In [1]:
```
#Q1
```

In [1]:
```
#An exception is an unexpected event that occurs during the program execution.
#Exceptions can be caught and handle by the program.
```

In [3]:
```
#errors mostly happen at compiletime like syntaxerrors however it can happen at run
#Missing parentheses, incorrect indentation, and misspelled keywords are examples o


#Exceptions occurs at runtime.Try-except blocks, the raise command, and finally blo
#The program halts when an exception occurs and switches to the closest exception h
#There are 2 types of exceptions 1)built-in exception 2)User defined exception.
```

In [4]:
```
#Q2
```

In [5]:
```
#If the assertion fails, Python uses ArgumentExpression as the argument for the Ass
#Python provides two very important features to handle any unexpected error in your
#1)Exception handling 2) Assertions

#standard exceptions are:- 1)ZeroDivisionError 2)FileNotFoundError 3)ZeroDivisionEr
```

In [6]:
```
#Q3
```

In [7]:
```
#Try and Except block is used to handle exceptions
try:
    numerator = 10
    denominator = 0

    result = numerator/denominator

    print(result)
except:
    print("Error: Denominator cannot be 0.")

#In the above example Here, we have placed the code that might generate an exceptio

#When an exception occurs, it is caught by the except block. The except block canno
```

```
Error: Denominator cannot be 0.
```

In [8]:
```
#Q4
```

In [9]:
```
# Try block execute when the exception is generate.Else block is executed when try
# block execute without the error.


# Finally:-finally block is always executed after leaving the try statement.
#In case if some exception was not handled by except block, it is re-raised after e
#The 'finally' block contains code that will always be executed, regardless of whet
```

In [10]:
```
#Q5
```

In [12]:
```
# In Python, we can define custom exceptions by creating a new class that is derive
# Here, CustomError is a user-defined error which inherits from the Exception class

# When we are developing a large Python program, it is a good practice to place all
# Many standard modules define their exceptions separately as exceptions.py or erro
```

In [13]:
```
#Ex
class CustomError(Exception):
    ...
    pass

try:
   ...

except CustomError:
    ...
```

In [14]:
```
#Q6
```

In [16]:
```
#Here, we have overridden the constructor of the Exception class to accept our own

#Then, the constructor of the parent Exception class is called manually with the se

#The custom self.salary attribute is defined to be used later.

#The inherited __str__ method of the Exception class is then used to display the co
```

In [17]:
```
#
class SalaryNotInRangeError(Exception):
    """Exception raised for errors in the input salary.

    Attributes:
        salary -- input salary which caused the error
        message -- explanation of the error
    """

    def __init__(self, salary, message="Salary is not in (5000, 15000) range"):
        self.salary = salary
        self.message = message
        super().__init__(self.message)


salary = int(input("Enter salary amount: "))
if not 5000 < salary < 15000:
    raise SalaryNotInRangeError(salary)
```

```
---------------------------------------------------------------------------
SalaryNotInRangeError                     Traceback (most recent call last)
Cell In[17], line 18
     16 salary = int(input("Enter salary amount: "))
     17 if not 5000 < salary < 15000:
---> 18     raise SalaryNotInRangeError(salary)

SalaryNotInRangeError: Salary is not in (5000, 15000) range
```

In [ ]: