Nested Try Except Block

In [ ]:

```
--> Nested Try blocks are possible in Python
--> Then based on Try block we need to same number of except block.
--> if you are using try block then it is forsure that you need to use atleast one excepy block
--> if try block is not getting any error then except block never be executed
--> Whenever you are using try exexcept block your program will terminate normalle
--> Nested Try except block is also possible but for each try block there must be an except block
```

Example

In [1]:

```python
#Nested Try Except Block
print("Stmt-1")
try:
    print(10/0)
    print("stmt-2")
except:
    try:
        print(5/0)
    except:
        try:
            print(20/0)
        except:
            print("stmt-3")
print("Stmt-4")
```

```
Stmt-1
stmt-3
Stmt-4
```

Try with Multiple Except Block

In [ ]:

```
--> You can use multiple except block with single Try
--> the except block must be the last block of the code if you are using except block in between the blocks then it will give
    you an error you cannot use except block in between any block.
```

Example

In [2]:

```python
#try with multiple except block
try:
    a=int(input("Enter a Number : "))
    b=int(input("Enter a Number : "))
    print(a/b)
except ValueError:
    print("Please give integer input ")
except ZeroDivisionError:
    print("Denominator cannot be zero")
except:
    print("Except block")
```

```
Enter a Number : 12
Enter a Number : ten
Please give integer input
```

Finally Block

In [ ]:

```
Sometimes we need something to exccute in our program weather the exception is occured or not,
finally block is always be excuted weather the exception is occured or not. Basciallu finally block
are used to perform cleanup activities(DB connection closinh , resource allocation , gc)

Example:
    try:
        risky code
    except:
        alternative code/handling code
    finally:
        cleanup code
```

Important Cases

In [3]:

```python
#Case1:
try:
    print("hello")
except:
    print("world")
finally:
    print("Hello")
```

```
hello
Hello
```

In [4]:

```python
#Case2:
try:
    print("inside Try")
    print(10/0)
except:
    print("Except")
finally:
    print("Finally")
```

```
inside Try
Except
Finally
```

In [5]:

```python
#Case3:
try:
    print("inside Try")
    print(10/0)
except NameError:
    print("Except")
finally:
    print("Finally")
```

```
inside Try
Finally
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Input In [5], in <cell line: 2>()
      2 try:
      3     print("inside Try")
----> 4     print(10/0)
      5 except NameError:
      6     print("Except")

ZeroDivisionError: division by zero
```

In [6]:

```python
#Case 4:
try:
    print("Inside Try")
    print(10/0)
except:
    print("Except Block")
except ValueError:
    print("Except")
finally:
    print("Finally")
```

```
  Input In [6]
    print(10/0)
    ^
SyntaxError: default 'except:' must be last
```

Else Block

In [ ]:

```
If no any error is there in try block then else part will executed.
```

Example

In [7]:

```python
try:
    print("try")
    print(10/2)
except:
    print("Except")
else:
    print("Else")
finally:
    print("Finally")
```

```
try
5.0
Else
Finally
```

Types of Exceptions

In [ ]:

```
Two Types of Exceptions:
    Predefined Exceptions --> for each exception a sepreate class is given we can use that class
    Example:
        Zerodivisionerror, nameerror,valueerror,eof error etc

    Userdefined Exceptions --> Such type of exceptions that are defined by developer are known as UserDefined Exceptions
                               or Customised exceptions.
    Example: too young exception ,toooldexception , insufiicientfund
```

Creation of User Defined Exception:

In [ ]:

```
Creation of User Defined Exception:
    1. Each and every userdefined exception is a child class of exception class.
    2. raise keyword we need to use for raising user defined exceptions.
    3. We can also pass a message while raising the User Defined Exceptions.
```

Example

In [31]:

```python
class TooYoungException(Exception):
    def __init__(self,x):
        self.x=x
age=int(input())
if age<18:
    raise  TooYoungException("Your age is less than 18")
else:
    print("You are perfect")
```

```
12
---------------------------------------------------------------------------
TooYoungException                         Traceback (most recent call last)
Input In [31], in <cell line: 5>()
      4 age=int(input())
      5 if age<18:
----> 6     raise  TooYoungException("Your age is less than 18")
      7 else:
      8     print("You are perfect")

TooYoungException: Your age is less than 18
```

In [32]:

```python
class Insuffient_Balance(Exception):
    def __init__(self,x):
        self.x=x
age=int(input())
if age<18:
    raise  Insuffient_Balance("Your age is less than 18")
else:
    print("You are perfect")
```

```
12
---------------------------------------------------------------------------
Insuffient_Balance                        Traceback (most recent call last)
Input In [32], in <cell line: 5>()
      4 age=int(input())
      5 if age<18:
----> 6     raise  Insuffient_Balance("Your age is less than 18")
      7 else:
      8     print("You are perfect")

Insuffient_Balance: Your age is less than 18
```

CSV File Handling

In [ ]:

```
CSV Files -->  A Comma Separated Values (CSV) file is a plain text file that stores data by delimiting data entries with commas.
          -->  if we want to store the data in form of excel(csv) then we can use csv file handling concept.
          -->  for using csv file in python  you need to import one module that is known as csv
Example:

        121, pratyush,9721372543,ashu
```

Write Operation in CSV Files

In [ ]:

```
For writing Purpose we in csv file we need to use 3 different Methods:

    writer method --> it ensures in which file you need to write the data
    writerow method --> it is used to write the data in the first row that will used as a coloumn for rest of the data.
                        In this function you need to pass argument as a list elemnt.
```

Example

In [35]:

```python
import csv
with open("employees.csv","w",newline="") as s:
    w=csv.writer(s)
    w.writerow(["Employee_Name","Employee_Number","Employee_Salary","Employee_Mobile"])
    n=int(input("Enter How many employee data you want to store: "))
    for i in  range(n):
        Employee_Name=input("Enter Employee Name")
        Employee_Number=int(input("Enter Employee Number"))
        Employee_Salary=int(input("Enter Employee Salary "))
        Employee_Mobile_no=int(input("Enter Mobile Number"))
        w.writerow([Employee_Name,Employee_Number,Employee_Salary,Employee_Mobile_no])
print("Total student data is updated")
```

```
Enter How many employee data you want to store: 4
Enter Employee NameKrish
Enter Employee Number90
Enter Employee Salary 90000000
Enter Mobile Number9721378976
Enter Employee NameArnav
Enter Employee Number923
Enter Employee Salary 90000000
Enter Mobile Number986
Enter Employee Namename
Enter Employee Number98
Enter Employee Salary 98
Enter Mobile Number98
Enter Employee Nametushar
Enter Employee Number99
Enter Employee Salary 99
Enter Mobile Number99
Total student data is updated
```

Read Operation in CSV File

In [ ]:

```
for reading purpose we are having one method named as:

    reader() --> In this function we need to pass file pointer.
```

Example

In [39]:

```python
#Reader --> which file you want to read
import csv
f=open("employees.csv","r")
x=csv.reader(f) #reader
y = list(x)
for i in y:
    print(i)
```

```
['Employee_Name', 'Employee_Number', 'Employee_Salary', 'Employee_Mobile']
['Krish', '90', '90000000', '9721378976']
['Arnav', '923', '90000000', '986']
['name', '98', '98', '98']
['tushar', '99', '99', '99']
```

Read Operation Using Pandas

In [41]:

```python
import pandas as pd
df=pd.read_csv("employees.csv")
df.head(5)
```

Out[41]:

	Employee_Name	Employee_Number	Employee_Salary	Employee_Mobile
0	Krish	90	90000000	9721378976
1	Arnav	923	90000000	986
2	name	98	98	98
3	tushar	99	99	99

In [ ]: