Q1.

What is an Exception in Python? Write the difference between Exceptions and Syntax errors.

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.

An error is an issue in a program that prevents the program from completing its task. In comparison, an exception is a condition that interrupts the normal flow of the program.

Q2.

What happens when an exception is not handled? Explain with an example

unhandled exception - program usually terminates abruptly.

In [1]:
```
def divide(a, b):
    return a / b

try:
    result = divide(5, 0)
    print("Result:", result)
except ValueError:
    print("Caught a ValueError")
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-8d073782af74> in <module>
      3
      4 try:
----> 5     result = divide(5, 0)
      6     print("Result:", result)
      7 except ValueError:

<ipython-input-1-8d073782af74> in divide(a, b)
      1 def divide(a, b):
----> 2     return a / b
      3
      4 try:
      5     result = divide(5, 0)

ZeroDivisionError: division by zero
```

Q3.

Which Pythn statements are used to catch and handle exceptions? Explain with an example

the try and except statements are used to catch and handle exceptions

In [2]:
```
def divide(a, b):
    return a / b

try:
    result = divide(5, 0)
    print("Result:", result)
except ValueError:
    print("Caught a ValueError")
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-8d073782af74> in <module>
      3
      4 try:
----> 5     result = divide(5, 0)
      6     print("Result:", result)
      7 except ValueError:

<ipython-input-2-8d073782af74> in divide(a, b)
      1 def divide(a, b):
----> 2     return a / b
      3
      4 try:
      5     result = divide(5, 0)

ZeroDivisionError: division by zero
```

Q4.

Explain with an example

a. try and else

In [3]:
```
try:
    num1 = int(input("Enter a numerator: "))
    num2 = int(input("Enter a denominator: "))
    result = num1 / num2
except ValueError:
    print("Error: Please enter valid numbers.")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
else:
    print("Result of division:", result)
```

```
Enter a numerator: 1
Enter a denominator: 2
Result of division: 0.5
```

b. finally

In [4]:
```
try:
    num1 = int(input("Enter a numerator: "))
    num2 = int(input("Enter a denominator: "))
    result = num1 / num2
except ValueError:
    print("Error: Please enter valid numbers.")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
else:
    print("Result of division:", result)
finally :
    print("File operation completed")
```

```
Enter a numerator: 1
Enter a denominator: 1
Result of division: 1.0
File operation completed
```

c. raise

In [5]:
```
def check_temperature(temperature):
    if temperature > 100:
        raise ValueError("Temperature is too high!")
    elif temperature < 0:
        raise ValueError("Temperature is too low!")
    else:
        print("Temperature is within the acceptable range.")

try:
    temperature = float(input("Enter the temperature: "))
    check_temperature(temperature)
except ValueError as ve:
    print("Error:", ve)
```

```
Enter the temperature: 96
Temperature is within the acceptable range.
```

Q5.

What are custom exceptions in python? Why do we need Custom exceptions? Explain with an example.

custom exceptions are user-defined exceptions that extend the built-in exception classes.

Need of custom exceptions :

In [6]:
```python
# Q6. and Example for Q5.

class WithdrawalError(Exception):
    def __init__(self, balance, amount):
        self.balance = balance
        self.amount = amount
        super().__init__(f"Insufficient balance ({balance}) for withdrawal of {

def perform_withdrawal(balance, amount):
    if balance >= amount:
        print("Withdrawal successful!")
    else:
        raise WithdrawalError(balance, amount)

try:
    account_balance = 500
    withdrawal_amount = 700
    perform_withdrawal(account_balance, withdrawal_amount)
except WithdrawalError as e:
    print("Error:", e)
```

```
Error: Insufficient balance (500) for withdrawal of 700
```

In [ ]: