**Exception Handling in Python**

The cause of an exception is often external to the program itself. For example, an incorrect input, a malfunctioning IO device etc. Because the program abruptly terminates on encountering an exception, it may cause damage to system resources, such as files. Hence, the exceptions should be properly handled so that an abrupt termination of the program is prevented. Python uses try and except keywords to handle exceptions. Both keywords are followed by indented blocks.

Syntax:

try :

#statements in try block

except :

#executed when error in try block

The try: block contains one or more statements which are likely to encounter an exception. If the statements in this block are executed without an exception, the subsequent except: block is skipped.

If the exception does occur, the program flow is transferred to the except: block. The statements in the except: block are meant to handle the cause of the exception appropriately. For example, returning an appropriate error message. You can specify the type of exception after the except keyword. The subsequent block will be executed only if the specified exception occurs. There may be multiple except clauses with different exception types in a single try block. If the type of exception doesn't match any of the except blocks, it will remain unhandled and the program will terminate.

Example:

try:...except blocks

Copy

try: a=5

b='0'

print(a/b)

except:

print('Some error occurred.')

print("Out of try except blocks.")

Output : Some error occurred. **(END OF ANSWER)**

**The finally block consists of statements which should be processed regardless of an exception occurring in the try block or not. As a consequence, the error-free try block skips the except clause and enters the finally block before going on to execute the rest of the code.**

**Raise an Exception**

Python also provides the raise keyword to be used in the context of exception handling. It

causes an exception to be generated explicitly. Built-in errors are raised implicitly. However, a built-in or custom exception can be forced during execution.

The following code accepts a number from the user. The try block raises a ValueError exception,if the number is outside the allowed range.

Example: Raise an Exception

Copy

try:

x=int(input('Enter a number upto 100: '))

if x > 100:

raise ValueError(x)

except ValueError:

print(x, "is out of allowed range")

else:

print(x, "is within the allowed range")

Output

Enter a number upto 100: 200

200 is out of allowed range

Enter a number upto 100: 50

50 is within the allowed range

**Python And Mysql**

 Working with MYSQL Database:-

Procedure To Follow In Python To Work With MySQL

1. Connect to the database.

2. Create an object for your database.

3. Execute the SQL query.

4. Fetch records from the result.

5. Informing the Database if you make any changes in the table.

**1**. Installing MySQL **:-**

MySQL is one of the most popular databases.

Download and install MySQL from the MySQL's official website. You need to install

the MySQL server to follow this tutorial.

Next, you have to install mysql.connector for Python. We need mysql.connector to

connect Python Script to the MySQL database. Download the mysql.connector from it’s official website and install it on your computer.

import mysql.connector

**2**. Connecting And **Creating**

Now, we will connect to the database using username and password of MySQL. If you don't

remember your username or password, create a new user with a password.

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

user = "root",

passwd = "dbms")

print(db)

**Updating Rows from a table:-**

 To update data in a MySQL table in Python, you follow the steps below:

 Connect to the database by creating a new MySQLConnection object.

 Create a new MySQLCursor object from the MySQLConnection object and call

the execute()method of the MySQLCursor object. To accept the changes, you call

the commit() method of the MySQLConnection object after calling the execute() method.

Otherwise, no changes will be made to the database.

 Close the cursor and database connection.

Example

Overwrite the address column from "Valley 345" to "Canyoun 123":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

passwd="yourpassword",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")

**Deleting Rows from A Table:-**

 To delete rows in a MySQL table from Python, you need to do the following steps:

 Connect to the database by creating a new MySQLConnection object.

 Instantiate a new cursor object and call its execute() method. To commit the changes,

you should always call the commit() method of the MySQLConnection object after

calling the execute()method.

 Close the cursor and database connection by calling close() method of the

corresponding objects.

**Example**

Delete any record where the address is "Mountain 21":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

passwd="yourpassword",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")**Creating Database Tables through Python:-**

 To create a table in MySQL, use the "CREATE TABLE" statement.

 Make sure you define the name of the database when you create the connection

Example

 Create a table named "customers":

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

passwd="yourpassword",

database="mydatabase"

)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255), address VARCHAR(255))")