

Exception Handling

1. What is the role of try and exception block?

ANSWER: The try block lets you test a block of code for errors. The except block lets you handle the error.

2. What is the syntax for a basic try-except block?

ANSWER:

```
try:  
    print(x)  
except:  
    print("An exception occurred")
```

3. What happens if an exception occurs inside a try block and there is no matching except block?

ANSWER: If any exception occurs, the try clause will be skipped and except clause will run. If any exception occurs, but the except clause within the code doesn't handle it, it is passed on to the outer try statements. If the exception is left unhandled, then the execution stops.

4. What is the difference between using a bare except block and specifying a specific exception type?

ANSWER: Bare except block: It becomes difficult to analyze and debug the error.

Specific exception type: It becomes easy to analyze and debug the error.

5. Can you have nested try-except blocks in Python? If yes, then give an example.

ANSWER: We can have nested try-except blocks in Python. In this case, if an exception is raised in the nested try block, the nested except block is used to handle it. In case the nested except is not able to handle it, the outer except blocks are used to handle the exception.

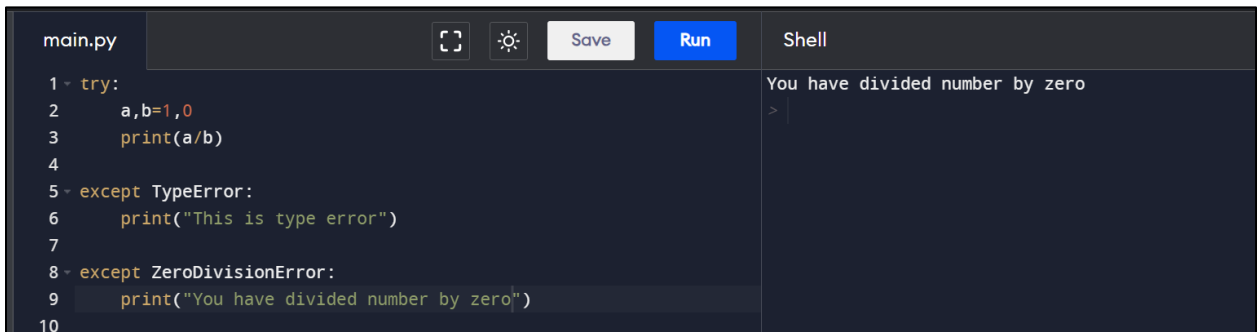
```
try:
    print("outer try block")
    print(10/0)
    try:
        print("Inner try block")
    except ZeroDivisionError:
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")
finally:
    print("outer finally block")
```

Output:

```
outer try block
outer except block
outer finally block
```

6. Can we use multiple exception blocks, if yes then give an example.

ANSWER: It is possible to have multiple except blocks for one try block.



The screenshot shows a code editor with a file named 'main.py'. The code contains a try block with a division by zero, followed by two except blocks: one for 'TypeError' and one for 'ZeroDivisionError'. The 'ZeroDivisionError' block prints a message. The 'Run' button is highlighted in blue. To the right, the 'Shell' window shows the output: 'You have divided number by zero'.

```
main.py  [Icons]  Save  Run  Shell
1 try:
2     a,b=1,0
3     print(a/b)
4
5 except TypeError:
6     print("This is type error")
7
8 except ZeroDivisionError:
9     print("You have divided number by zero")
10
```

You have divided number by zero
>

7. Write the reason due to which following errors are raised:

ANSWER: a) EOFError: EOFError is short for "End-of-Line Error." This error occurs when Python has reached the end of user input without receiving any input. The reason that EOFError occurs is that Python attempts to print out your input in variable string when no data is given.

b) FloatingPointError: Python FloatingPointError is an error that occurs when a floating-point calculation is attempted that is not supported by the available hardware or software. It occurs when trying to calculate a number that is too large or too small to represent as a floating-point number.

c) **IndexError:** The IndexError: list index out of range error occurs in Python when an item from a list is attempted to be accessed that is outside the index range of the list.

d) **MemoryError:** A MemoryError means that the interpreter has run out of memory to allocate to your Python program. This may be due to an issue in the setup of the Python

environment or it may be a concern with the code itself loading too much data at the same time.

e) **OverflowError:** Based on the fact that Python OverflowError occurs when a calculation exceeds the maximum allowed value for a numeric type, someone can face this Python error when an attempt is made to convert a number to a type that is too small to contain it, such as when attempting to convert a large integer to a float.



f) **TabError:** The Python error “TabError: inconsistent use of tabs and spaces in indentation” occurs when you mix tabs and spaces to indent lines in a code block.

g) **ValueError:** ValueError in Python is raised when a user gives an invalid value to a function but is of a valid argument. It usually occurs in mathematical operations that will require a certain kind of value, even when the value is the correct argument.



8. Write code for the following given scenario and add try-exception block to it.

- a. Program to divide two numbers
- b. Program to convert a string to an integer
- c. Program to access an element in a list
- d. Program to handle a specific exception
- e. Program to handle any exception

ANSWER: a) Program to divide two numbers:

| | | | | |
|---|---|------|-----|---|
| main.py |   | Save | Run | Shell |
| <pre>1 try: 2 a,b=1,0 3 print(a/b) 4 5 except ZeroDivisionError: 6 print("You have divided number by zero") 7</pre> | | | | <pre>You have divided number by zero ></pre> |

b) Program to convert a string to an integer:

| | | | | |
|--|---|------|-----|---|
| main.py |   | Save | Run | Shell |
| <pre>1 string = "abcd" 2 try: 3 string_int = int(string) 4 print(string_int) 5 except ValueError: 6 print('Please enter an integer') 7</pre> | | | | <pre>Please enter an integer ></pre> |

C) Program to access an element in a list:

| main.py | Shell |
|--|-------------------------------------|
| <pre>1 list1 = [1,2,3,4,5] 2 try: 3 if 3 in list1: 4 print("element in the list") 5 except: 6 print('Please enter an integer') 7</pre> | <pre>element in the list ></pre> |

d) Program to handle a specific exception

| main.py | Shell |
|---|---|
| <pre>1 a,b=1,0 2 try: 3 print(a/b) 4 except ZeroDivisionError: 5 print("Yoe are dividing number by zero") 6 finally: 7 print("exception is executed") 8</pre> | <pre>Yoe are dividing number by zero exception is executed ></pre> |

e) Program to handle any exception:

| main.py | Shell |
|--|----------------------------------|
| <pre>1 list1=[1,2,3] 2 try: 3 print(x) 4 except: 5 print("x is not defined") 6 7</pre> | <pre>x is not defined ></pre> |