1. In MATLAB, there is a handy function called reshape which can reshape an m x n matrix into a new one with a different size r x c keeping its original data. You are given an m x n matrix mat and two integers r and c representing the number of rows and the number of columns of the wanted reshaped matrix. The reshaped matrix should be filled with all the elements of the original matrix in the same row-traversing order as they were. If the reshape operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

Example 1:

A picture containing application

Description automatically generated

Input: mat = [[1,2], [3,4]], r = 1, c = 4

Output: [[1,2,3,4]]

Example 2:

A picture containing text, clock

Description automatically generated

Input: mat = [[1,2], [3,4]], r = 2, c = 2

Output: [[1,2], [3,4]]

def reshape (mat, r, c):

return newshape

1. Create a stack to take in stack of numbers and then simulate a ring game. A ring stand is such that only a ring of higher diameter can be placed on lower one. The user gives the diameters, the program would compare the diameter of ring at stack top with the diameter of ring to be placed if condition specified is true ring is added to the stack otherwise keep popping and put them into temporary ring stand to arrange them into specific order. You are required to simulate a ring game by implementing and maintaining a stack of numbers. A number represents the diameter of a ring, and the stack represents a stand to hold the rings. Keep on reading numbers(diameters) from user and add in a stack as per the given criteria.
2. Create a program to take in two strings and pass them to a stack object. The strings are made in the following manner. They contain alphanumeric characters and # and # here signifies backspace.

Example – string1= ab#c string2=ad#c

Perform push () operations to put the strings into stack objects.

Perform pop () operations and whenever # is encountered pop out the next character also. (#means backspace)

For example:

ab#c when pushed and popped becomes ac.

ad#c when pushed and popped becomes ac.

Finally check if both the strings are equal or not.

Test cases:

AreEqual(“ab#c”,”ad#c”) should return True as both strings become ‘ac’.

AreEqual(“ab##”,”c#d#”) should return True as both strings become ‘’ after backspace handling.

AreEqual(‘a#c’,’b’) should return False.

**MySQL- Python connectivity**

1. Consider the table “ITEM” having the following fields: Itemcode varchar, Itemname varchar, Price float.

i) Create the table ITEM in the mydb database.

ii) Create a menu driven program in python to have:

a) function for inserting records in the table.

b) function for displaying all the records from the table item.

c) Function for searching for a particular record based on Itemcode.

1. Create a Table “STUDENT” in MySQL with the following attributes.

(i) Write a menu driven program in Python for the user to enter the details and save the data in MySQL table

(ii) Allow the user to update the details for a particular rollno and ensure the changes have been made in the table student.

|  |  |  |  |
| --- | --- | --- | --- |
| ColumnName | Datatype | Size | Constraint |
| RollNo | Number |  | Primary Key |
| Name | Varchar | 30 | Not Null |
| Class | Number |  |  |
| DOB | Date |  |  |
| Gender | Varchar | 2 |  |

**STUDENT**

1. Create a Table “BUS” in MySQL with shown structure: BusNo int Primary Key, Origin Varchar, Dest, Varchar, Rate Number, Km Number. Now build a connection with Python to add a new record and display the details in the above table. You may use Tkinter to create the front end.

**QUESTION 11**

* **CODE :-**

def custom\_reshape(mat, r, c):

m,n = len(mat),len(mat[0])

if m\*n == r\*c:

flat\_mat = []

for row in mat:

flat\_mat.extend(row)

reshaped\_mat = [flat\_mat[i:i+c] for i in range(0, len(flat\_mat),c)]

else:

reshaped\_mat = mat

return reshaped\_mat

# Example 1

mat1 = [[1,2],[3,4]]

r1, c1 = 1,4

output1 = custom\_reshape(mat1,r1,c1)

print(output1)

# Example 2

mat2 = [[1,2],[3,4]]

r2, c2 = 2,2

output2 = custom\_reshape(mat2,r2,c2)

print(output2)

# Example 3

mat3 = [[1,2],[3,4],[5,6]]

r3, c3 = 2,3

r4, c4 = 3,2

r5, c5 = 1,6

output3 = custom\_reshape(mat3,r3,c3)

print(output3)

output4 = custom\_reshape(mat3,r4,c4)

print(output4)

output5 = custom\_reshape(mat3,r5,c5)

print(output5)

* **OUTPUT :-**

[[1, 2, 3, 4]]

[[1, 2], [3, 4]]

[[1, 2, 3], [4, 5, 6]]

[[1, 2], [3, 4], [5, 6]]

[[1, 2, 3, 4, 5, 6]]

**QUESTION 12**

* **CODE :-**

a = int(input("Enter a ring diameter to be added into the stack: "))

stack = []

stack.append(a)

n = 'y'

while n=='y':

l = []

a = int(input("Enter more: "))

if a >= stack[-1]:

stack.append(a)

else:

while a < stack[-1]:

l.append(stack[-1])

stack.pop()

if stack==[]:

break

stack.append(a)

for i in l[::1]:

stack.append(i)

n = input("Do you want to add more (y/n)? ")

print(stack[::1])

* **OUTPUT :-**

Enter a ring diameter to be added into the stack: 10

Enter more: 12

Do you want to add more (y/n)? y

Enter more: 8

Do you want to add more (y/n)? y

Enter more: 9

Do you want to add more (y/n)? n

[9, 8, 12, 10]

**QUESTION 13**

* **CODE :-**

def backspaceCompare(s,t):

def build\_stack(string):

stack = []

for char in string:

if char == '#' and stack:

stack.pop()

elif char != '#':

stack.append(char)

return stack

return build\_stack(s) == build\_stack(t)

def areEqual(s1,s2):

return backspaceCompare(s1,s2)

ch = 'y'

while ch == 'y':

s1 = input("Enter String 1: ")

s2 = input("Enter String 2: ")

print(areEqual(s1,s2))

ch = input("Do you want to continue (y/n): ")

* **OUTPUT :-**

Enter String 1: abc##dc

Enter String 2: agh##dc

True

Do you want to continue (y/n): y

Enter String 1: DUT##VC

Enter String 2: DJH##VC

True

Do you want to continue (y/n): y

Enter String 1: abc#ab

Enter String 2: acs#ab

False

Do you want to continue (y/n): n

**QUESTION 14**

* **CODE :-**

import pymysql as pm

mydb = pm.connect(host='localhost',user='root',passwd='')

print(mydb)

cur = mydb.cursor()

def create\_Database():

cur.execute("CREATE DATABASE AGPRAC")

cur.execute("SHOW DATABASES")

for x in cur:

print(x)

def create\_Table():

cur.execute("USE AGPRAC")

cur.execute("CREATE TABLE ITEM (ITEMCODE VARCHAR(10), ITEMNAME VARCHAR(20), PRICE DECIMAL(8,2))")

cur.execute("SHOW TABLES")

for x in cur:

print(x)

def insertvalues():

cur.execute("USE AGPRAC")

ic = input("Enter Item Code: ")

im = input("Enter Item Name: ")

p = input("Enter Price: ")

values = (ic, im, p)

query = "INSERT INTO ITEM VALUES(%s,%s,%s)"

cur.execute(query, values)

mydb.commit()

cur.execute("SELECT \* FROM ITEM")

for x in cur:

print(x)

def searchcustomer():

cur.execute("USE AGPRAC")

ic = input("Enter Item code: ")

query = "SELECT \* FROM ITEM WHERE ITEMCODE=%s"

cur.execute(query,ic)

for x in cur:

print(x)

while True:

print("1. Create Database")

print("2. Create Table")

print("3. Insert Values")

print("4. Search Customer Id: ")

op = int(input("Enter your choice: "))

if op == 1:

create\_Database()

elif op == 2:

create\_Table()

elif op == 3:

insertvalues()

elif op == 4:

searchcustomer()

else:

print("Program Terminated.")

break

mydb.close()

* **OUTPUT :-**

<pymysql.connections.Connection object at 0x000001D5BC5F11F0>

1. Create Database

2. Create Table

3. Insert Values

4. Search Customer Id:

Enter your choice: 3

Enter Item Code: D80

Enter Item Name: Highlighter

Enter Price: 30

('A21', 'Notebook', Decimal('40.00'))

('B50', 'Pencil', Decimal('10.00'))

('C20', 'Eraser', Decimal('5.00'))

('A90', 'Ruler', Decimal('10.00'))

('D80', 'Highlighter', Decimal('30.00'))

1. Create Database

2. Create Table

3. Insert Values

4. Search Customer Id:

Enter your choice: 4

Enter Item code: C20

('C20', 'Eraser', Decimal('5.00'))

1. Create Database

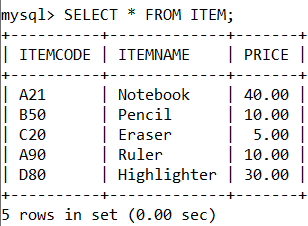
2. Create Table

3. Insert Values

4. Search Customer Id:

Enter your choice: 5

Program Terminated.



**QUESTION 15**

* **CODE :-**

import pymysql as pm

mydb = pm.connect(host='localhost',user='root',passwd='Aakar123')

print(mydb)

mycur = mydb.cursor()

mycur.execute('USE AGPRAC')

mycur.execute('CREATE TABLE IF NOT EXISTS STUDENT(ROLL\_NO INT PRIMARY KEY, NAME VARCHAR(30), CLASS INT, DOB DATE, GENDER VARCHAR(2))')

def insert\_record():

a = int(input("Enter Roll Number: "))

b = input("Enter Name: ")

c = input("Enter Class: ")

d = input("Enter DOB: ")

e = input("Enter Gender: ")

val = (a,b,c,d,e)

query = 'INSERT INTO STUDENT VALUES(%s,%s,%s,%s,%s)'

mycur.execute(query, val)

mydb.commit()

print("Record Added.")

