        list3.append(each)

    mainlist.append(list3)

    length = len(tuplevalues)
    list4 = list()

    for i in range(length):
        for var in range(len(tuplevalues[i])):
            list4.append(tuplevalues[i][var])
    mainlist.append(list4)

    only_odd = [num for num in list4 if num % 2 ==1]
    mainlist.append(only_odd)
    mainlist = str(mainlist).replace('[','(').replace(']',')')
    return(mainlist)
```

Python - Cryptography

```
#
# Complete the 'encrdecr' function below.
#
# The function is expected to return a LIST.
# The function accepts following parameters:
#  1. STRING keyval
#  2. STRING textencr
#  3. Byte-code textdecr
#
from cryptography.fernet import Fernet

def encrdecr(keyval, textencr, textdecr):
    res = []
    cipher = Fernet(keyval)
    encrval = cipher.encrypt(textencr)
    res.append(encrval)
    decrbytes = cipher.decrypt(textdecr)
    res.append(decrbytes.decode('utf8'))
    return res
```

Python - Calendar

```
#
# Complete the 'calen' function below.
#
# The function accepts TUPLE datetuple as parameter.
#

import calendar
import datetime
from collections import Counter

def usingcalendar(datetuple):
    # Write your code here
    year=int(datetuple[0])
    mon=datetuple[1]
    if year % 4== 0 or year % 100 == 0 or year % 400 == 0:
        mon=2
    date=calendar.TextCalendar(calendar.MONDAY)
    print(date.formatmonth(year,mon))
    l = []
    obj = calendar.Calendar()
    for day in obj.itermonthdates(year, mon):
        l.append(day)
    rev = l[:-8:-1]
    rev.reverse()
    print(rev)
    count=Counter(d.strftime('%A') for d in obj.itermonthdates(year,mon) if
d.month==mon)
    for i,j in count.most_common(1):
        print(i)
```

Python - Collections - NOT COMPLETED

```
#
# Complete the 'collectionfunc' function below.
#
# The function accepts following parameters:
#  1. STRING text1
#  2. DICTIONARY dictionary1
#  3. LIST key1
#  4. LIST val1
#  5. DICTIONARY deduct
#  6. LIST list1
#

import collections

def collectionfunc(text1, dictionary1, key1, val1, deduct, list1):
    # Write your code here
    tmp = list()
    mydict = dict()
    li = list(text1.split(" "))

    items = collections.Counter(li)
    y = sorted(items.items())
    fix = str(y).replace('[(', '{').replace(')]',
'}').replace('\',','\':').replace('), (',', ')
    print(fix)

    items = collections.Counter(dictionary1)

    res1 = {key: items[key] - deduct.get(key, 0) for key in items.keys()}
    res2 = {key: items[key] - deduct.get(key, 0) for key in deduct.keys()}
    res = {**res1, **res2}
    print(res)

    keyval = dict(zip(key1, val1))
    count = int()
    for k,v in keyval.items():
        count += 1
        if count == 2:
            keyval.pop(k)
            break
```

```
keyval.update([(k, v)])
print(keyval)

even=list()
odd=list()

for i in list1:
    if (i % 2) == 0:
        even.append(i)
    else:
        odd.append(i)

oedict = {}

if len(odd) > 0 and len(even) > 0:
    print("{'odd': " + str(odd) + ", 'even': " + str(even) + "}")
elif len(odd) == 0 and len(even) > 0:
    print("{'even': " + str(even) + "}")
elif len(odd) > 0 and len(even) == 0:
    print("{'odd': " + str(odd) + "}")
else:
    print(oedict)
```