Q1 what is an exception in python? write the defference betwen exception and syntax error

In Python, an exception is an error that occurs during the execution of a program. When an exceptional event happens, an object representing that event, called an exception object, is created and thrown, causing the current function or statement to immediately stop executing. Exceptions can be caused by various reasons, such as invalid input, incorrect use of functions, network or file I/O errors, and so on.

On the other hand, a syntax error occurs when the Python interpreter encounters a code that violates the rules of the Python language syntax. This could be a missing or misplaced symbol, a typo, or a syntax construct used incorrectly.

The main difference between exceptions and syntax errors is that syntax errors occur during the compilation or parsing of the Python code, while exceptions occur during the runtime of the code. Syntax errors prevent the code from running altogether, whereas exceptions can occur during the execution of the code and can be handled or caught by the programmer.

In summary, syntax errors occur due to incorrect syntax of the Python language, while exceptions occur during the execution of the program due to errors or unexpected events in the code.

Q2 what happen when an exception is not handel ?explain with example

When an exception is not handled, it results in an error message and the program terminates abruptly. The error message displays the type of exception that occurred, the location in the code where the exception occurred, and a traceback of the function calls that led to the exception.

Here's an example that raises an exception and does not handle it:

def divide_by_zero(): return 1 / 0

divide_by_zero()

Q3 .Which Python statementsare used to catch and handle exceptions? explain with example

Python, the try-except statements are used to catch and handle exceptions. The try block contains the code that might raise an exception, and the except block catches and handles the exception if it occurs. If no exception is raised, the except block is skipped.

Here's an example of using try-except to handle an exception

In [11]:
```
#Q3 EXAMPLE

def divide_numbers(num1, num2):
    try:
        result = num1 / num2
        return result
    except ZeroDivisionError:
        print("Cannot divide by zero")
        return None
```

```
print(divide_numbers(10, 2)) # Output: 5.0
print(divide_numbers(10, 0)) # Output: Cannot divide by zero\nNone
```

```
5.0
Cannot divide by zero
None
```

In [17]:
```
#Q4
try :
    f= open("test.txt" , 'w')
    f.write(" write into my file " )
    f.close()
except Exception  as e :
    print("this is my except block " , e)
else :
    f.close()
    print ('thise will be once your try will execute without eroor  ')
```

```
thise will be once your try will execute without eroor
```

In [18]:
```
#Q4
try :
    g = open ('koni.king','w')
    g.write('koni is busy now dont disturb me')
finally:
    print('vishal koni is now busy dont try to disturb him')
```

```
vishal koni is now busy dont try to disturb him
```

In [20]:
```
#Q4
def vallidateage(age ):
    if age < 0:
        raise vallidateage ('enterd age is negative')
    elif age > 200 :
        raise vallidateage ('enterd age is veery veery high ')
    else:
        print('age is vallid')
```

In [25]:
```
#Q6 create a custeom exception class . use thise to handel an exception


class NegativeNumberError(Exception):
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return f"{self.value} is a negative number. Please provide a non-negative n


def calculate_square_root(num):
    if num < 0:
        raise NegativeNumberError(num)
    else:
        return num ** 0.5


try:
    result = calculate_square_root(-9)
    print(result)
except NegativeNumberError as e:
    print(e)
```

```
-9 is a negative number. Please provide a non-negative number.
```

In [ ]: