Exception Handling

|  |  |
| --- | --- |
| numerator = 10  denominator = 0  print(numerator/denominator)  Even though our code was correct syntax wise, it's not allowed to divide a number by 0 in Python. This is an exception. In this case, we are getting the ZeroDivisionError exception. | Traceback (most recent call last):  File "<string>", line 1, in <module>  ZeroDivisionError: division by zero |

|  |  |
| --- | --- |
| When our program encounters an exception, our code ends abruptly with an error message. Most of the time, rather than showing this default message, we may want to show a custom message that's more helpful or run a different set of code.  This is known as exception handling. It's the process of responding to exceptions in a custom way during the execution of a program.  In Python, we use the try..except block to handle exceptions. Its syntax is:  try:  # code that may cause exception  except:  # code to run when exception occurs  Inside the try block, we write the code that may raise an exception.  If an exception occurs, the control of the program jumps to the except block.  If exceptions don't occur, the except block is skipped.  try:  numerator = int(input("Enter numerator: "))  denominator = int(input("Enter denominator: "))  result = numerator/denominator  print(result)  except:  print("Denominator cannot be 0. Try again.")  print("Program ends") | Enter numerator: 3  Enter denominator: 0  Denominator cannot be 0. Try again.  Program ends |
| **Handling Specific Exception**   the except block handles all types of exception. If we want to make it handle only certain type of exception we can:  try:  numerator = int(input("Enter numerator: "))  denominator = int(input("Enter denominator: "))  result = numerator/denominator  print(result)  except ZeroDivisionError:  print("Denominator cannot be 0. Try again.")  print("Program ends") | Enter numerator: 3  Enter denominator: 0  Denominator cannot be 0. Try again.  Program ends |
| This is mostly useful if the code inside try block can raise more than one exception.  For example:  try:  numerator = int(input("Enter numerator: "))  denominator = int(input("Enter denominator: "))  result = numerator/denominator  print(result)    my\_list = [1, 2, 3]  i = int(input("Enter index: "))  print(my\_list[i])  except ZeroDivisionError:  print("Denominator cannot be 0. Try again.")  except IndexError:  print("Index is wrong.")  print("Program ends") | Enter numerator: 6  Enter denominator: 5  1.2  Enter index: 4  Index is wrong.  Program ends |
| **Python try...finally**  A try statement can also have an optional finally block which is executed regardless of whether an exception occurs or not.  Its syntax looks something like this:  try:  print(1/0)  except:  print("Wrong denominator")  finally:  print("Always printed")  The code gives the ZeroDivisionError so the except block is executed. Finally, the code inside the finally block is also executed.  However, if an exception doesn't occur in the try block, the except block is skipped but finally block is still executed at the end. | Wrong denominator  Always printed |
| # import module sys to get the type of exception  import sys  randomList = ['a', 0, 2]  for entry in randomList:  try:  print("The entry is", entry)  r = 1/int(entry)  break  except:  print("Oops!", sys.exc\_info()[0], "occurred.")  print("Next entry.")  print()  print("The reciprocal of", entry, "is", r) | The entry is a  Oops! <class 'ValueError'> occurred.  Next entry.  The entry is 0  Oops! <class 'ZeroDivisionError'> occured.  Next entry.  The entry is 2  The reciprocal of 2 is 0.5 |
| **Raising Exception**  The raise statement allows the programmer to force a specific exception to occur. The sole argument in raise indicates the exception to be raised. This must be either an exception instance or an exception class (a class that derives from Exception).  try:  raise NameError("Hi there") # Raise Error  except NameError:  print ("An exception")  raise # To determine whether the exception was raised or not | An exception |
| 1. WAP to demonstrate usage of one try block with multiple except block and else clause. |  |
| 1. WAP to demonstrate usage of Python debugging module. |  |