NAME: **Meet Raut**

DIV: **S2-1**

ROLL NO: **2201084**

EXPERIMENT – 11:

- ***AIM:*** **To study and implement program on Exception Handling.**
- ***THEORY:***

What is an Exception?

An exception in Python is an incident that happens while executing a program that causes the regular course of the program's commands to be disrupted. When a Python code comes across a condition it can't handle, it raises an exception. An object in Python that describes an error is called an exception.

When a Python code throws an exception, it has two options: handle the exception immediately or stop and quit.

Try and Except Statement - Catching Exceptions

In Python, we catch exceptions and handle them using try and except code blocks. The try clause contains the code that can raise an exception, while the except clause contains the code lines that handle the exception.

How to Raise an Exception

If a condition does not meet our criteria but is correct according to the Python interpreter, we can intentionally raise an exception using the raise keyword. We can use a customized exception in conjunction with the statement.

Assertions in Python

When we're finished verifying the program, an assertion is a consistency test that we can switch on or off.

The simplest way to understand an assertion is to compare it with an if-then condition. An exception is thrown if the outcome is false when an expression is evaluated.

Assertions are made via the assert statement, which was added in Python 1.5 as the latest keyword.

Assertions are commonly used at the beginning of a function to inspect for valid input and at the end of calling the function to inspect for valid output.

The assert Statement

Python examines the adjacent expression, preferably true when it finds an assert statement. Python throws an AssertionError exception if the result of the expression is false.

The syntax for the assert clause is –

```
assert Expressions[, Argument]
```

Python uses ArgumentException, if the assertion fails, as the argument for the AssertionError. We can use the try-except clause to catch and handle AssertionError exceptions, but if they aren't, the program will stop, and the Python interpreter will generate a traceback.

Try with Else Clause

Python also supports the else clause, which should come after every except clause, in the try, and except blocks. Only when the try clause fails to throw an exception the Python interpreter goes on to the else block.

Finally Keyword in Python

The finally keyword is available in Python, and it is always used after the try-except block. The finally code block is always executed after the try block has terminated normally or after the try block has terminated for some other reason.

User-Defined Exceptions

By inheriting classes from the typical built-in exceptions, Python also lets us design our customized exceptions.

Here is an illustration of a RuntimeError. In this case, a class that derives from RuntimeError is produced. Once an exception is detected, we can use this to display additional detailed information.

We raise a user-defined exception in the try block and then handle the exception in the except block.

Python Exceptions List

Here is the complete list of Python in-built exceptions.

Sr.No.	Name of the Exception	Description of the Exception
1	**Exception**	All exceptions of Python have a base class.
2	**StopIteration**	If the next() method returns null for an iterator, this exception is raised.
3	**SystemExit**	The sys.exit() procedure raises this value.
4	**StandardError**	Excluding the StopIteration and SystemExit, this is the base class for all Python built-in exceptions.
5	**ArithmeticError**	All mathematical computation errors belong to this base class.
6	**OverflowError**	This exception is raised when a computation surpasses the numeric data type's maximum limit.
7	**FloatingPointError**	If a floating-point operation fails, this exception is raised.
8	**ZeroDivisionError**	For all numeric data types, its value is raised whenever a number is attempted to be divided by zero.
9	**AssertionError**	If the Assert statement fails, this exception is raised.
10	**AttributeError**	This exception is raised if a variable reference or assigning a value fails.
11	**EOFError**	When the endpoint of the file is approached, and the interpreter didn't get any input value by raw_input() or input() functions, this exception is raised.
12	**ImportError**	This exception is raised if using the import keyword to import a module fails.

13	**KeyboardInterrupt**	If the user interrupts the execution of a program, generally by hitting Ctrl+C, this exception is raised.
14	**LookupError**	LookupErrorBase is the base class for all search errors.
15	**IndexError**	This exception is raised when the index attempted to be accessed is not found.
16	**KeyError**	When the given key is not found in the dictionary to be found in, this exception is raised.
17	**NameError**	This exception is raised when a variable isn't located in either local or global namespace.
18	**UnboundLocalError**	This exception is raised when we try to access a local variable inside a function, and the variable has not been assigned any value.
19	**EnvironmentError**	All exceptions that arise beyond the Python environment have this base class.
20	**IOError**	If an input or output action fails, like when using the print command or the open() function to access a file that does not exist, this exception is raised.
22	**SyntaxError**	This exception is raised whenever a syntax error occurs in our program.
23	**IndentationError**	This exception was raised when we made an improper indentation.
24	**SystemExit**	This exception is raised when the sys.exit() method is used to terminate the Python interpreter. The parser exits if the situation is not addressed within the code.
25	**TypeError**	This exception is raised whenever a data type-incompatible action or function is tried to be executed.
26	**ValueError**	This exception is raised if the parameters for a built-in method for a particular data type are of the correct type but have been given the wrong values.

27	**RuntimeError**	This exception is raised when an error that occurred during the program's execution cannot be classified.
28	**NotImplementedError**	If an abstract function that the user must define in an inherited class is not defined, this exception is raised.

- **PROGRAM:**

```
for i in range (1,10):
    print("\n1. Zero Division Error\n 2. Value Error\n 3. Type Error\n 4. Index Error\n 5. Assertion Error")


    ch=int(input("Enter Your Choice: "))

    if(ch==1):
        try:
            a=int(input("Enter a: "))
            b=int(input("Enter b: "))
            d=a/b
            print("Division: ",d)
        except ZeroDivisionError:
            print("Zero division error occurs")
        print("Test")

    elif(ch==2):
        try:
            a=int(input("Enter a: "))
            b=int(input("Enter b: "))
            d=a/b
```

```
        print("Division: ",d)
    except ValueError:
        print("Value error occurs")
    print("Test")

elif(ch==3):
    try:
        a=int(input("Enter a: "))
        b=int(input("Enter b: "))
        d=a/'b'
        print("Division: ",d)
    except TypeError:
        print("Type error occurs")
    print("Test")

elif(ch==4):
    try:
        a=[1,2,3,4]
        print("Index 5= ",a[5])
    except IndexError:
        print("Index error occurs")
    print("Test")

elif(ch==5):
    a=int(input("Enter number b/w 2 to 6: "))
    try:
        assert a>=2 and a<=6
        print("Entered number is: ",a)
    except AssertionError:
```

```
        print("Assertion error occurs")
    print("Test")

else:
    print("Invalid Choice")

print("Thank You")
```

- **OUTPUT:**

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 1
Enter a: 12
Enter b: 6
Division:  2.0
Test
Thank You

1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 1
Enter a: 12
Enter b: 0
Zero division error occurs
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 2
Enter a: 2
Enter b: a
Value error occurs
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 2
Enter a: 2
Enter b: 2
Division:  1.0
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 3
Enter a: 12
Enter b: 2
Type error occurs
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 4
Index error occurs
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 5
Enter number b/w 2 to 6: 3
Entered number is:  3
Test
Thank You

1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 5
Enter number b/w 2 to 6: 7
Assertion error occurs
Test
Thank You
```

```
1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice: 6
Invalid Choice
Thank You

1. Zero Division Error
 2. Value Error
 3. Type Error
 4. Index Error
 5. Assertion Error
Enter Your Choice:
```

- ***CONCLUSION:*** **Hence, we have successfully implemented program on Exception Handling; LO 1.**