Exception Handling

Program 1

Python Program should get the Name, Age, 6 Subjects marks as an input from the user. Then generate the dictionary as below

Before generating an dictionary you need to check whether entered age value is positive or not. If negative then we should add the details to the dictionary.

Use User-Defined Exception Handling

In [1]:
```
class ValueError(Exception):
    pass
try:
    name=input("enter name:")
    age=int(input("enter age:"))
    t=int(input("enter t mark:"))
    e=int(input("enter e mark:"))
    m=int(input("enter m mark:"))
    s=int(input("enter s mark:"))
    ss=int(input("enter ss mark:"))
    h=int(input("enter h mark:"))
    dict={}
    if(age<0):
        dict={"Name":name ,"Age":age ,"Tamil":t ,"English":e ,"Maths":m ,"Science":s ,"
        print(dict)
    if((age>0) or (age==0)):
        raise ValueError
except(ValueError):
          print("enter only negative numbers")
```

```
{'Name': 'selva', 'Age': -2, 'Tamil': 11, 'English': 22, 'Maths': 33, 'Science': 44, 'So
cial': 55, 'Hindi': 66}
```

Program 2

Python Program should read the Name, Roll Number and the 6 Subject marks.

While entering the marks if we have entering the marks greater than 100 or less than 0 then it should throw an error (**User generated Exception**).

For calculating the total,average, minimum and maximum use function

```
def calculate_average_total(marks):




        raise



    return total,avg,min_mark,max_mark
```

Sample Output

```
Enter the Student Name: Hari
Enter the Roll Number :150
Enter the 6 subject marks
Enter Subject 1 Marks :
89
Enter Subject 2 Marks :
78
Enter Subject 3 Marks :
56
Enter Subject 4 Marks :
150
Enter Subject 5 Marks :
56
Enter Subject 6 Marks :
56

---------------------------------------------------------------------------
InvalidMarkError                          Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_26140/2527275713.py in <module>
      6     print("Enter Subject",i+1,"Marks : ")
      7     marks.append(int(input()))
----> 8 t,a,m,ma = calculate_average_total(marks)
      9 if m >=50:
     10     print("Total = ",t)

~\AppData\Local\Temp/ipykernel_26140/20899550.py in calculate_average_total(marks)
      7     for mark in marks:
      8         if mark < 0 or mark > 100:
----> 9             raise InvalidMarkError ("Mark Entered is Less than 0 or greater than 100")
     10         total += mark
     11     avg = round(total / 6, 2)

InvalidMarkError: Mark Entered is Less than 0 or greater than 100
```

In [12]:

```
class InvalidMarkError(Exception):
    pass
try:
    name=input("enter name:")
    roll_no=int(input("enter rollno:"))
    t=int(input("enter t mark:"))
    e=int(input("enter e mark:"))
    m=int(input("enter m mark:"))
    s=int(input("enter s mark:"))
    ss=int(input("enter ss mark:"))
    h=int(input("enter h mark:"))
    if((t>0) and(e>0) and(m>0) and(s>0) and(ss>0) and(h>0)):
        if((t<100) and(e<100) and(m<100) and(s<100) and(ss<100) and(h<100)):
            def calculate_average_total(*marks):
```

```
                total=t+e+m+s+ss+h
                average=total/6
                minimum=min(marks)
                maximum=max(marks)
                return total,average,minimum,maximum
            total,average,minimum,maximum =calculate_average_total(t,e,m,s,ss,h)

    if((t<0) or(e<0) or(m<0) or(s<0) or(ss<0) or(h<0)or(t>100) or(e>100) or(m>100) or(s
        raise InvalidMarkError
except(InvalidMarkError):
    print("Mark should be entered between 0 and 100 ")
else:
    print("total:",total)
    print("average :",average)
    print("maximum:",maximum)
    print("minimum:",minimum)
```

```
total: 297
average : 49.5
maximum: 77
minimum: 22
```

Now Handle the above using Try and Execpt Block.

If all the marks are entered correctly then it should display the Total, Average and his minimum and maximum mark.

For calculating the total,average, minimum and maximum use function

```
def calculate_average_total(marks):
    total = 0
    try:



    except InvalidMarkError:

        return 0,0,0,0


    return total,avg,min_mark,max_mark
```

If the entered mark is invalid then the function should "0" if it is valid then function should return total,average,minimum and maximum

Use User-Defined Exception

In addition to above add one condition when displaying the total,average,minimum and maximum.

if he got marks less that 50 then just display "have failed", else display the all the detials.

Now change the code accordingly and display the output as below

Sample output 1 - with valid marks

```
Enter the Student Name: Hari
Enter the Roll Number :150
Enter the 6 subject marks
Enter Subject 1 Marks :
89
Enter Subject 2 Marks :
89
Enter Subject 3 Marks :
78
Enter Subject 4 Marks :
56
Enter Subject 5 Marks :
89
Enter Subject 6 Marks :
78
Total =  479
Average =  79.83
Minium Mark =  56
Maximum Mark =  89
```

Sample output 1 - with valid marks but he FAILED

```
Enter the Student Name: GOKUL
Enter the Roll Number :145
Enter the 6 subject marks
Enter Subject 1 Marks :
89
Enter Subject 2 Marks :
78
Enter Subject 3 Marks :
65
Enter Subject 4 Marks :
48
Enter Subject 5 Marks :
89
Enter Subject 6 Marks :
78
GOKUL  have failed
```

Sample output 2 - with invalid marks

```
Enter the Student Name: JAGADESH
Enter the Roll Number :155
Enter the 6 subject marks
Enter Subject 1 Marks :
89
Enter Subject 2 Marks :
78
Enter Subject 3 Marks :
89
Enter Subject 4 Marks :
78
Enter Subject 5 Marks :
98
Enter Subject 6 Marks :
105
Mark should be between 0 to 100
You have entered invalid mark(s)
```

In [5]:

```
class InvalidMarkError(Exception):
    pass
class FailMarkError(Exception):
    pass
try:
    name=input("enter name:")
    roll_no=int(input("enter rollno:"))
    t=int(input("enter t mark:"))
    e=int(input("enter e mark:"))
    m=int(input("enter m mark:"))
    s=int(input("enter s mark:"))
    ss=int(input("enter ss mark:"))
    h=int(input("enter h mark:"))
    if((t>0) and(e>0) and(m>0) and(s>0) and(ss>0) and(h>0)):
        if((t<100) and(e<100) and(m<100) and(s<100) and(ss<100) and(h<100)):

            def calculate_average_total(*marks):
                total=t+e+m+s+ss+h
                average=total/6
                minimum=min(marks)
                maximum=max(marks)
                return total,average,minimum,maximum
            total,average,minimum,maximum =calculate_average_total(t,e,m,s,ss,h)


    if((t<50) or(e<50) or(m<50) or(s<50) or(ss<50) or(h<50)):
         raise FailMarkError


    if((t<0) or(e<0) or(m<0) or(s<0) or(ss<0) or(h<0)or(t>100) or(e>100) or(m>100) or(s
        raise InvalidMarkError

except(FailMarkError):
    print(name,"have failed")
```

```
except(InvalidMarkError):
    print("Mark should be entered between 0 and 100 ")
    print("You have entered invalid marks")


else:
    print("total:",total)
    print("average :",average)
    print("maximum:",maximum)
    print("minimum:",minimum)
```

```
selva have failed
```

FILES

PROGRAM 1

Write a function in python to count the number of lines from a text file "proverb.txt" which is not starting with an alphabet "D".

Example:

If the file "proverb.txt" contains the following lines:

Many hands make light work

Honesty is the best policy

Don't bite the hand that feeds you

Don't judge a book by its cover

Birds of a feather flock together

Diamonds cut diamonds

In [22]:
```
file=open("proverb.txt","r")
count=0
for i in file:
    count+=1
print("the number of lines:",count)
```

```
the number of lines: 7
```

Program 2

Python Program that display the occurance of a particular character or a string in give input file "proverb.txt"

If the file **"proverb.txt"** contains the following lines:

Many hands make light work

Honesty is the best policy

Don't bite the hand that feeds you

Don't judge a book by its cover

Birds of a feather flock together

Diamonds cut diamonds

Sample Output

```
Enter the character or string to be search: the
The character/string the has occured 3 times in the file
```

In [31]:

```
file=open("proverb.txt","r")
a=input("enter the character or string to be search:")
count=0
for i in file:
    for j in i.split(" "):
        if j ==a:
            count+=1
print("The character/string",a,"has occured",count,"times in the file" )
```

```
The character/string the has occured 3 times in the file
```

Program 3

Write a python code to append the following proverb to the **"proverb.txt"** file.

"Diligence is the mother of good fortune"

After updating the file content must be, read the file content and display it.

```
Enter the proverb to be added : Diligence is the mother of good fortune
Many hands make light work
Honesty is the best policy
Donâ€™t bite the hand that feeds you
Donâ€™t judge a book by its cover
Birds of a feather flock together
Diamonds cut diamonds
Diligence is the mother of good fortune
```

In [38]:

```
file=open("proverb.txt","a+")
b=str(input("enter the proverb to be added:"))
file.write("\n")
file.write(b)
file.seek(0)
print(file.read())
```

```
Many hands make light work
Honesty is the best policy
Don’t bite the hand that feeds you
Don’t judge a book by its cover
Birds of a feather flock together
Diamonds cut diamonds
Diligence is the mother of good fortune
```

Program 4

Now convert all the character in the file "proverb.txt" to uppercase and write to a new file "u_proverb.txt" and display the new contents files also

```
MANY HANDS MAKE LIGHT WORK
HONESTY IS THE BEST POLICY
DONÂ€™T BITE THE HAND THAT FEEDS YOU
DONÂ€™T JUDGE A BOOK BY ITS COVER
BIRDS OF A FEATHER FLOCK TOGETHER
DIAMONDS CUT DIAMONDS
DILIGENCE IS THE MOTHER OF GOOD FORTUNE
```

In [30]:

```
file=open("proverb.txt","r")
file1=open("u_proverb.txt","a+")
for i in file:
    file1.write(i.upper())
file1.seek(0)
print(file1.read())
```

```
MANY HANDS MAKE LIGHT WORK
HONESTY IS THE BEST POLICY
DON’T BITE THE HAND THAT FEEDS YOU
DON’T JUDGE A BOOK BY ITS COVER
BIRDS OF A FEATHER FLOCK TOGETHER
DIAMONDS CUT DIAMONDS
DILIGENCE IS THE MOTHER OF GOOD FORTUNE
```