def update\_record():

x = int(input("Enter the roll number where you need to update: "))

a = input("Enter Name: ")

b = input("Enter Class: ")

c = input("Enter DOB: ")

d = input("Enter Gender: ")

val = (a,b,c,d,x)

query = 'UPDATE STUDENT SET NAME=%s, CLASS=%s, DOB=%s, GENDER=%s WHERE ROLL\_NO=%s'

mycur.execute(query, val)

mycur.execute('SELECT \* FROM STUDENT')

print(mycur.fetchall())

mydb.commit()

def menu():

while True:

print("1. Insert Record")

print("2. Update Record")

print("3. Exit")

op = int(input("Enter your choice: "))

if op == 1:

insert\_record()

elif op == 2:

update\_record()

elif op == 3:

print("Program Terminated.")

break

else:

print("Invalid Choice.")

menu()

mydb.close()

* **OUTPUT :-**

<pymysql.connections.Connection object at 0x000002304D2DFB90>

1. Insert Record

2. Update Record

3. Exit

Enter your choice: 1

Enter Roll Number: 3

Enter Name: Aakriti

Enter Class: 12

Enter DOB: 20060313

Enter Gender: F

Record Added.

1. Insert Record

2. Update Record

3. Exit

Enter your choice: 2

Enter the roll number where you need to update: 3

Enter Name: Aakriti

Enter Class: 12

Enter DOB: 20060331

Enter Gender: F

((1, 'Aadi', 12, datetime.date(2006, 12, 12), 'M'), (2, 'Aakar', 12, datetime.date(2006, 4, 7), 'M'), (3, 'Aakriti', 12, datetime.date(2006, 3, 31), 'F'))

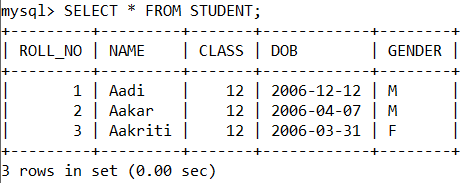
1. Insert Record

2. Update Record

3. Exit

Enter your choice: 3

Program Terminated.



**QUESTION 16**

* **CODE :-**

import pymysql

from tkinter import \*

conn = pymysql.connect(host="localhost", user="root", password="")

print(conn)

cursor = conn.cursor()

cursor.execute("USE AGPRAC")

cursor.execute("CREATE TABLE IF NOT EXISTS BUS (BusNo INT PRIMARY KEY, Origin VARCHAR(255), Dest VARCHAR(255), Rate DECIMAL(10, 2), Km DECIMAL(10, 2))")

def add\_record():

bus\_no = int(bus\_no\_entry.get())

origin = origin\_entry.get()

dest = dest\_entry.get()

rate = float(rate\_entry.get())

km = float(km\_entry.get())

cursor.execute("INSERT INTO BUS VALUES (%s, %s, %s, %s, %s)", (bus\_no, origin, dest, rate, km))

conn.commit()

bus\_no\_entry.delete(0, END)

origin\_entry.delete(0, END)

dest\_entry.delete(0, END)

rate\_entry.delete(0, END)

km\_entry.delete(0, END)

print("Record added successfully.")

root = Tk()

root.title("Bus Details")

Label(root, text="Bus No:", font=('Poppins', 16)).grid(row=0, column=0, pady=5, padx=5)

bus\_no\_entry = Entry(root, font=('Poppins', 16))

bus\_no\_entry.grid(row=0, column=1, pady=5, padx=5)

Label(root, text="Origin:", font=('Poppins', 16)).grid(row=1, column=0, pady=5, padx=5)

origin\_entry = Entry(root, font=('Poppins', 16))

origin\_entry.grid(row=1, column=1, pady=5, padx=5)

Label(root, text="Dest:", font=('Poppins', 16)).grid(row=2, column=0, pady=5, padx=5)dest\_entry = Entry(root, font=('Poppins', 16))

dest\_entry.grid(row=2, column=1, pady=5, padx=5)

Label(root, text="Rate:", font=('Poppins', 16)).grid(row=3, column=0, pady=5, padx=5)

rate\_entry = Entry(root, font=('Poppins', 16))

rate\_entry.grid(row=3, column=1, pady=5, padx=5)

Label(root, text="Km:", font=('Poppins', 16)).grid(row=4, column=0, pady=5, padx=5)

km\_entry = Entry(root, font=('Poppins', 16))

km\_entry.grid(row=4, column=1, pady=5, padx=5)

Button(root, text="Add Record", command=add\_record, font=('Poppins', 16)).grid(row=5, column=0, columnspan=2, pady=10)

root.mainloop()

cursor.close()

conn.close()

* **OUTPUT :-**

<pymysql.connections.Connection object at 0x0000027BFF25FA10>

Record added successfully.

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated