|  |  |  |
| --- | --- | --- |
| **Ex.No.:01** | **BINARY SEARCH** | **Date:** |

**AIM:**

To write a python program to search a list using Binary Search.

**ALGORITHM**:

1. Get the list of elements as input in ascending order
2. Get the element to be searched for
3. Start with the middle element:
   * If the **target** value is equal to the middle element of the array, then return the index of the middle element.
   * If not, then compare the middle element with the target value,
     + If the target value is greater than the number in the middle index, then pick the elements to the right of the middle index, and start with Step 1.
     + If the target value is less than the number in the middle index, then pick the elements to the left of the middle index, and start with Step 1.
4. When a match is found, return the index of the element matched.
5. If no match is found, then return -1

**PROGRAM:**

**#Program to find the element using Binary Search Method**

**def BinarySearch(list\_no, item):**

**l=0**

**u=len(list\_no)-1**

**while (l<=u):**

**mid=(l+u)//2**

**if (list\_no[mid]==item):**

**return mid**

**elif (list\_no[mid]<item):**

**l=mid+1**

**else:**

**u=mid-1**

**return -1**

**#\_\_main\_\_**

**list\_no=eval(input("Enter the List : "))**

**item = int(input("Enter the no to search for : "))**

**pos=BinarySearch(list\_no,item)**

**if (pos == -1):**

**print("No not in the list")**

**else:**

**print("No found at index : ",pos)**

**SAMPLE OUTPUT:**

**Enter the List : [12,23,34,45,56,78,89,90]**

**Enter the no to search for : 89**

**No found at index : 6**

**RESULT :**

Thus the python program to search a list using Binary search mechanism was executed

successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:02** | **BUBBLE SORT** | **Date:** |

**AIM:**

To write a python program to sort a list using Bubble Sort (Ascending order).

**ALGORITHM**:

1. Read a list of list of elements as input from the user
2. Starting with the first element(index = 0), compare the current element with the next element of the array.
3. If the current element is greater than the next element of the array, swap them.
4. If the current element is less than the next element, move to the next element. **Repeat Step 2** no of elements times.
5. Print the sorted list

**PROGRAM:**

**#Program to Sort a list of elements using Bubble Sort**

**def BubbleSort(List):**

**n=len(List)**

**for i in range(n):**

**for j in range(0,n-i-1):**

**if List[j]>List[j+1]:**

**List[j],List[j+1]=List[j+1],List[j]**

**print("List after sorting :",List)**

**#\_\_main\_\_**

**List=eval(input("Enter the list of numbers : "))**

**BubbleSort(List)**

**SAMPLE OUTPUT:**

**Enter the list of numbers : [45,12,56,23,67,14,32,29]**

**List after sorting : [12, 14, 23, 29, 32, 45, 56, 67]**

**RESULT** :

Thus the python program to sort a list using Bubble sort mechanism was sorted

successfully in ascending order and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:03** | **INSERTION SORT** | **Date:** |

**AIM:**

To write a python program to sort a list using Insertion Sort (Ascending order).

**ALGORITHM:**

1. Read a list of list of elements as input from the user
2. Make the second element of the given array, i.e. element at index 1, the key.
3. Compare the key element with the element(s) before it, in this case, element at index 0:
   1. If the key element is less than the first element, we insert the key element before the first element.
   2. If the key element is greater than the first element, then we insert it after the first element.
4. Make the third element of the array as key and will compare it with elements to it's left and insert it at the right position.
5. Repeat the above steps , until the array is sorted.

**PROFRAM**:

#Program to Sort a list of elements using Insertion Sort

def InsertionSort(List):

for i in range(1,len(List)):

key=List[i]

j=i-1

while j>=0 and key<List[j]:

List[j+1]=List[j]

j=j-1

else:

List[j+1]=key

print("List after sorting:",List)

#\_\_main\_\_

List=eval(input("Enter the list of numbers : "))

InsertionSort(List)

**SAMPLE OUTPUT:**

Enter the list of numbers : [90, 34, 56, 12, 78, 23, 87]

List after sorting : [12, 23, 34, 56, 78, 87, 90]

**RESULT**:

Thus the python program to sort a list using Insertion sort mechanism was sorted

successfully in ascending order and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:04** | **READ A TEXT FILE LINE BY LINE AND DISPLAY EACH WORD SEPARATED BY A #** | **Date:** |

**AIM:**

To write a python program to read a text file line by line and display each word in it ,separated by a #

**ALGORITHM:**

1. start the program
2. connect file “file.txt ” using open method
3. using for loop access and store line by line and split using split() method
4. print using nested for word by word ends with “#”
5. end the program

**PROGRAM:**

**#Reading a text file line by line and display each word by separated by a '#'.**

**file = open ('file.txt' , 'r')**

**for line in file:**

**word\_list = line.split()**

**for word in word\_list:**

**print(word , end ='#')**

**file.close()**

**SAMPLE OUTPUT:**

**hai#hello#this#is#your#first#python#class#stay#safe#**

**RESULT:**

Thus the python program read the File and to display each word was displayed separated by ‘#’ was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:05** | **DISPLAY THE COUNT OF DIFFERENT TYPE OF CHARACTERS IN THE FILE** | **Date:** |

**AIM:**

To write a Python program to read a text file and display the number of vowels/ consonants/ uppercase/ lowercase characters in the file.

**ALGORITHM:**

1. Start the program
2. Connect file “sample.txt ” using open method
3. Using read() method collect all data from file
4. Using for loop with isalpha() and isupper() method check count
5. Print the counted vowels, consonants, alphabets
6. Close the file
7. End the program

**PROGRAM:**

**#Read a text file and display the number of #vowels/consonants/uppercase/lowercase characters in the file**

**vowels=['a','e','i','o','u','A','E','I','O','U']**

**v,c,u,l=0,0,0,0**

**file=open('sample.txt','r')**

**content=file.read()**

**for ch in content:**

**if ch.isalpha():**

**if ch in vowels:**

**v=v+1**

**else:**

**c=c+1**

**if ch.isupper():**

**u=u+1**

**else:**

**l=l+1**

**#\_\_main\_\_**

**print('Number of vowels in text file is : ',v)**

**print('Number of consonants in text file is : ',c)**

**print('Number of upper case alphabets in text file is : ',u)**

**print('Number of lower case alphabets in text file is : ',l)**

**SAMPEL OUTPUT:**

**Number of vowels in text file is : 14**

**Number of consonants in text file is : 28**

**Number of upper case alphabets in text file is : 9**

**Number of lower case alphabets in text file is : 33**

**RESULT:**

Thus the Python program to read a text file and display the number of vowels/ consonants/ uppercase/ lowercase characters in the file was executed successfully and output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:06** | **SEARCH AND DISPLAY A RECORD**  **FROM BINARY FILE** | **Date:** |

**AIM :**

To Create a binary file with name and roll number. Search for a given roll number and display the name, if not found display appropriate message.

**ALGORITHM:**

1. Start the program
2. Import pickle module and connect binary file “funfil.txt ” using open method
3. Using while loop get rollno and name and store using dump() method
4. Get the details from file using load() method,
5. Find and print the expected roll no using if statement
6. Close the file
7. End the program

**PROGRAM:**

**#Create a binary file with name and roll number. Search for a given roll number and display the name, if not found display appropriate message**

**import pickle**

**f=open("funfil.dat","wb")**

**n=int(input("Enter the No of Students : "))**

**for i in range(n):**

**rollno=int(input("Enter the Roll no : "))**

**name=input("Enter the Name : ")**

**d={'rollno':rollno,'name':name}**

**pickle.dump(d,f)**

**f.close()**

**#Searching for the Roll no**

**f=open("funfil.dat","rb")**

**r=int(input("Enter the Roll No to be searched : "))**

**Exist=False**

**while True:**

**try:**

**rec=pickle.load(f)**

**if rec['rollno']==r:**

**print("The following Roll No Exists and the Name is",rec['name'])**

**Exist=True**

**except EOFError:**

**break**

**if Exist==False:**

**print("The Roll No does not exist")**

**f.close()**

**SAMPLE OUTPUT:**

**Enter the No of Students : 3**

**Enter the Roll no : 21**

**Enter the Name : Abc**

**Enter the Roll no : 22**

**Enter the Name : Def**

**Enter the Roll no : 23**

**Enter the Name : Hij**

**Enter the Roll No to be searched : 21**

**The following Roll No Exists and the Name is Abc**

**RESULT:**

Thus the Python program to create and search Binary file was created and search process implemented successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:07** | **CREATING UPDATING DATA**  **IN A BINARY FILE** | **Date:** |

**AIM:**

To create a binary file with roll number, name and marks. Input a roll number and update the marks

**ALGORITHM:**

1. start the program
2. import pickle module and connect binary file “info.dat ” using open method
3. using while loop get rollno and name and mark to store using dump() method
4. using for find the specific rollno and mark update the data mark
5. After updating load and print the data using load() method
6. close the file
7. end the program

**PROGRAM:**

**#Creating a binary file with roll number, name and marks. Input a roll number and updating the marks**

**import pickle**

**#Creating a binary file**

**file=open("info.dat","wb")**

**while True:**

**rollno=int(input("Enter Rollno : "))**

**name=input("Enter Name : ")**

**marks=int(input("Enter Marks : "))**

**d={"Rollno":rollno, "Name":name, "Marks":marks}**

**pickle.dump(d,file)**

**choice=int(input("Enter 1 to continue : "))**

**if choice!=1:**

**break**

**file.close()**

**#Reading from the file and making a list of dictionaries**

**file=open("info.dat","rb")**

**l=[]**

**while True:**

**try:**

**d=pickle.load(file)**

**l.append(d)**

**except EOFError:**

**break**

**file.close()**

**#Updating the list of dictionaries**

**print()**

**print("To UPDATE MARK :")**

**print()**

**rno=int(input("Enter the Rollno which is to be Updated : "))**

**mark=int(input("Enter the Mark to be Updated : "))**

**for i in range(len(l)):**

**if l[i]['Rollno']==rno:**

**l[i]['Marks']=mark**

**#Writing the list back into the file**

**file=open("info.dat","wb")**

**for i in range(len(l)):**

**pickle.dump(l[i],file)**

**file.close()**

**#Displaying the updated file**

**print()**

**file=open("info.dat","rb")**

**while True:**

**try:**

**d=pickle.load(file)**

**print(d)**

**except EOFError:**

**break**

**file.close()**

**SAMPLE OUTPUT:**

**Enter Rollno : 1**

**Enter Name : Abc**

**Enter Marks : 73**

**Enter 1 to continue : 1**

**Enter Rollno : 2**

**Enter Name : Def**

**Enter Marks : 96**

**Enter 1 to continue : 1**

**Enter Rollno : 3**

**Enter Name : Ghi**

**Enter Marks : 54**

**Enter 1 to continue : 2**

**To UPDATE MARK :**

**Enter the Rollno which is to be Updated : 1**

**Enter the Mark to be Updated : 83**

**{'Rollno': 1, 'Name': 'Abc', 'Marks': 83}**

**{'Rollno': 2, 'Name': 'Def', 'Marks': 96}**

**{'Rollno': 3, 'Name': 'Ghi', 'Marks': 54}**

**RESULT:**

Thus the Python program for creating Binary file was created and marks was updated for the given roll no successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:08** | **REMOVING LINES IN A TEXT FILE** | **Date:** |

**AIM:**

To write a program to find all the lines that contain the character `a' in a file and write it to another file.

**ALGORITHM:**

1. start the program
2. Connect file “example.txt ” using open function and in read mode
3. using readlines() method collect all data from file
4. using for loop with identity operator in check the presence of “a” in a line and write it in to another file “a.text”
5. close the files
6. end the program

**PROGRAM:**

**#Remove all the lines that contain the character 'a' in a file and write it to another file.**

**file = open("example.txt","r")**

**afile = open("a.txt","w")**

**list\_lines = file.readlines()**

**for line in list\_lines:**

**if 'a' in line:**

**afile.write(line)**

**file.close()**

**afile.close()**

**RESULT:**

Thus the program to find all the lines that contain the character `a' in a file and write it to another file was executed successfully and the output was verifies.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:09** | **RANDOM NUMBERS** | **Date:** |

**AIM:**

To write a Python program to print random numbers

**ALGORITHM:**

1. start the program
2. import random module
3. using while loop access and print random numbers from 1 to 6 and print using random.randint()method
4. Stop the program

**PROGRAM:**

**#A random number generator that generates random numbers between 1 and 6 (simulates a dice)**

**import random as rm**

**print("DICE SIMULATOR\n")**

**while True:**

**resp = input("To generate a random number, press y. To quit enter n.\n\n")**

**if resp == 'y':**

**print("Number =", rm.randint(1, 6), "\n")**

**elif resp == 'n':**

**print("TERMINATING THE PROGRAM")**

**break**

**else:**

**print("!!! INVALID RESPONSE !!!\n")**

**SAMPLE OUTPUT:**

**DICE SIMULATOR**

**To generate a random number, press y. To quit enter n.**

**y**

**Number = 5**

**To generate a random number, press y. To quit enter n.**

**y**

**Number = 6**

**To generate a random number, press y. To quit enter n.**

**y**

**Number = 1**

**To generate a random number, press y. To quit enter n.**

**t**

**!!! INVALID RESPONSE !!!**

**To generate a random number, press y. To quit enter n.**

**n**

**TERMINATING THE PROGRAM**

**RESULT:**

Thus the Python program to generate random numbers was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:10** | **ADD AND REVERSE DIGITS FOR THE GIVEN NUMBER** | **Date:** |

**AIM:**

To write a python program to add and reverse the digits from the given number

**ALGORITHM**:

1. Read the number
2. Initialize reverse, sum to 0
3. Isolate the last digit in number

digit = number % 10

1. Append Digit to reverse

reverse = (reverse \* 10) + digit

Add Digit to sum

Sum = sum + digit

1. Remove last digit from number

number = number / 10

1. Repeat this process until **number** is reduced to zero and **reverse** is completed.

**PROGRAM:**

**# Add and reverse digits for the given number**

**n=int(input("Enter a number"))**

**reverse=0**

**sum\_n=0**

**while (n!=0):**

**digit=n%10**

**sum\_n=sum\_n+digit**

**reverse=(reverse\*10)+digit**

**n=n//10**

**print("Sum of digits of the number is",sum\_n)**

**print("Reversed number:",reverse)**

**SAMPLE OUTPUT:**

**Enter a number576**

**Sum of digits of the number is 18**

**Reversed number: 675**

**RESULT:**

Thus the Python Program to add and reverse the digits of a no was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:11** | **FIBONACCI SERIES** | **Date:** |

**AIM:**

To write a python program to print ‘***n’*** terms from Fibonacci series.

**ALGORITHM:**

1. Start the Program
2. Initialize the variables, a=0, b=1, and c =0
3. Enter the number of terms of Fibonacci series to be printed
4. Print First two terms of series
5. Repeat the following steps for the no of terms times  
   ->c=a+b  
   ->a=b  
   ->b=c  
   ->increase value of i each time by 1  
   -> print the value of show
6. Stop t he Program

**PROGRAM:**

**# Fibonacci series**

**n=int(input("(Enter a number>=3):"))**

**a=0**

**b=1**

**c=0**

**print(a)**

**print(b)**

**for i in range(3,n+1):**

**c=a+b**

**print(c,end=” “)**

**a,b=b,c**

**SAMPLE OUTPUT:**

**Enter a number>=3):7**

**0 1 1 2 3 5 8**

**RESULT:**

Thus the Python program to print ‘***n’*** terms from Fibonacci series was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:12** | **AMSTRONG NUMBER** | **Date:** |

**AIM:**

To write a python program to check whether if the given number is Armstrong or not.

**ALGORITHM:**

1. Start the Program
2. Read num from User
3. Initialize Variable sum=0 and temp=num
4. Repeat Until num>=0

sum=sum + cube of last digit i.e [(num%10)\*(num%10)\*(num%10)]

num=num/10

1. if sum==temp

Print "Armstrong Number"

else

Print "Not Armstrong Number"

1. End the program

**PROGRAM:**

**# Armstrong no**

**num=int(input("Enter a number:"))**

**sum\_n=0**

**temp=num**

**while (num!=0):**

**sum\_n=sum\_n+(num%10)\*\*3**

**num=num//10**

**if sum\_n==temp:**

**print("Armstrong number")**

**break**

**else:**

**print("Not a Armstrong number")**

**SAMPLE OUTPUT:**

**Enter a number:407**

**Armstrong number**

**--------------------------**

**Enter a number:400**

**Not a Armstrong number**

**RESULT**:

Thus the Python program to check if the given number is Armstrong or not was checked successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:13** | **CHECK STRING IS PALINDROME OR NOT** | **Date:** |

**AIM:**

To write a Python Program to read a string and check whether the given string is palindrome or not.

**ALGORITHM:**

1. Start the program
2. Read the string

3. Declare a variable j with the length of the string -1.

4. Declare a for loop, using half of the length of the string as a reference point.

5. Check if each letter is the same as its mirror equivalent — or, a character on the other side

6. If they are not same print It is not a palindrome

7. if the loop exits without satisfying the above condition , print it is a palindrome

8. Stop the Program

**PROGRAM:**

#Program to find if the given string is a palindrome

str = input("Enter a string: ")

j = len(str)-1

for i in range ( len(str)/2 ):

if str[i] != str[j] :

print("Not a Palindrome")

break

j = j-1

else:

print("Palindrome")

**SAMPLE OUTPUT:**

Enter a string: malayalam

Palindrome

--------------------------------

Enter a string: school

Not a Palindrome

**RESULT:**

Thus the Python Program to check whether the given string is a palindrome or not was checked successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:14** | **IMPLEMENTATION OF STACK** | **Date:** |

**AIM: TO WRITE A PYTHON PROGRAM TO IMPLEMENT STACK USING A LIST DATA-STRUCTURE.**

**ALGORITHM - FUNCTION : ISEMPTY(STK):**

**STEP 01: START**

**STEP 02: IF STK == []:**

**STEP 03: RETURN TRUE**

**ELSE**

**STEP 04: RETURN FALSE**

**ALGORITHM - FUNCTION : PUSH(STK,ITEM):**

**STEP 01: START**

**STEP 02: STK.APPEND(ITEM)**

**STEP 03: TOP = LEN(STK)-1**

**STEP 04: STOP**

**ALGORITHM - FUNCTION : POP(STK):**

**STEP 01: START**

**STEP 02: IF ISEMPTY(STK):**

**STEP 03: RETURN "UNDERFLOW"**

**ELSE**

**STEP 05: ITEM = STK.POP ()**

**STEP 06: IF LEN(STK) == 0:**

**STEP 07: TOP = NONE**

**ELSE**

**STEP 08: TOP=LEN(STK)-1**

**STEP 09: RETURN ITEM**

**ALGORITHM - FUNCTION: PEEK (STK):**

**STEP 01: START**

**STEP 02: IF ISEMPTY(STK):**

**STEP 03: RETURN "UNDERFLOW"**

**ELSE**

**STEP 04: TOP=LEN(STK)-1**

**STEP 05: RETURN STK[TOP]**

**ALGORITHM - FUNCTION : DISPLAY(STK):**

**STEP 01: START**

**STEP 02: IF ISEMPTY (STK):**

**STEP 03: PRINT('STACK IS EMPTY')**

**ELSE**

**STEP 04: TOP = LEN (STK)-1**

**STEP 05: PRINT (STK[TOP],'<-TOP')**

**STEP 06: FOR I IN RANGE(TOP-1,-1,-1):**

**STEP 07: PRINT(STK[I])**

**STEP 08: STOP**

**ALGORITHM - DRIVER CODE : MAIN():**

**STEP-1: START.**

**STEP-2: PRINT (“1.PUSH, 2.POP ,**

**3.PEEK , 4.DISPLAY STACK & 5.EXIT” )**

**STEP-3: INPUT CH**

**STEP-4: IF CH=1**

**INPUT ITEM,PUSH ITEM**

**STEP-5: ELIF CH=2**

**POP STACK**

**STEP-6: ELIF CH=3**

**PEEK STACK**

**STEP-7: ELIF CH=4**

**DISPLAY STACK**

**STEP-8: ELIF CH=5**

**STEP-9: STOP**

**PROGRAM:**

**#IMPLIMENTATION OF STACK**

**def isempty(stk):**

**if stk==[]:**

**return True**

**else:**

**return False**

**def push(stk,item):**

**stk.append(item)**

**top=len(stk)-1**

**def pop(stk):**

**if isempty(stk):**

**return "underflow"**

**else:**

**item=stk.pop()**

**if len(stk)==0:**

**top=None**

**else:**

**top=len(stk)-1**

**return item**

**def peek(stk):**

**if isempty(stk):**

**return "underflow"**

**else:**

**top=len(stk)-1**

**return stk[top]**

**def display(stk):**

**if isempty(stk):**

**print('stack is empty')**

**else:**

**top=len(stk)-1**

**print(stk[top],'<-top')**

**for i in range(top-1,-1,-1):**

**print(stk[i])**

**#Driver Code**

**def main():**

**stk=[]**

**top=None**

**while True:**

**print('''stack operation**

**1.push**

**2.pop**

**3.peek**

**4.display**

**5.exit''')**

**choice=int (input('enter choice:'))**

**if choice==1:**

**item=int(input('enter item:'))**

**push(stk,item)**

**elif choice==2:**

**item=pop(stk)**

**if item=="underflow":**

**print('stack is underflow')**

**else:**

**print('poped')**

**elif choice==3:**

**item=peek(stk)**

**if item=="underflow":**

**print('stack is underflow')**

**else:**

**print('top most item is:',item)**

**elif choice==4:**

**display(stk)**

**elif choice==5:**

**break**

**else:**

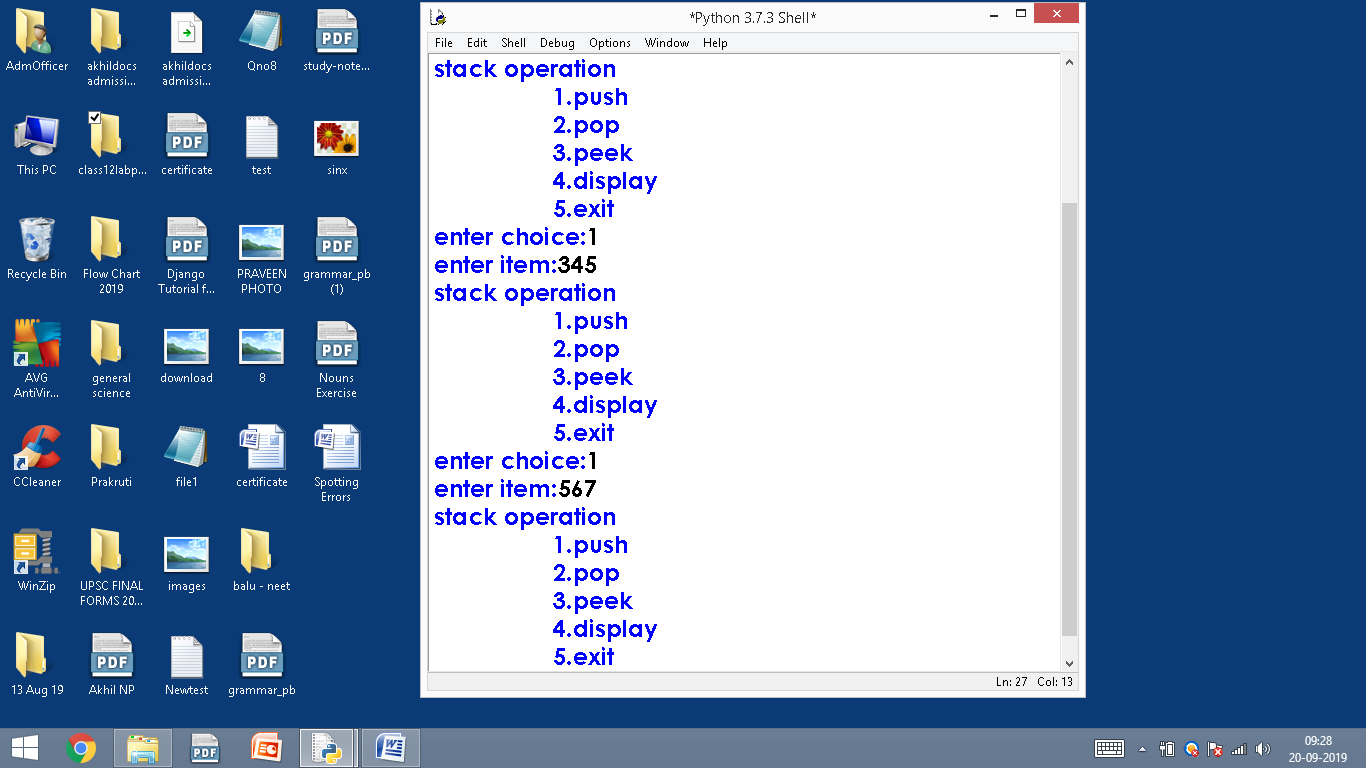
**print('invalid')**

**exit()**

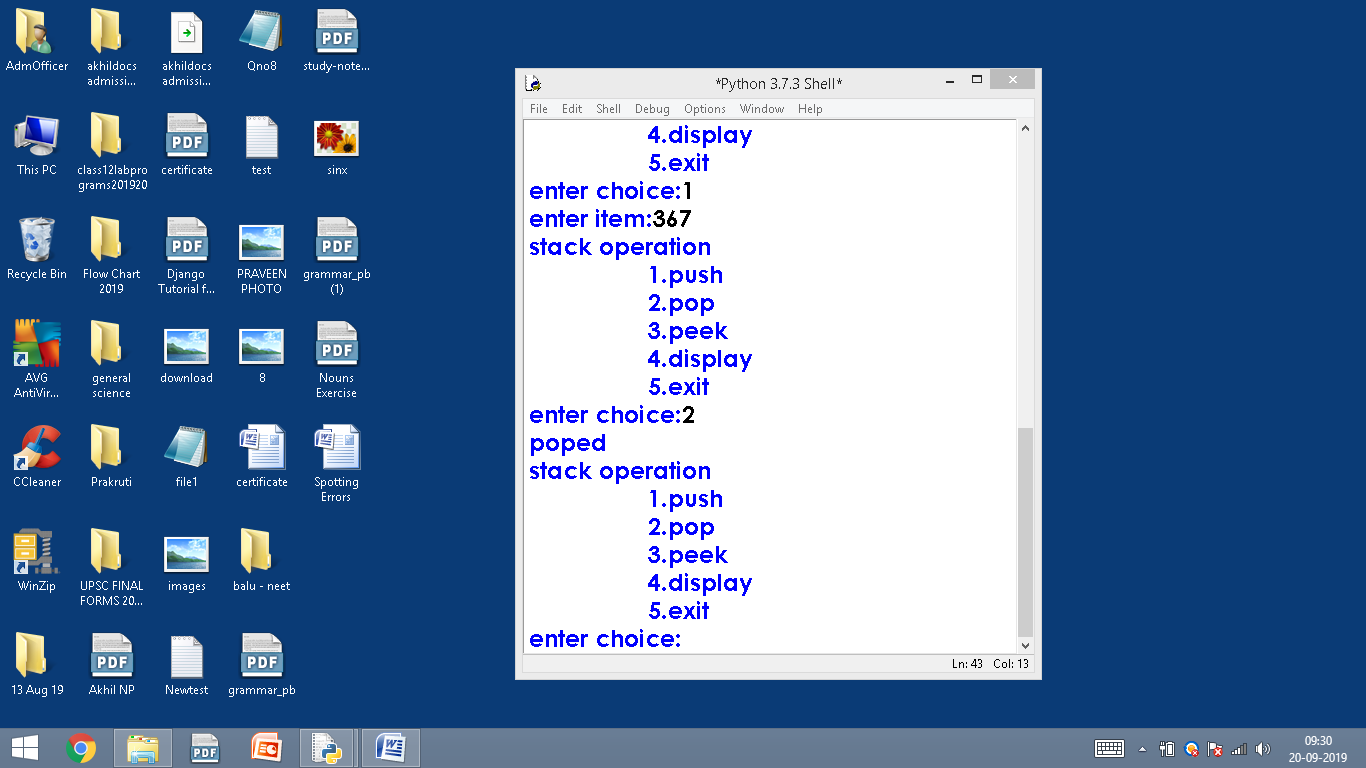
**main()**

**SAMPLE OUTPUT:**

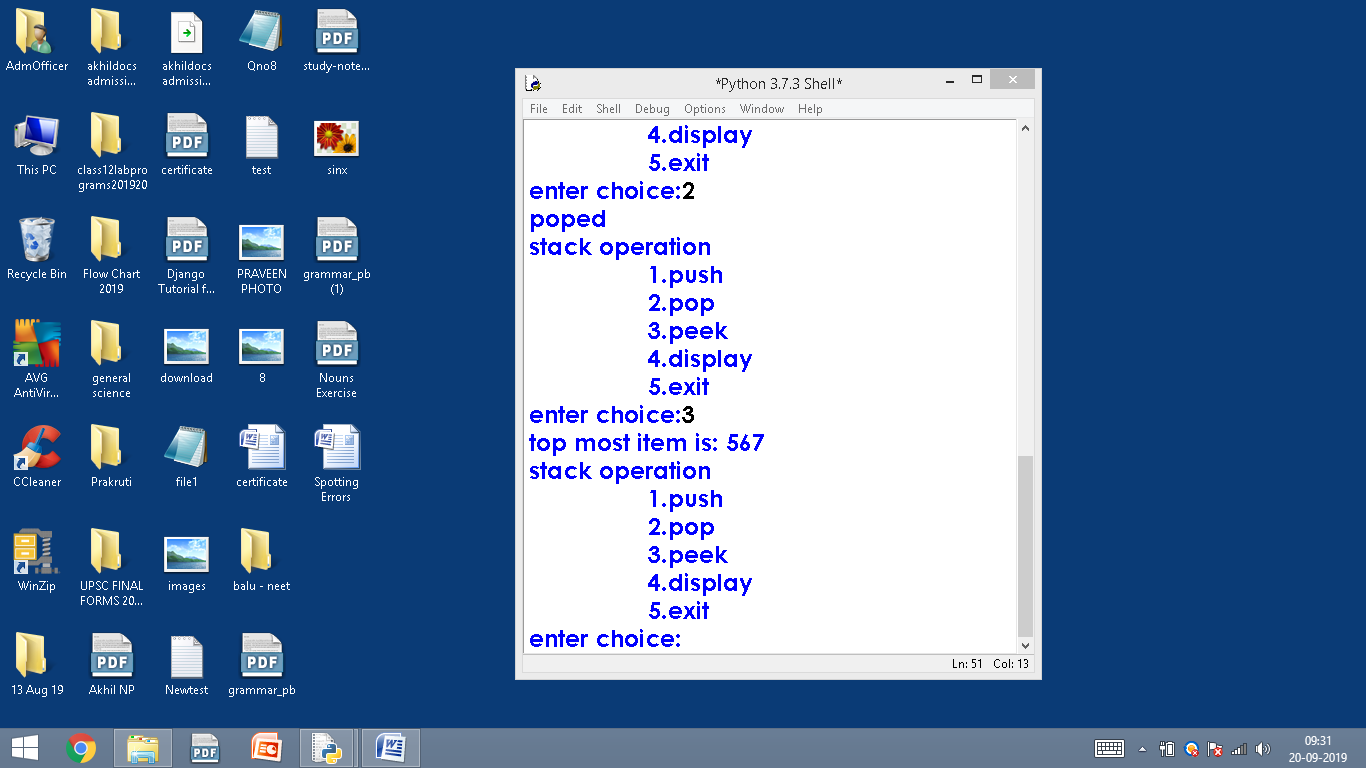
**PUSH OPERATION**

****

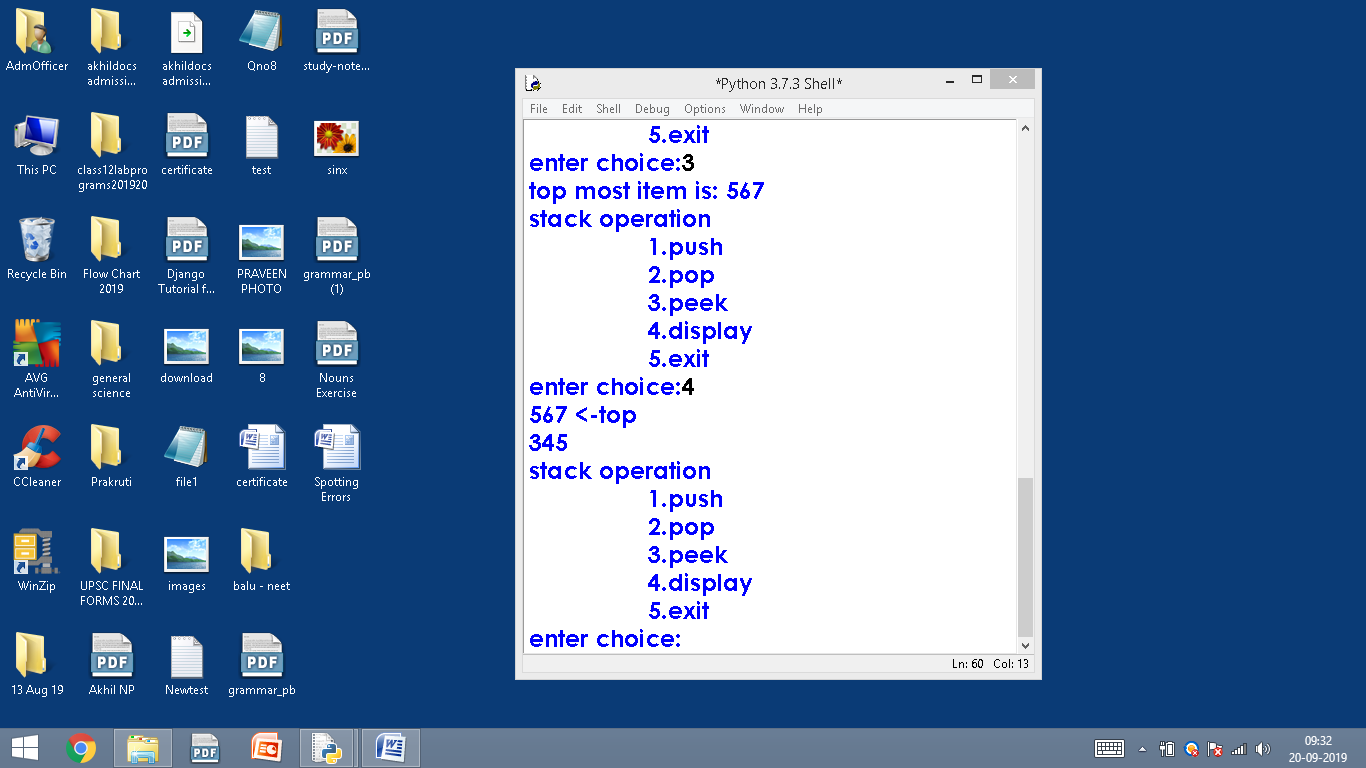
**STACK OUTPUT: POP OPERATION**



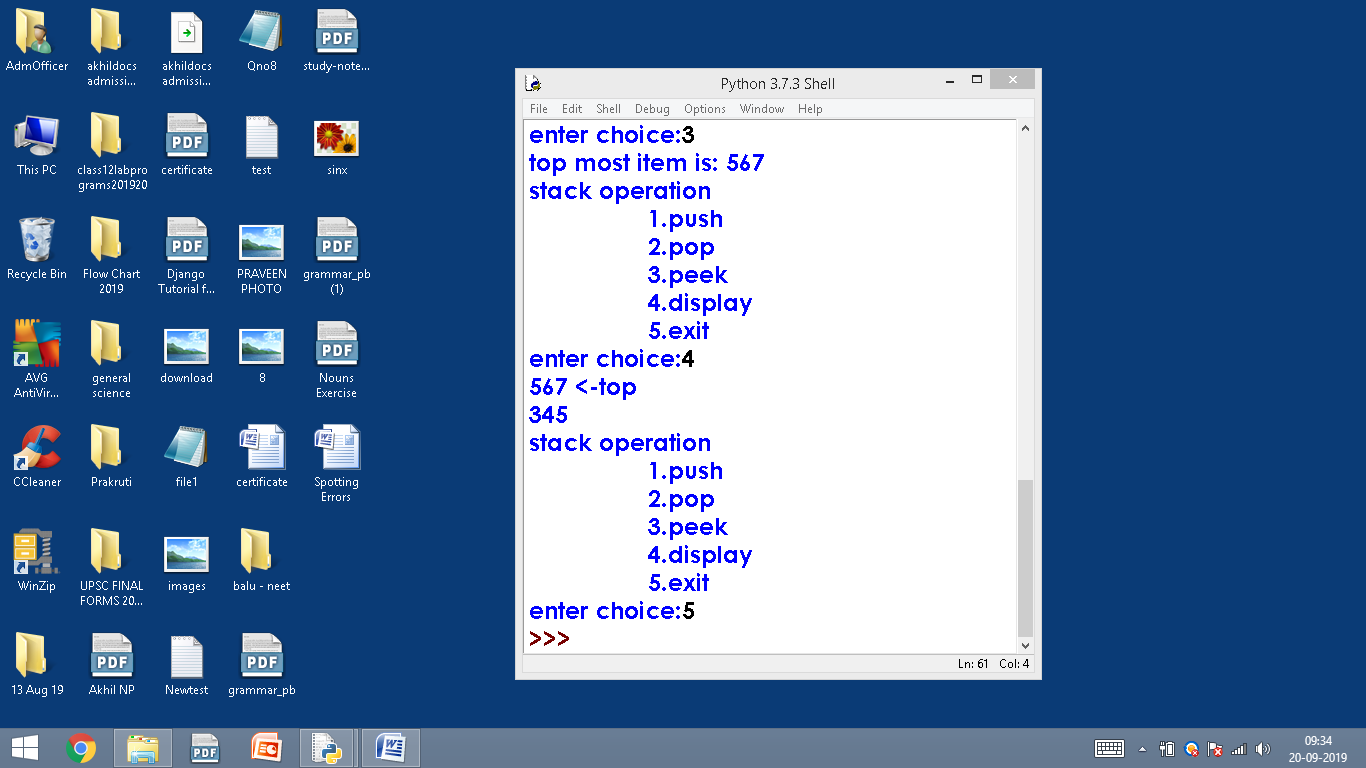
**STACK OUTPUT: PEEK OPERATION**



**STACK OUTPUT: DISPLAY OPERATION**



**STACK OUTPUT: EXIT OPERATION**



**RESULT:**

**THUS THE PROGRAM FOR IMPLEMENTAION OF STACK HAS BEEN EXECUTED SUSSESSFULLY AND THE OUTPUT WAS VERIFIED.**

|  |  |  |
| --- | --- | --- |
| **Ex.No.:15** | **IMPLEMENTATION OF QUEUE** | **Date:** |

**AIM:**

**TO WRITE A PYTHON PROGRAM TO IMPLEMENT QUEUE USING A LIST DATA-STRUCTURE.**

**ALGORITHM - FUNCTION : CLS()**

**STEP 01: START**

**STEP 02: PRINT('\n'\*100)**

**STEP 03: STOP**

**ALGORITHM - FUNCTION : ISEMPTY(QU):**

**STEP 01: START**

**STEP 02: IF QU==[]:**

**STEP 03: RETURN TRUE**

**STEP 04: ELSE:**

**RETURN FALSE**

**ALGORITHM - FUNCTION : ENQUEUE(QU,ITEM):**

**STEP 01: START**

**STEP 02: QU.APPEND(ITEM)**

**STEP 03: IF LEN(QU)==1:**

**STEP 04: FRONT=REAR=0**

**STEP 05: ELSE**

**REAR=LEN(QU)-1**

**STEP 06: STOP**

**ALGORITHM - FUNCTION : DEQUEUE(QU):**

**STEP 01: START**

**STEP 02: IF ISEMPTY(QU):**

**STEP 03: RETURN 'UNDERFLOW'**

**STEP 04: ELSE:**

**ITEM=QU.POP(0)**

**STEP 05: IF LEN(QU)==0:**

**STEP 06: FRONT=REAR=NONE**

**STEP 07: RETURN ITEM**

**ALGORITHM - FUNCTION : PEEK(QU):**

**STEP 01: START**

**STEP 02: IF ISEMPTY(QU):**

**STEP 03: RETURN 'UNDERFLOW'**

**STEP 04: ELSE:**

**FRONT=0**

**STEP 05: RETURN QU[FRONT]**

**ALGORITHM - FUNCTION : DISPLAY(QU):**

**STEP 01: START**

**STEP 02: IF ISEMPTY(QU):**

**STEP 03: PRINT('QUEUE EMPTY')**

**STEP 04: ELIF LEN(QU)==1:**

**PRINT(QU[0],'<==FRONT,REAR')**

**STEP 05: RETURN QU[FRONT]**

**STEP 06: ELSE:**

**STEP 07: FRONT=0**

**STEP 08: REAR=LEN(QU)-1**

**STEP 08: PRINT(QU[FRONT],'<-FRONT')**

**STEP 09: FOR A IN RANGE(1,REAR):**

**STEP 10: PRINT(QU[A])**

**STEP 11: PRINT(QU[REAR],'<-REAR')**

**ALGORITHM: DRIVER CODE : MAIN ( ):**

**STEP 01: START**

**STEP 02: QUEUE=[]**

**STEP 03: WHILE TRUE:**

**STEP 04: PRINT('''QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.PEEK**

**4.DISPLAY QUEUE**

**5.EXIT''')**

**STEP 05: CH=INT(INPUT('ENTER YOUR CHOICE.......[1-5]:'))**

**STEP 06: IF CH==1:**

**STEP 07: ITEM=INT(INPUT('ENTER ITEM:'))**

**STEP 08: ENQUEUE(QUEUE,ITEM)**

**STEP 09: INPUT('PRESS ENTER TO CONTINUE....')**

**STEP 10: ELIF CH==2:**

**STEP 11: ITEM=DEQUEUE(QUEUE)**

**STEP 12: IF ITEM=='UNDERFLOW':**

**STEP 13: PRINT('UNDERFLOW!QUEUE IS EMPTY')**

**STEP 14: ELSE: PRINT('DEQUEUED ITEM IS:',ITEM)**

**STEP 15: INPUT('PRESS ENTER TO CONTINUE....')**

**STEP 16: ELIF CH==3:**

**STEP 17: ITEM=PEEK(QUEUE)**

**STEP 18: IF ITEM=='UNDERFLOW':**

**STEP 19: PRINT('UNDERFLOW!QUEUE IS EMPTY')**

**STEP 20: ELSE: PRINT('FRONTMOST ITEM IS:',ITEM)**

**STEP 21: INPUT('PRESS ENTER TO CONTINUE....')**

**STEP 22: ELIF CH==4:**

**STEP 23: DISPLAY(QUEUE)**

**STEP 24: INPUT('PRESS ENTER TO CONTINUE....'))**

**STEP 25: ELIF CH==5:**

**STEP 26: BREAK**

**STEP 27: ELSE: PRINT('INVALID CHOICE!!!')**

**STEP 28: INPUT('PRESS ENTER TO CONTINUE....')**

**STEP 29: STOP**

**PROGRAM:**

**def cls():**

**print('\n'\*100)**

**def isEmpty(Qu):**

**if Qu==[]:**

**return True**

**else:**

**return False**

**def Enqueue(Qu,item):**

**Qu.append(item)**

**if len(Qu)==1:**

**front=rear=0**

**else:**

**rear=len(Qu)-1**

**def Dequeue(Qu):**

**if isEmpty(Qu):**

**return 'underflow'**

**else:**

**item=Qu.pop(0)**

**if len(Qu)==0:**

**front=rear=None**

**return item**

**def Peek(Qu):**

**if isEmpty(Qu):**

**return 'underflow'**

**else:**

**front=0**

**return Qu[front]**

**def Display(Qu):**

**if isEmpty(Qu):**

**print('queue empty')**

**elif len(Qu)==1:**

**print(Qu[0],'<==front,rear')**

**else:**

**front=0**

**rear=len(Qu)-1**

**print(Qu[front],'<-front')**

**for a in range(1,rear):**

**print(Qu[a])**

**print(Qu[rear],'<-rear')**

**#Driver Code**

**def main():**

**queue=[]**

**while True:**

**print('''QUEUE OPERATIONS**

**1.ENQUEUE**

**2.DEQUEUE**

**3.PEEK**

**4.DISPLAY QUEUE**

**5.EXIT''')**

**ch=int(input('Enter your choice.......[1-5]:'))**

**if ch==1:**

**item=int(input('Enter item:'))**

**Enqueue(queue,item)**

**input('Press Enter to continue....')**

**elif ch==2:**

**item=Dequeue(queue)**

**if item=='underflow':**

**print('Underflow!queue is empty')**

**else:**

**print('Dequeued item is:',item)**

**input('Press Enter to continue....')**

**elif ch==3:**

**item=Peek(queue)**

**if item=='underflow':**

**print('Underflow!queue is empty')**

**else:**

**print('Frontmost item is:',item)**

**input('Press Enter to continue....')**

**elif ch==4:**

**Display(queue)**

**input('Press Enter to continue....')**

**elif ch==5:**

**break**

**else:**

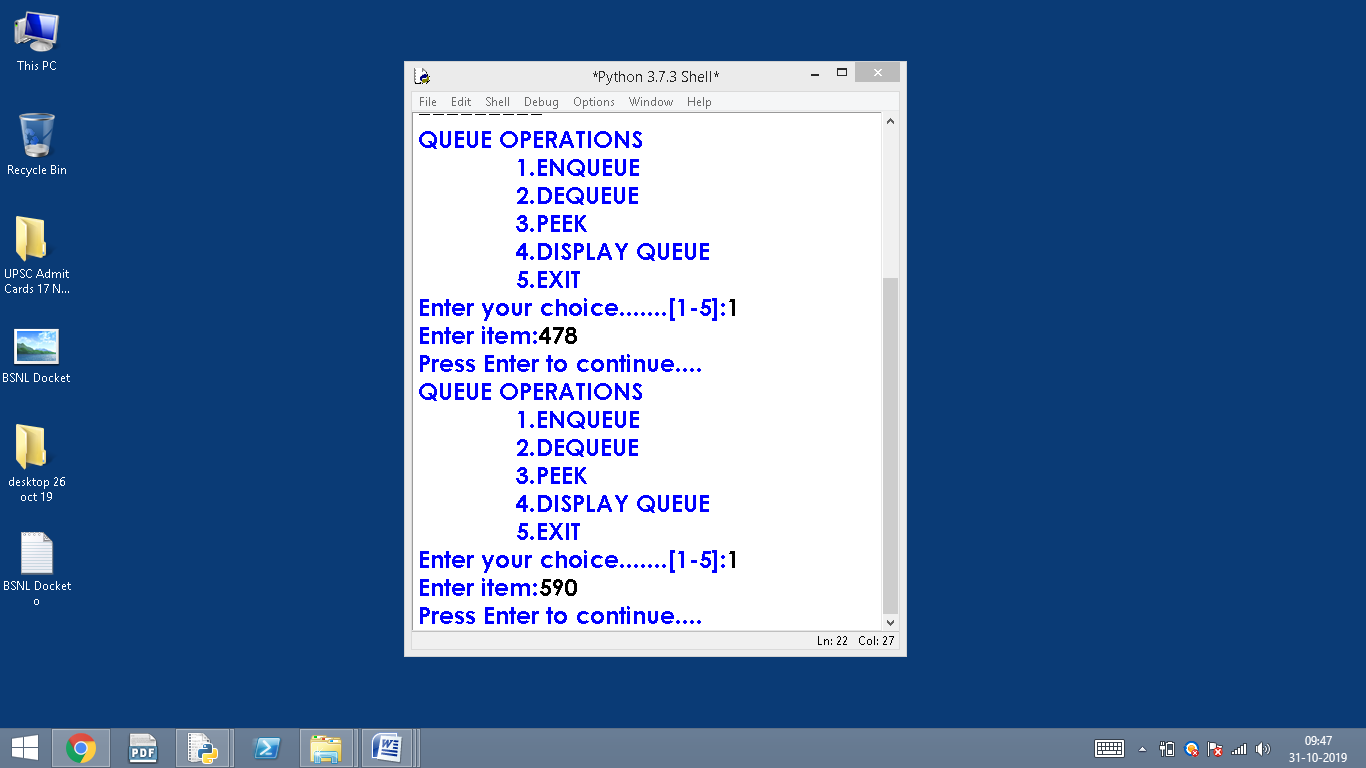
**print('Invalid choice!!!')**

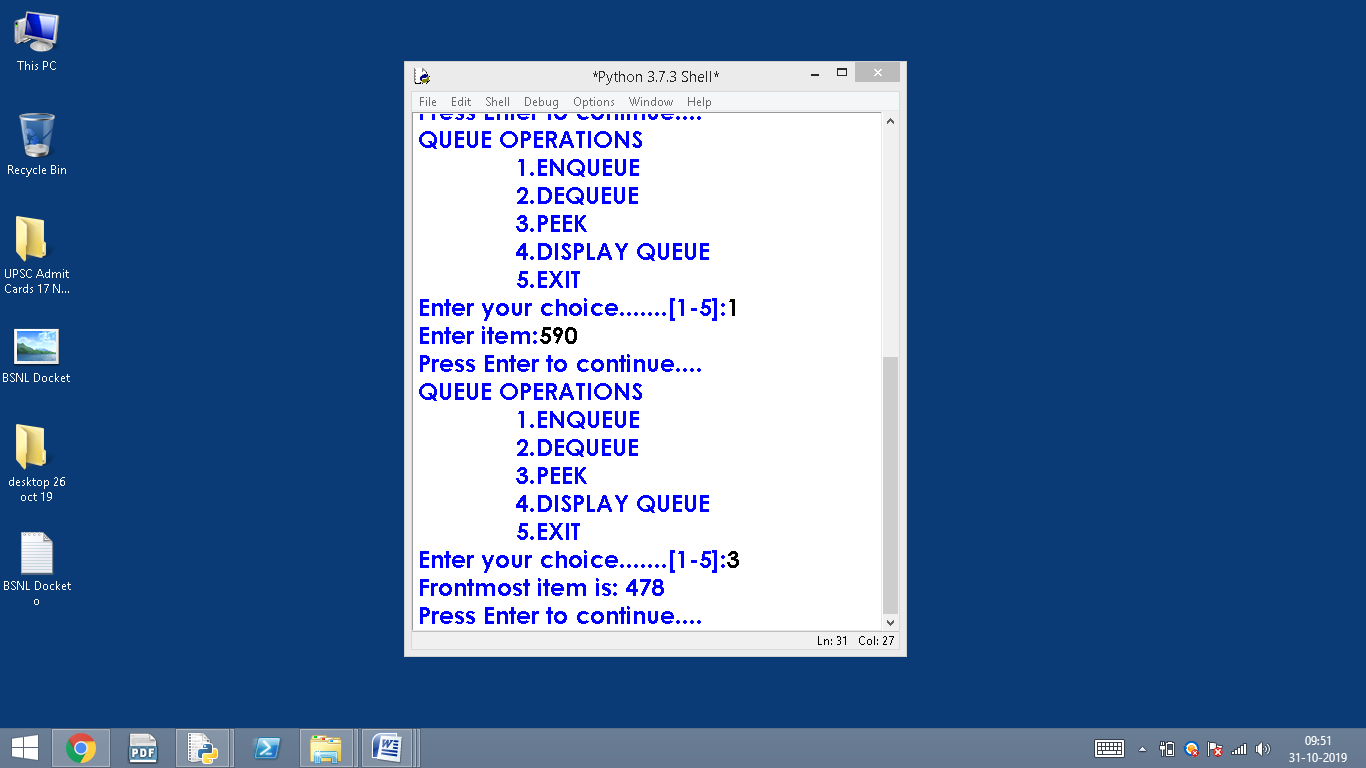
**input('Press Enter to continue....')**

**main()**

**SAMPLE OUTPUT:**

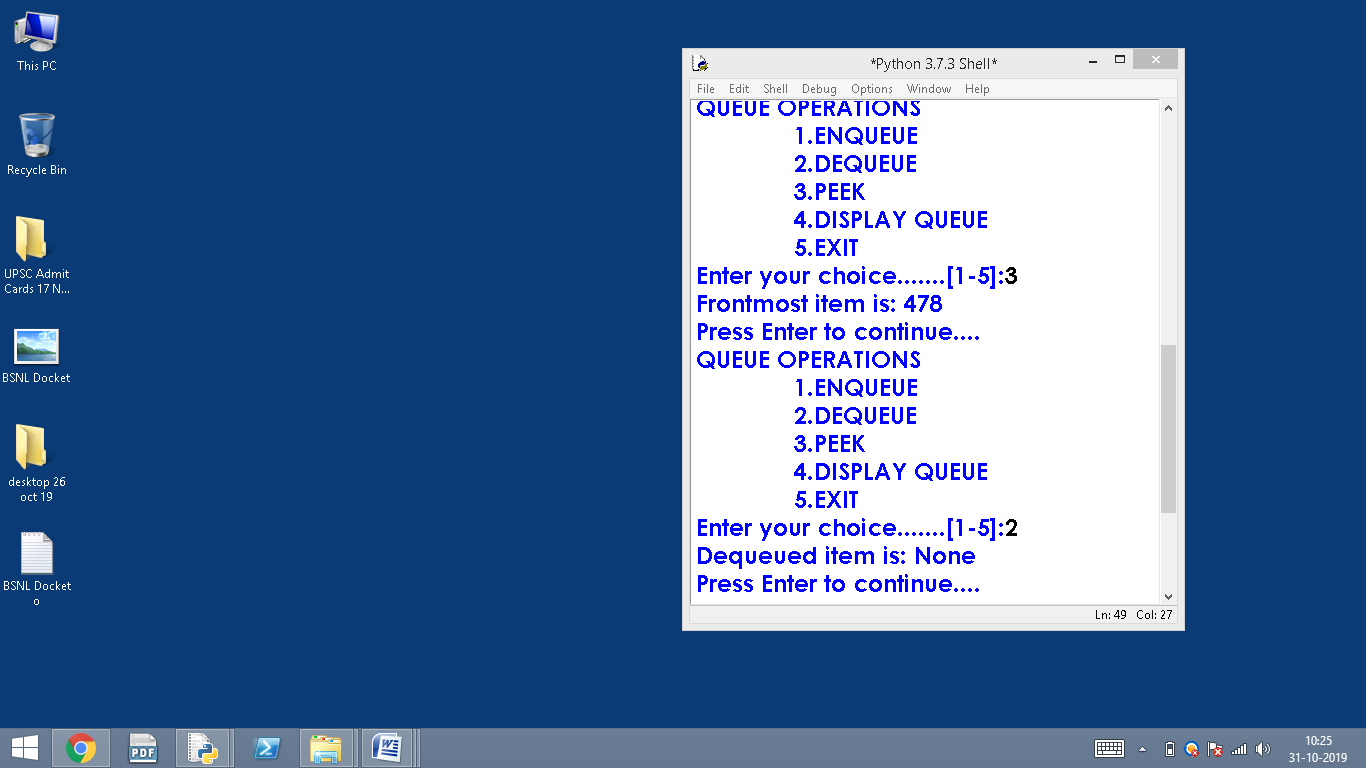
**QUEUE OUTPUT : ENQUEUE OPERATION**

****

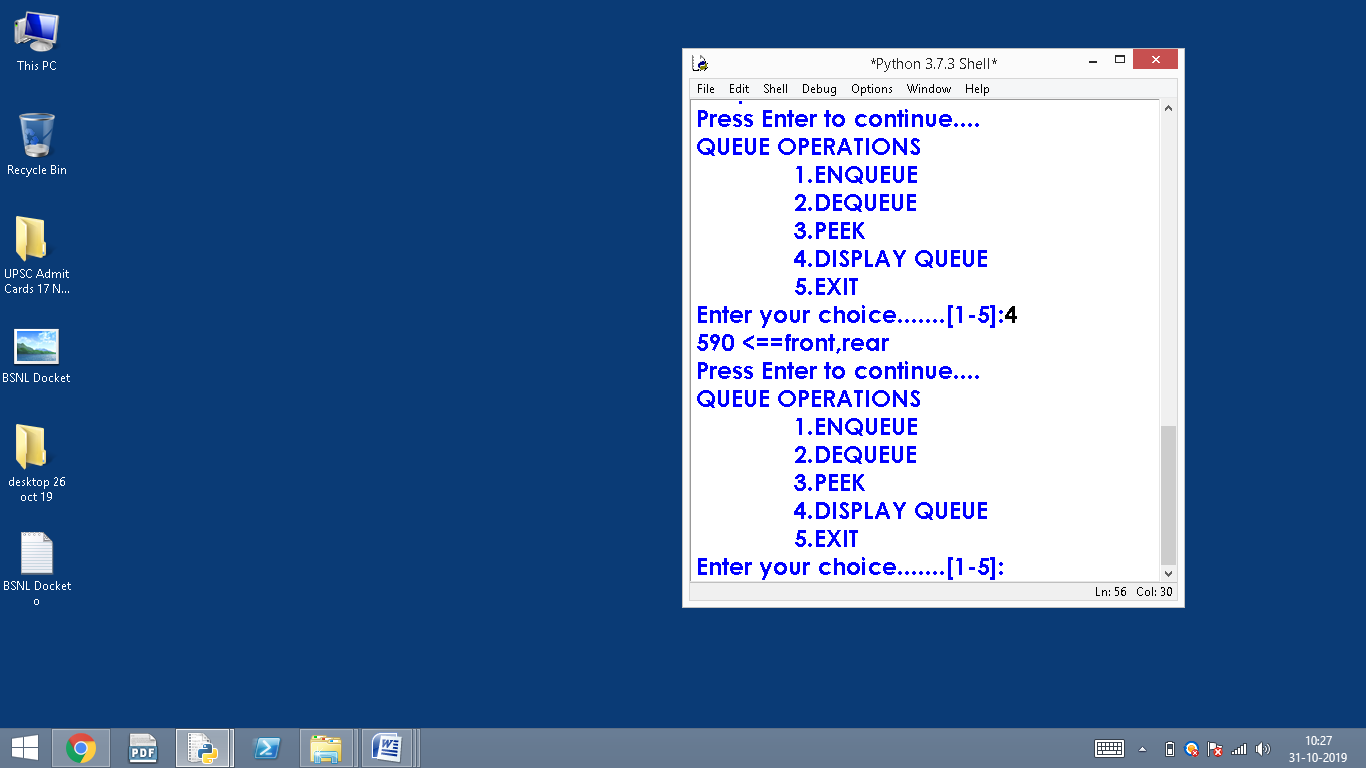
****

**QUEUE OUTPUT : PEEK OPERATION**

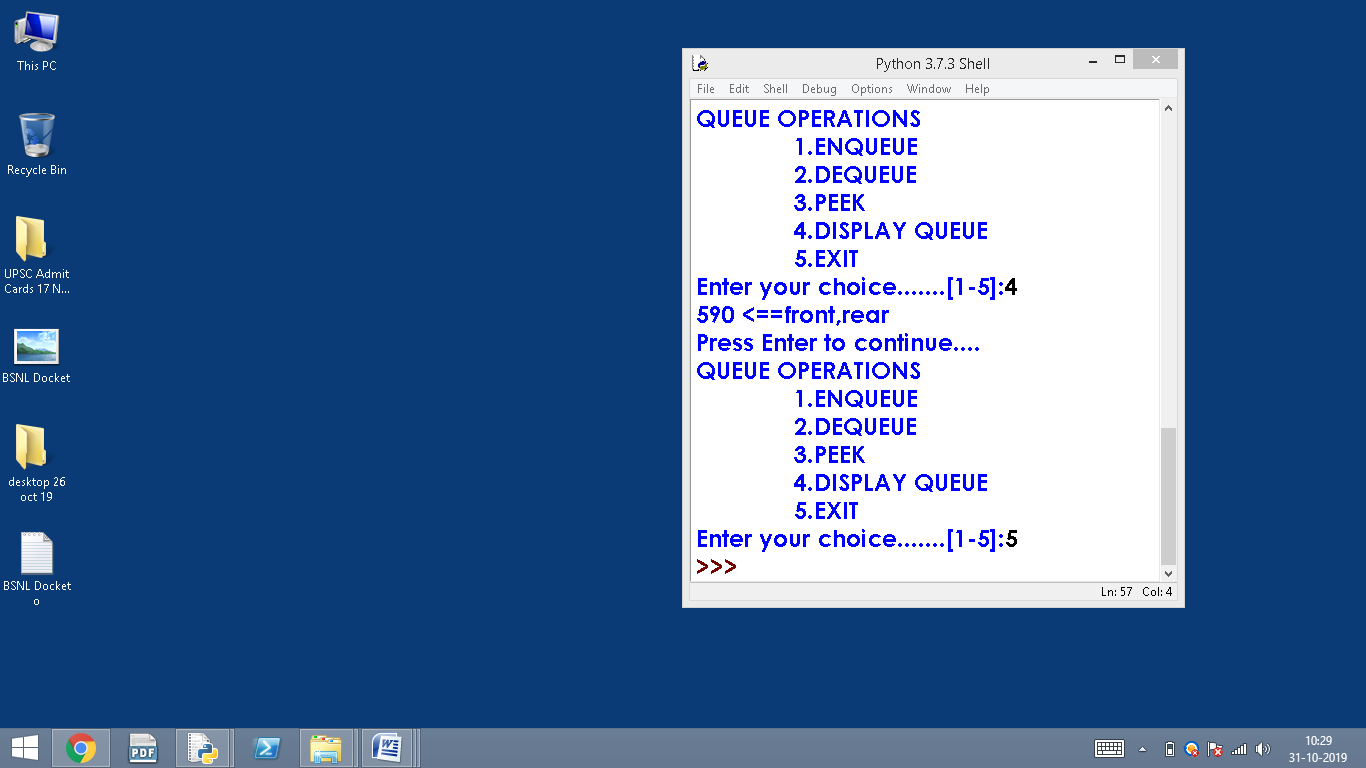
**QUEUE OUTPUT : DEQUEUE OPERATION**



**QUEUE OUTPUT : DEQUEUE OPERATION**



**QUEUE OUTPUT : EXIT OPERATION**



**RESULT:**

**THUS THE PROGRAM FOR IMPLEMENTAION OF QUEUE HAS BEEN EXECUTED SUSSESSFULLY AND THE OUTPUT WAS VERIFIED.**

|  |  |  |
| --- | --- | --- |
| **Ex.No.:16** | **SEARCH AND DISPLAY FROM A *CSV* FILE** | **Date:** |

**AIM :**

To Create a csv file by entering userid and password , then read from the file and display the password for the given userid.

**ALGORITHM:**

1. Start the program
2. Import csv module and open csv file “newcsv.csv” in write mode
3. Get userid and password and write them into the file
4. Close the file
5. Open the file again in read mode
6. Read from the file
7. Process record by record and if the userid matches display the password
8. Close the file
9. Stop the program

**PROGRAM:**

**import csv**

**#creating a csv file**

**f = open("newcsv.csv","w", newline='')**

**writerobj = csv.writer(f)**

**choice = 'y'**

**while choice == 'y':**

**userid = input("Enter userid:")**

**password = input("Enter password:")**

**writerobj.writerow( [userid , password] )**

**choice = input("Enter ‘y’ to continue:")**

**f.close()**

**#reading from a csv file**

**f = open("newcsv.csv","r")**

**readerobj = csv.reader(f)**

**user = input("\nEnter userid to fetch password:")**

**for rec in readerobj:**

**if rec[0] == user:**

**print ("Password : ",rec[1])**

**f.close()**

**SAMPLE OUTPUT:**

**Enter userid:1001**

**Enter password:hello@123**

**Enter ‘y’ to continue:y**

**Enter userid:1002**

**Enter password:hi@456**

**Enter ‘y’ to continue”y**

**Enter userid”1003**

**Enter password:bye@789**

**Enter ‘y’ to continue:n**

**Enter userid to fetch password:1002**

**Password : hi@456**

**RESULT:**

Thus the Python program to create and search in CSV file was created and the data was searched successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:17** | **MYSQL QUERIES - I** | **Date:** |

**AIM:**

To write MySQL commands(queries) for (1) to (10) and write output for (11) to (14) on the basis of ***Teacher*** relation given.

**TABLE: TEACHER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NO.** | **NAME** | **AGE** | **DEPARTMENT** | **DATEOFJOIN** | **SALARY** | **SEX** |
| **1** | **JUGAL** | **34** | **COMPUTER** | **10/01/97** | **12000** | **M** |
| **2** | **SHARMILA** | **31** | **HISTORY** | **24/03/98** | **20000** | **F** |
| **3** | **SANDEEP** | **32** | **MATHS** | **12/12/96** | **30000** | **M** |
| **4** | **SANGEETA** | **35** | **HISTORY** | **01/07/99** | **40000** | **F** |
| **5** | **RAKESH** | **42** | **MATHS** | **05/09/97** | **25000** | **M** |
| **6** | **SHYAM** | **50** | **HISTORY** | **27/06/98** | **30000** | **M** |
| **7** | **SHIV OM** | **44** | **COMPUTER** | **25/02/97** | **21000** | **M** |
| **8** | **SHALAKHA** | **33** | **MATHS** | **31/07/97** | **20000** | **F** |

**PROCEDURE:**

1. Start MySQL
2. Create a relation(table) given.
3. Insert the appropriate values into the table.
4. Write a queries for the given questions.
5. Execute and find the output for the given queries.
6. Stop MySQL

**QUERIES:**

1. To show all information about the teacher of history department.

**MYSQL> SELECT \* FROM TEACHER WHERE DEPARTMENT=’HISTORY’;**

1. To list the names of female teachers who are in Hindi department.

**MYSQL> SELECT \* FROM TEACHER WHERE DEPARTMENT=’HINDI’ AND SEX=’F’;**

1. To list names of all teachers with their date of joining in ascending order.

**MYSQL> SELECT \* FROM TEACHER ORDER BY DATEOFJOIN;**

1. To display teacher’s Name, Salary, Age for male teacher only**.**

**MYSQL> SELECT NAME, SALARY, AGE FROM TEACHER WHERE SEX=’M’;**

1. To count the number of teachers with Age >23.

**MYSQL> SELECT COUNT(\*) FROM TEACHER WHERE AGE>23;**

1. To insert a new row in the Teacher table with the following data:

**(9,”Vijay”, 26,”Computer”, {13/05/95}, 35000,”M”)**

**MYSQL> INSERT INTO TEACHER VALUES (9, ”Vijay”, 26,”Computer”, ‘13/05/95’, 35000,”M”);**

1. To show all information about the teachers in this table:

**MYSQL> SELECT \*FROM TEACHER;**

1. Add a new column named “Address”.

**MYSQL> ALTER TABLE TEACHER ADD ADDRESS CHAR (40);**

1. DISPLAY table in the alphabetical order to name :

**MYSQL> SELECT \*FROM TEACHER ORDER BY NAME;**

1. Display the age of the teachers whose name starts with ‘S’.

**MYSQL> SELECT AGE FROM TEACHER WHERE NAME LIKE “S%’;**

1. **Select count (Distinct Department) from Teacher;**

|  |
| --- |
| **count (Distinct Department)** |
| **COMPUTER**  **HISTORY**  **MATHS** |

1. **Select AVG (Salary) from Teacher where DateofJoin<’12/07/96’;**

|  |
| --- |
| **AVG (Salary)** |
| **NULL** |

1. **Select SUM (Salary) from Teacher where DateofJoin<’12/07/96’;**

|  |
| --- |
| **SUM (Salary)** |
| **NULL** |

1. **Select MAX(Age) from Teacher where Sex=”F”;**

|  |
| --- |
| **MAX(Age)** |
| **35** |

**RESULT:**

Thus the MySQL commands(queries) for (1) to (10) and output for (11) to (14) on the basis of ***Teacher*** relation given was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:18** | **MYSQL QUERIES - II** | **Date:** |

**AIM:**

To write MySQL commands(queries) for (1) to (4) and write output for (5) and (6) on the basis of relation ***WORKERS*** and ***DESIG*** given.

**TABLE WORKERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W\_ID** | **FIRSTNAME** | **LASTNAME** | **ADDRESS** | **CITY** |
| **102** | **SAM** | **TONES** | **33 ELM ST.** | **PARIS** |
| **105** | **SARAH** | **ACKERMAN** | **440 U.S. 110** | **NEW YORK** |
| **144** | **MANILA** | **SENGUPTA** | **24 FRIENDS STREET** | **NEW DELHI** |
| **210** | **GEORGE** | **SMITH** | **83 FIRST STREET** | **HOWARD** |
| **255** | **MARY** | **JONES** | **842 VINE AVE.** | **LOSANTIVILLE** |
| **300** | **ROBERT** | **SAMUEL** | **9 FIFTH CROSS** | **WASHINGTON** |
| **335** | **HENRY** | **WILLIAMS** | **12MOORE STREET** | **BOSTON** |
| **403** | **RONNY** | **LEE** | **121 HARRISON ST.** | **NEW YORK** |
| **451** | **PAT** | **THOMPSON** | **11 RED ROAD** | **PARIS** |

**TABLE: DESIG**

|  |  |  |  |
| --- | --- | --- | --- |
| **W\_ID** | **SALARY** | **BENEFITS** | **DESIGNATION** |
| **102** | **75000** | **15000** | **MANAGER** |
| **105** | **85000** | **25000** | **DIRECTOR** |
| **144** | **70000** | **15000** | **MANAGER** |
| **210** | **75000** | **12500** | **MANAGER** |
| **255** | **50000** | **12000** | **CLERK** |
| **300** | **45000** | **10000** | **CLERK** |
| **335** | **40000** | **10000** | **CLERK** |
| **403** | **32000** | **7500** | **SALESMAN** |
| **451** | **28000** | **7500** | **SALESMAN** |

**PROCEDURE:**

1. Start MySQL
2. Create a relation(table) given.
3. Insert the appropriate values into the table.
4. Write a queries for the given questions.
5. Execute and find the output for the given queries.
6. Stop MySQL

**QUERIES:**

1. To display the content of workers table in ascending order of first name.

**MYSQL>SELECT \* FROM WORKERS ORDER BY FIRSTNAME;**

1. To display the FIRSTNAME, CITY and TOTAL SALARY of all Clerks from the tables workers and design, where TOTAL SALARY = SALARY + BENEFITS.

**MYSQL>SELECT FIRSTNAME, CITY, SALARY + BENEFITS “TOTAL SALARY” FROM WORKERS;**

1. To display the minimum SALARY among Managers and Clerks from the table DESIG.

**MYSQL>SELECT MIN(SALARY) FROM DESIG WHERE DESIGNATION IN (MANAGER, CLERK);**

1. Increase the BENEFITS of all Salesmen by 10% in table DESIG.

**MYSQL>UDATE DESIG SET BENEFITS= BENEFITS+ BENEFITS/100\*10;**

1. **MYSQL>SELECT FIRSTNAME, SALARY FROM WORKERS, DESIG WHERE DESIGNATION = ‘Manager’ AND WORKERS.W\_ID = DESIG.W\_ID;**

|  |  |
| --- | --- |
| **FIRSTNAME** | **SALARY** |
| **SAM**  **MANILA**  **GEORGE** | **75000**  **70000**  **75000** |

1. **MYSQL>SELECT DESIGNATION, SUM(SALARY) FROM DESIG GROUP BY DESIGNATION HAVING COUNT (\*)>=2 ;**

|  |  |
| --- | --- |
| **DESIGNATION** | **SUM(SALARY)** |
| **MANAGER**  **CLERK**  **SALESMAN** | **42500**  **32000**  **15000** |

**RESULT:**

Thus the MySQL commands(queries) for (1) to (4) and output for (5) and (6) on the basis of ***Workers*** and ***Desig*** relation given was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:19** | **MYSQL QUERIES - III** | **Date:** |

**AIM:**

To write MySQL commands(queries) for (1) to (6) and write output for (7) and (10) on the basis of relation ***INTERIORS*** and ***NEWONES*** given.

**TABLE: INTERIORS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **ITEMNAME** | **TYPE** | **DATE OF STOCK** | **PRICE** | **DISCOUNT** |
| **1.**  **2.**  **3.**  **4.**  **5.**  **6.**  **7.**  **8.**  **9.**  **10.** | **RED ROSE**  **SOFT TOUCH**  **JERRY’S HOME**  **ROUGH WOOD**  **COMFORT ZONE**  **JERRY LOOK**  **LION KING**  **ROYAL TIGER**  **PARK SITTING**  **DINE PARADISE** | **DOUBLE BED**  **BABY COT**  **BABY COT**  **OFFICE TABLE**  **DOUBLE BED**  **BABY COT**  **OFFICE TABLE**  **SOFA**  **SOFA**  **DINING TABLE** | **23/02/02**  **20/01/02**  **19/02/02**  **01/01/02**  **12/01/02**  **24/02/02**  **20/02/02**  **22/02/02**  **13/12/01**  **19/02/02** | **32000**  **9000**  **8500 20000**  **15000**  **7000**  **16000**  **30000**  **9000**  **11000** | **15**  **10**  **10**  **20**  **20**  **19**  **20**  **25**  **15**  **15** |

**TABLE: NEWONES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO** | **ITEMNAME** | **TYPE** | **DATEOFSTOCK** | **PRICE** | **DISCOUNT** |
| **11.**  **12.**  **13.** | **WHITE WOOD**  **JAMES 007**  **TOM LOOK** | **DOUBLE BED**  **SOFA**  **BABY COT** | **23/03/03**  **20/02/03**  **21/02/03** | **20000**  **15000**  **7000** | **20**  **15**  **10** |

**PROCEDURE:**

1. Start MySQL
2. Create a relation(table) given.
3. Insert the appropriate values into the table.
4. Write a queries for the given questions.
5. Execute and find the output for the given queries.
6. Stop MySQL

**QUERIES:**

1. To show all information about the Sofa from the INTERIORS table.

**MYSQL> SELECT \*FROM INTERIORS WHERE TYPE=’SOFA’;**

1. To list the ITEMNAME which are priced at more than 10000 from the INTERIORS table?

**MYSQL> SELECT ITEMNAME FROMINTERIORS WHERE PRICE >10000;**

1. To list ITEMNAME and type of those items, in which DATEOFSTOCK is before 22/01/02 from INTERIORS table in descending order of ITEMNAME.

**MYSQL> SELECT \* FROMINTERIORS WHERE DATEOFSTOCK< ‘22/01/02’ ORDER BY ITEMNAME;**

1. To display ITEMNAME and DATEOFSTOCK of those items, in which the DISCOUNT percentage is more than 15 from INTERIORS table.

**MYSQL> SELECT ITEMNAME, DATEOFSTOCK FROMINTERIORS WHERE DISCOUNT>15;**

1. To count the number of items, whose TYPE is “Double Bed” from INTERIORS table?

**MYSQL> SELECT COUNT(\*) FROM INTERIORS WHERE TYPE=’DOUBLE BED’;**

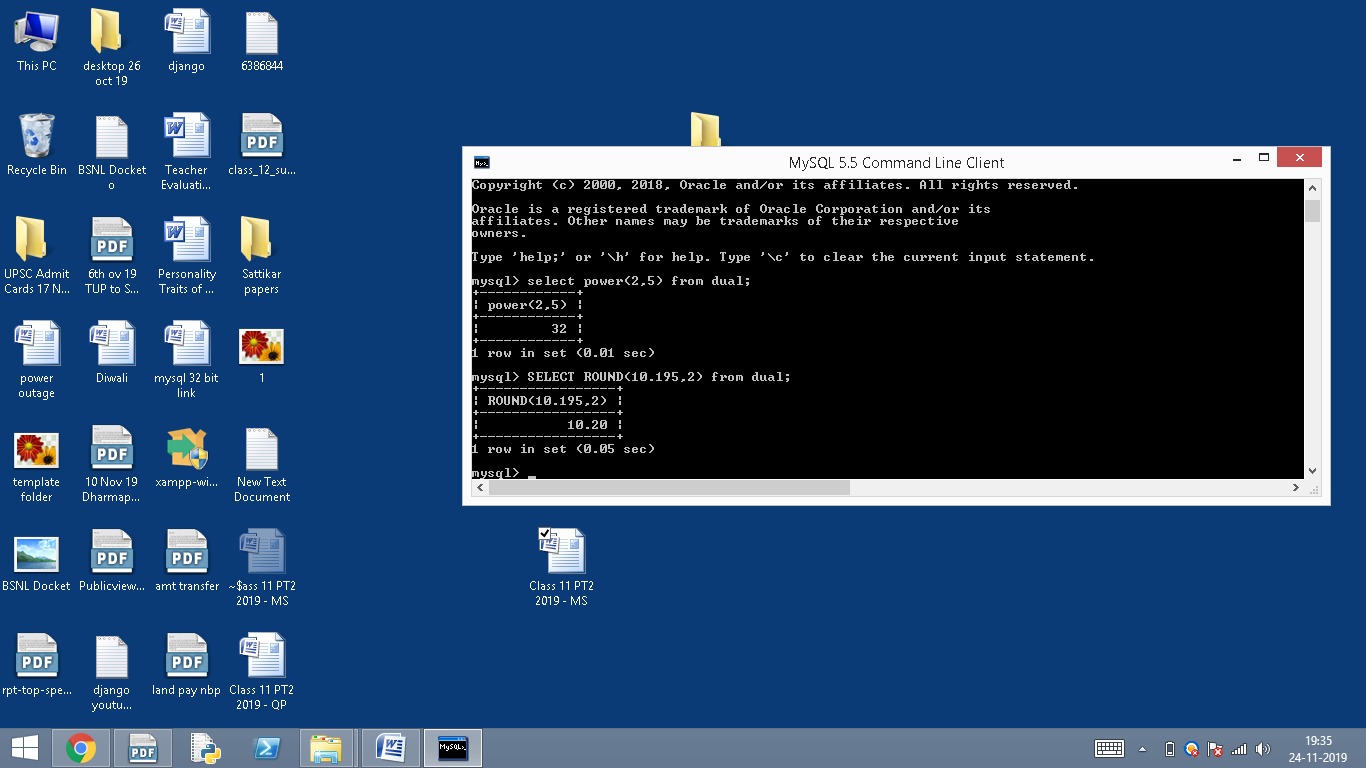
1. To insert a new row in the NEWONES table with the following data:

**(14,”True Indian”, “Office Table”, {28/03/03}, 15000, 20)**

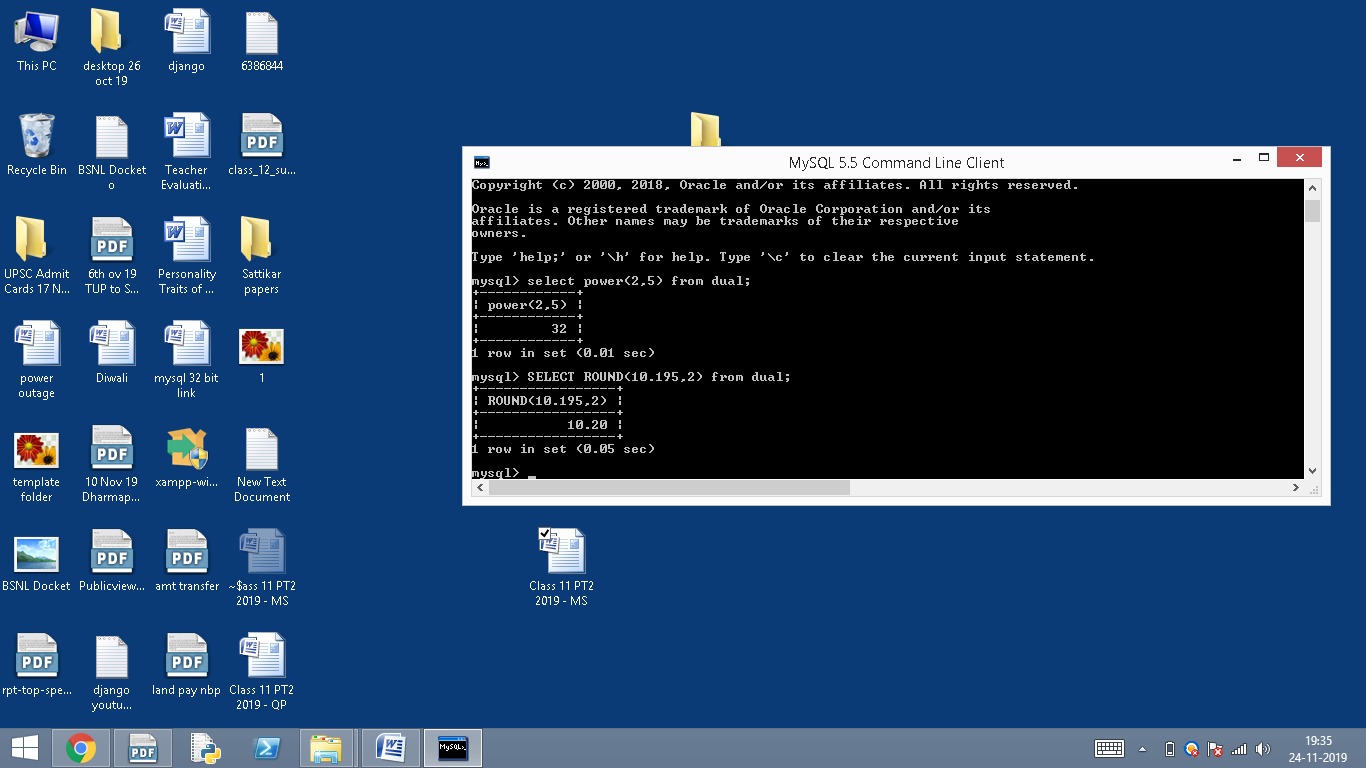
**MYSQL>INSERT INTO NEWONES VALUES (14,”TRUE INDIAN”, “OFFICE TABLE”, ‘28/03/03’, 15000, 20);**

**Write the outputs of the following SQL :**

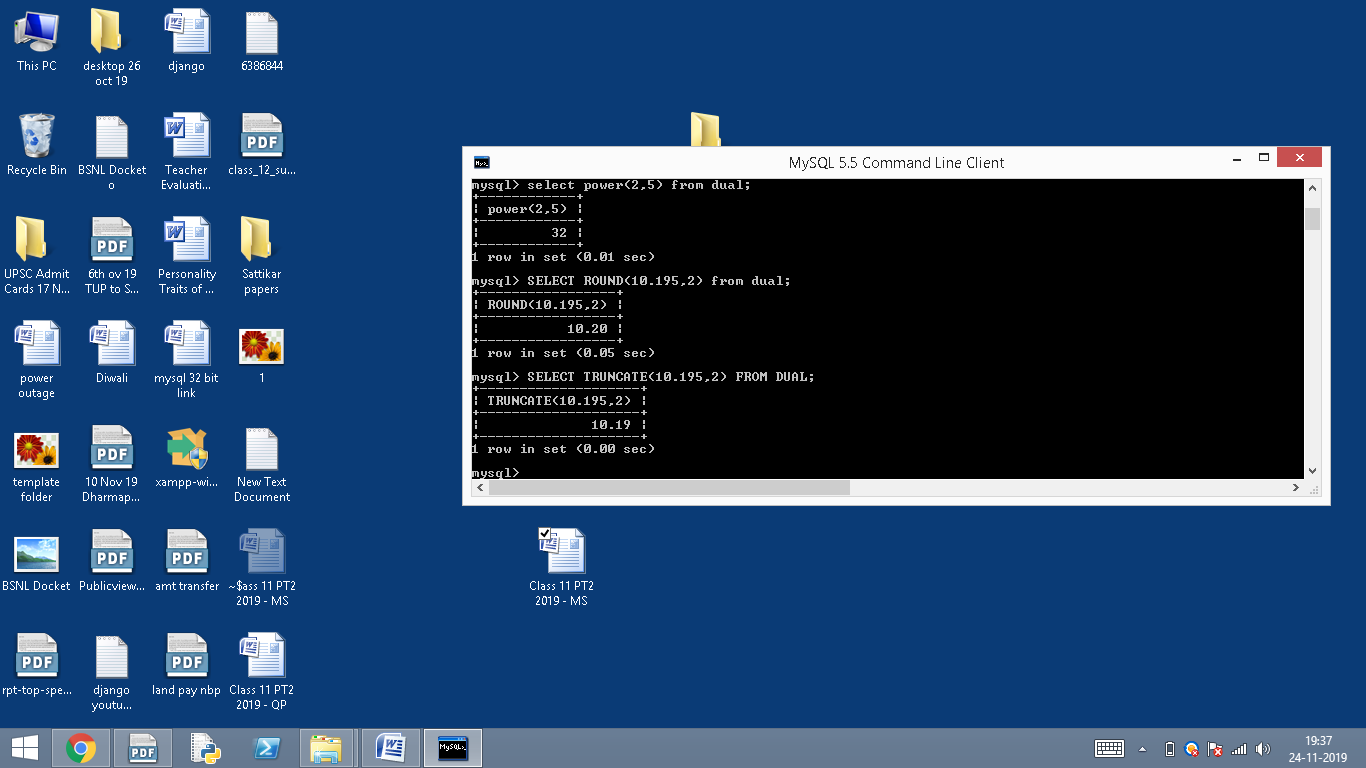
**7.) MYSQL> SELECT POWER(2,5) DUAL;**

****

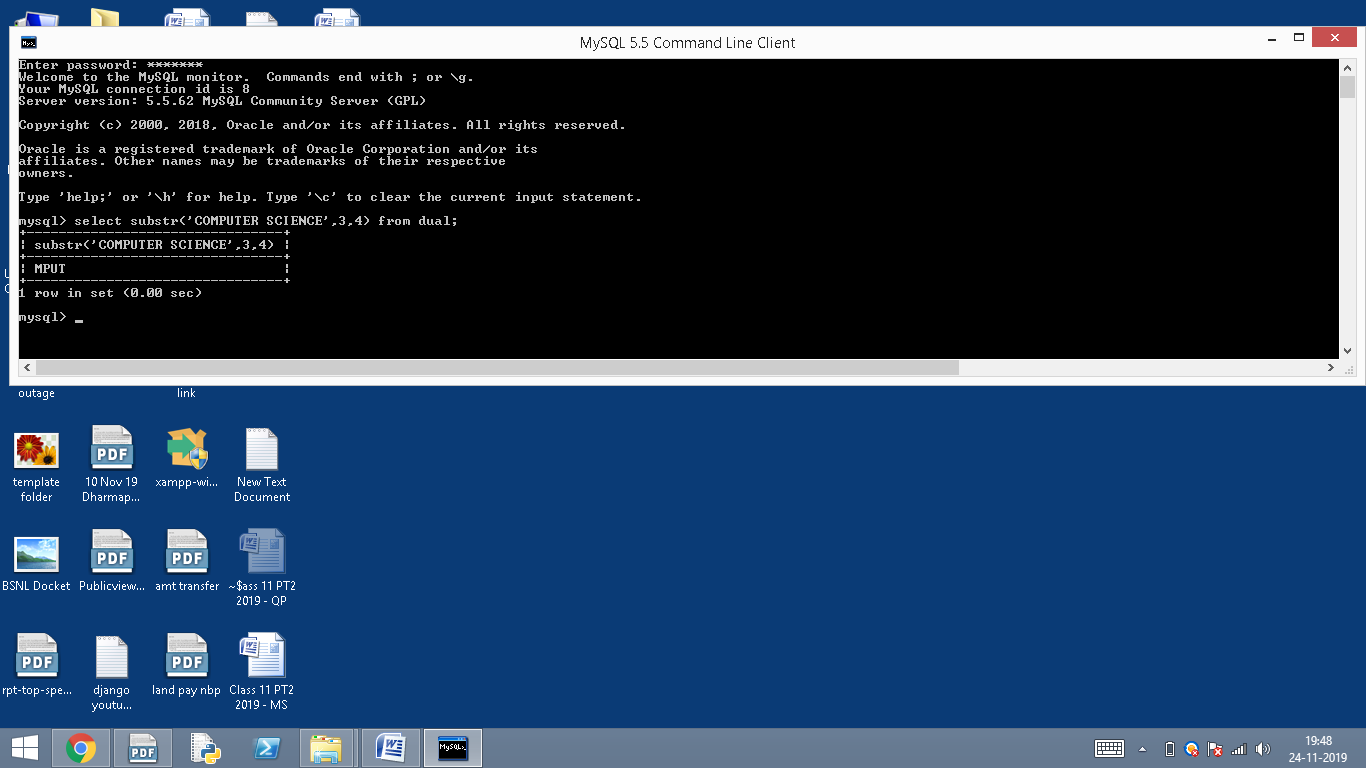
**8.) MYSQL> SELECT ROUND(10.195,2) FROM DUAL;**

****

**9.) MYSQL> SELECT TRUNCATE(10.195,2) FROM DUAL;**

****

**10.) MYSQL> SELECT substr(‘COMPUTER SCIENCE’,3,4)FROM DUAL;**

****

**RESULT:**

Thus the MySQL commands(queries) for (1) to (6) and output for (7) and (10) on the basis of ***Interiors*** and ***Newones*** relation given was executed successfully and the output was verified.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:20** | **INTERFACE PYTHON WITH MYSQL**  **CREATE & DISPLAY DATA FROM TEACHER TABLE** | **Date:** |

**AIM:**

To write a python program to connect with mysql in that create and insert the rows as per the below given table and fetch all records of the Teacher table.

**Table: Teacher**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name | Age | Department | Salary | Sex |
| 1 | Jigal | 34 | Computer | 12000 | M |
| 2 | Sharmila | 31 | History | 20000 | F |
| 3 | Sandeep | 32 | Maths | 30000 | M |
| 4 | Sangeeta | 35 | History | 40000 | F |
| 5 | Rakesh | 42 | Maths | 25000 | M |

**ALGORITHM:**

1. Start the Python program.
2. Import the packages required for database connectivity.
3. Open a connection to database by using ***connect()*** function and by passing required arguments.
4. Create a Cursor instance by using ***cursor()*** function.
5. Execute the given query.
6. Extract data from the result set.
7. Close the database connection by using ***close()*** function.
8. Stop the program.

**PROGRAM:**

**import mysql.connector as md**

**#CONNECTING TO DATABASE**

**mycon=md.connect(host="localhost",user="root",passwd="sagar@123",database="school")**

**if mycon.is\_connected():**

**print("Connected")**

**cur=mycon.cursor()**

**#CREATING TABLE**

**t="create table teacher(No int, Name char(10),Age int,Department char(15),Salary int,Sex char(6))"**

**cur.execute(t)**

**mycon.commit()**

**#INSERTING VALUES INTO TABLE**

**i="insert into teacher(no,name,age,department,salary,sex) values({},'{}',{},'{}',{},'{}')".format(1,"Jigal",34,"Computer",12000,'M')**

**cur.execute(i)**

**i="insert into teacher(no,name,age,department,salary,sex) values({},'{}',{},'{}',{},'{}')".format(2,"Sharmila",31,"History",20000,'F')**

**cur.execute(i)**

**i="insert into teacher(no,name,age,department,salary,sex) values({},'{}',{},'{}',{},'{}')".format(3,"Sandeep",32,"Maths",30000,'M')**

**cur.execute(i)**

**i="insert into teacher(no,name,age,department,salary,sex) values({},'{}',{},'{}',{},'{}')".format(4,"Sangeeta",35,"History",40000,'F')**

**cur.execute(i)**

**i="insert into teacher(no,name,age,department,salary,sex) values({},'{}',{},'{}',{},'{}')".format(5,"Rakesh",42,"Maths",25000,'M')**

**cur.execute(i)**

**mycon.commit()**

**#FETCHING DATA FROM TABLE**

**cur.execute("select \* from teacher")**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**mycon.close()**

**SAMPLE OUTPUT:**

**DataBase Connected**

**(1, 'Jigal', 34, 'Computer', 12000, 'M')**

**(2, 'Sharmila', 31, 'History', 20000, 'F')**

**(3, 'Sandeep', 32, 'Maths', 30000, 'M')**

**(4, 'Sangeeta', 35, 'History', 40000, 'F')**

**(5, 'Rakesh', 42, 'Maths', 25000, 'M')**

**Total rows retrieved so far is, 5**

**RESULT:**

Thus the python program to connect with mysql was connected and the displaying queries was executed successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:21** | **INTERFACE PYTHON WITH MYSQL**  **SEARCHING DATA FROM STUDENT TABLE** | **Date:** |

**AIM:**

To write a python program to connect with mysql in that create and insert the rows as per the below given table and fetch only those records of the students who have scored more than 450.

**Table: Student.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Name** | **Age** | **Department** | **Score** | **Sex** |
| 1 | Arpit | 62 | Surgery | 300 | M |
| 2 | Zayana | 18 | ENT | 250 | F |
| 3 | Kareem | 68 | Orthopedic | 460 | M |
| 4 | Abhilash | 26 | Surgery | 300 | M |
| 5 | Dhanya | 24 | ENT | 350 | F |

**ALGORITHM:**

1. Start the Python program.
2. Import the packages required for database connectivity.
3. Open a connection to database by using ***connect()*** function and by passing required arguments.
4. Create a Cursor instance by using ***cursor()*** function.
5. Execute the given query.
6. Extract data from the result set.
7. Close the database connection by using ***close()*** function.
8. Stop the program.

**PROGRAM:**

**import mysql.connector as md**

**#CONNECTING TO DATABASE**

**mycon=md.connect(host="localhost",user="root",passwd="sagar@123",database="school")**

**if mycon.is\_connected():**

**print("Database Connected Successfully")**

**cur=mycon.cursor()**

**#CREATING TABLE**

**t="create table student(No int, Name char(10),Age int,Department char(15),Score int,Sex char(6))"**

**cur.execute(t)**

**mycon.commit()**

**print("Table Created Successfully")**

**#INSERTING VALUES INTO TABLE**

**i="insert into student(no,name,age,department,score,sex) values({},'{}',{},'{}',{},'{}')".format(1,"Arpit",62,"Surgery",300,'M')**

**cur.execute(i)**

**i="insert into student(no,name,age,department,score,sex) values({},'{}',{},'{}',{},'{}')".format(2,"Zayana",18,"ENT",250,'F')**

**cur.execute(i)**

**i="insert into student(no,name,age,department,score,sex) values({},'{}',{},'{}',{},'{}')".format(3,"Kareem",68,"Orthopedic",460,'M')**

**cur.execute(i)**

**i="insert into student(no,name,age,department,score,sex) values({},'{}',{},'{}',{},'{}')".format(4,"Abhilash",26,"Surgery",300,'M')**

**cur.execute(i)**

**i="insert into student(no,name,age,department,score,sex) values({},'{}',{},'{}',{},'{}')".format(5,"Dhanya",24,"ENT",350,'F')**

**cur.execute(i)**

**mycon.commit()**

**print("Values Inserted Successfully")**

**#FETCHING DATA FROM TABLE**

**q="select \* from student where score>={}".format(450)**

**cur.execute(q)**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**print("Mark above 450 is follows")**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**mycon.close()**

**SAMPLE OUTPUT:**

**Database Connected Successfully**

**Table Created Successfully**

**Values Inserted Successfully**

**Mark above 450 is follows**

**(3, 'Kareem', 68, 'Orthopedic', 460, 'M')**

**Total rows retrieved so far is, 1**

**RESULT:**

Thus the python program to connect with mysql was connected and the searching queries was executed successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:22** | **INTERFACE PYTHON WITH MYSQL**  **DELETING RECORD FROM BOOK TABLE** | **Date:** |

**AIM:**

To write a python program to connect with mysql in that create and insert the rows as per the below given table and delete all records from ***Book*** table

**Table: Book**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B\_Id | Author\_Name | Publisher | Price | Quantity |
| C01 | Lata Kapoor | EPB | 355 | 5 |
| F01 | William Hopkins | First | 650 | 20 |
| T01 | Brain Brooke | FPB | 350 | 10 |
| T02 | A.W.Rossaine | TDH | 350 | 15 |
| F02 | Anna Roberts | First | 750 | 50 |

**ALGORITHM:**

1. Start the Python program.
2. Import the packages required for database connectivity.
3. Open a connection to database by using ***connect()*** function and by passing required arguments.
4. Create a Cursor instance by using ***cursor()*** function.
5. Execute the given query.
6. Extract data from the result set.
7. Close the database connection by using ***close()*** function.
8. Stop the program.

**PROGRAM:**

**import mysql.connector as md**

**#CONNECTING TO DATABASE**

**mycon=md.connect(host="localhost",user="root",passwd="sagar@123",database="school")**

**if mycon.is\_connected():**

**print("DataBase Connected")**

**cur=mycon.cursor()**

**#CREATING TABLE**

**t="create table Book(B\_Id char(3), Author\_Name char(15),Publisher char(10),Price int,Quantity int)"**

**cur.execute(t)**

**mycon.commit()**

**print("Table Created Successfully")**

**#INSERTING VALUES INTO TABLE**

**i="insert into Book(B\_Id,Author\_Name,Publisher,Price,Quantity) values('{}','{}','{}',{},{})".format('C01','Lata Kapoor','EPB',355,5)**

**cur.execute(i)**

**i="insert into Book(B\_Id,Author\_Name,Publisher,Price,Quantity) values('{}','{}','{}',{},{})".format('F01','William Hopkins','First',650,20)**

**cur.execute(i)**

**i="insert into Book(B\_Id,Author\_Name,Publisher,Price,Quantity) values('{}','{}','{}',{},{})".format('T01','Brain Brooke','FPB',350,10)**

**cur.execute(i)**

**i="insert into Book(B\_Id,Author\_Name,Publisher,Price,Quantity) values('{}','{}','{}',{},{})".format('T01','A.W.Rossaine','TDH',350,15)**

**cur.execute(i)**

**i="insert into Book(B\_Id,Author\_Name,Publisher,Price,Quantity) values('{}','{}','{}',{},{})".format('F01','Anna Roberts','First',750,50)**

**cur.execute(i)**

**mycon.commit()**

**print("Values Inserted Successfully")**

**print("BEFORE REMOVING RECORDS FROM BOOK-TABLE:")**

**#FETCHING DATA FROM TABLE**

**q="select \* from Book"**

**cur.execute(q)**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**#REMOVING RECORDS FROM TABLE:**

**q="delete from Book"**

**cur.execute(q)**

**mycon.commit()**

**print("\nRECORDS DELETED")**

**print("\nAFTER REMOVING RECORDS FROM BOOK-TABLE:")**

**#FETCHING DATA FROM TABLE**

**q="select \* from Book"**

**cur.execute(q)**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**mycon.close()**

**SAMPLE OUTPUT:**

DataBase Connected

Table Created Successfully

Values Inserted Successfully

BEFORE REMOVING RECORDS FROM BOOK-TABLE:

('C01', 'Lata Kapoor', 'EPB', 355, 5)

('F01', 'William Hopkins', 'First', 650, 20)

('T01', 'Brain Brooke', 'FPB', 350, 10)

('T01', 'A.W.Rossaine', 'TDH', 350, 15)

('F01', 'Anna Roberts', 'First', 750, 50)

Total rows retrieved so far is, 5

RECORDS DELETED

AFTER REMOVING RECORDS FROM BOOK-TABLE:

Total rows retrieved so far is, 0

**RESULT:**

Thus the python program to connect with mysql was connected and the deleting records was deleted and executed successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:23** | **INTERFACE PYTHON WITH MYSQL**  **UPDATING RECORDS FROM CANDIDATE TABLE** | **Date:** |

**AIM:**

To write a python program to connect with mysql in that create and insert the rows as per the above given table and update the marks of a Candidate whose name and Avgmarks to be updated are given by the user.

**Table: Candidate.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sno** | **Name** | **Stipend** | **Subject** | **Avgmark** |
| 1 | KARAN | 400 | PHYSICS | 68 |
| 2 | DIWAKAR | 450 | COMP. Sc. | 68 |
| 3 | DIVYA | 300 | CHEMISTRY | 62 |
| 4 | REKHA | 350 | PHYSICS | 63 |
| 5 | ARJUN | 500 | MATHS | 70 |

**ALGORITHM:**

1. Start the Python program.
2. Import the packages required for database connectivity.
3. Open a connection to database by using ***connect()*** function and by passing required arguments.
4. Create a Cursor instance by using ***cursor()*** function.
5. Execute the given query.
6. Extract data from the result set.
7. Close the database connection by using ***close()*** function.
8. Stop the program.

**PROGRAM:**

**import mysql.connector as md**

**#CONNECTING TO DATABASE**

**mycon=md.connect(host="localhost",user="root",passwd="sagar@123",database="murali")**

**if mycon.is\_connected():**

**print("DataBase Connected")**

**cur=mycon.cursor()**

**#CREATING TABLE**

**t="create table Candidate(Sno int, Name char(15),Stipend int,Subject char(15),Avgmark int)"**

**cur.execute(t)**

**mycon.commit()**

**print("Table Created Successfully")**

**#INSERTING VALUES INTO TABLE**

**i="insert into Candidate(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(1,'KARAN',400,'PHYSICS',68)**

**cur.execute(i)**

**i="insert into Candidate(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(2,'DIWAKAR',450,'COMP.Sc.',68)**

**cur.execute(i)**

**i="insert into Candidate(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(3,'DIVYA',300,'CHEMISTRY',62)**

**cur.execute(i)**

**i="insert into Candidate(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(4,'REKHA',350,'PHYSICS',63)**

**cur.execute(i)**

**i="insert into Candidate(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(5,'ARJUN',500,'MATHS',70)**

**cur.execute(i)**

**mycon.commit()**

**print("Values Inserted Successfully")**

**print("BEFORE UPDATING RECORDS FROM TABLE:")**

**#FETCHING DATA FROM TABLE**

**q="select \* from Candidate"**

**cur.execute(q)**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**#UPDATING RECORDS IN A TABLE:**

**print("\nUpdating Record")**

**name=input("Enter a Name:")**

**mark=int(input("Enter a New Mark:"))**

**q="update Candidate set Avgmark={} where Name='{}'".format(mark,name)**

**cur.execute(q)**

**mycon.commit()**

**print("\nRECORD UPDATED")**

**print("\nAFTER UPDATING RECORDS FROM TABLE:")**

**#FETCHING DATA FROM TABLE**

**q="select \* from Candidate"**

**cur.execute(q)**

**data=cur.fetchall()**

**#DISPLAYING TABLE DATA**

**for i in data:**

**print(i)**

**print("Total rows retrieved so far is,",cur.rowcount)**

**mycon.close()**

**SAMPLE OUTPUT:**

**DataBase Connected**

**Table Created Successfully**

**Values Inserted Successfully**

**BEFORE UPDATING RECORDS FROM TABLE:**

**(1, 'KARAN', 400, 'PHYSICS', 68)**

**(2, 'DIWAKAR', 450, 'COMP.Sc.', 68)**

**(3, 'DIVYA', 300, 'CHEMISTRY', 62)**

**(4, 'REKHA', 350, 'PHYSICS', 63)**

**(5, 'ARJUN', 500, 'MATHS', 70)**

**Total rows retrieved so far is, 5**

**Updating Record**

**Enter a Name:KARAN**

**Enter a New Mark:86**

**RECORD UPDATED**

**AFTER UPDATING RECORDS FROM TABLE:**

**(1, 'KARAN', 400, 'PHYSICS', 86)**

**(2, 'DIWAKAR', 450, 'COMP.Sc.', 68)**

**(3, 'DIVYA', 300, 'CHEMISTRY', 62)**

**(4, 'REKHA', 350, 'PHYSICS', 63)**

**(5, 'ARJUN', 500, 'MATHS', 70)**

**Total rows retrieved so far is, 5**

**RESULT:**

Thus the python program to connect with mysql was connected and the updating records was updated and executed successfully.

|  |  |  |
| --- | --- | --- |
| **Ex.No.:24** | **INTERFACE PYTHON WITH MYSQL**  **ADDING NEW COLUMN TO SCHOOL TABLE** | **Date:** |

**AIM:**

To write a python program to connect with mysql in that create and insert the rows as per the above given table and insert one more column ***“total”*** as integer type.

**Table: School**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sno | Name | Stipend | Subject | Avgmark |
| 1 | KARAN | 400 | PHYSICS | 68 |
| 2 | DIWAKAR | 450 | COMP. Sc. | 68 |
| 3 | DIVYA | 300 | CHEMISTRY | 62 |
| 4 | REKHA | 350 | PHYSICS | 63 |
| 5 | ARJUN | 500 | MATHS | 70 |

**ALGORITHM:**

1. Start the Python program.
2. Import the packages required for database connectivity.
3. Open a connection to database by using ***connect()*** function and by passing required arguments.
4. Create a Cursor instance by using ***cursor()*** function.
5. Execute the given query.
6. Extract data from the result set.
7. Close the database connection by using ***close()*** function.
8. Stop the program.

**PROGRAM:**

import mysql.connector as md

#CONNECTING TO DATABASE

mycon=md.connect(host="localhost",user="root",passwd="sagar@123",database="murali")

if mycon.is\_connected():

print("DataBase Connected")

cur=mycon.cursor()

#CREATING TABLE

t="create table School(Sno int, Name char(15),Stipend int,Subject char(15),Avgmark int)"

cur.execute(t)

mycon.commit()

print("Table Created Successfully")

#INSERTING VALUES INTO TABLE

i="insert into School(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(1,'KARAN',400,'PHYSICS',68)

cur.execute(i)

i="insert into School(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(2,'DIWAKAR',450,'COMP.Sc.',68)

cur.execute(i)

i="insert into School(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(3,'DIVYA',300,'CHEMISTRY',62)

cur.execute(i)

i="insert into School(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(4,'REKHA',350,'PHYSICS',63)

cur.execute(i)

i="insert into School(Sno,Name,Stipend,Subject,Avgmark) values({},'{}',{},'{}',{})".format(5,'ARJUN',500,'MATHS',70)

cur.execute(i)

mycon.commit()

print("Values Inserted Successfully")

print("BEFORE ADDING COLUMN IN A TABLE:")

#FETCHING DATA FROM TABLE

q="select \* from School"

cur.execute(q)

data=cur.fetchall()

#DISPLAYING TABLE DATA

for i in data:

print(i)

print("Total rows retrieved so far is,",cur.rowcount)

#ADDING COLUMN INTO A TABLE:

print("\nAdding Column")

a="alter table School add Total int"

cur.execute(a)

mycon.commit()

print("\nNEW COLUMN ADDED SUCCESSFULLY")

print("\nAFTER ADDING NEW COLUMN TO A TABLE:")

#FETCHING DATA FROM TABLE

q="select \* from School"

cur.execute(q)

data=cur.fetchall()

#DISPLAYING TABLE DATA

for i in data:

print(i)

print("Total rows retrieved so far is,",cur.rowcount)

mycon.close()

**SAMPLE OUTPUT:**

DataBase Connected

Table Created Successfully

Values Inserted Successfully

BEFORE ADDING COLUMN IN A TABLE:

(1, 'KARAN', 400, 'PHYSICS', 68)

(2, 'DIWAKAR', 450, 'COMP.Sc.', 68)

(3, 'DIVYA', 300, 'CHEMISTRY', 62)

(4, 'REKHA', 350, 'PHYSICS', 63)

(5, 'ARJUN', 500, 'MATHS', 70)

Total rows retrieved so far is, 5

Adding Column

NEW COLUMN ADDED SUCCESSFULLY

AFTER ADDING NEW COLUMN TO A TABLE:

(1, 'KARAN', 400, 'PHYSICS', 68, None)

(2, 'DIWAKAR', 450, 'COMP.Sc.', 68, None)

(3, 'DIVYA', 300, 'CHEMISTRY', 62, None)

(4, 'REKHA', 350, 'PHYSICS', 63, None)

(5, 'ARJUN', 500, 'MATHS', 70, None)

Total rows retrieved so far is, 5

**RESULT:**

Thus the python program to connect with mysql was connected and the adding new column was added and executed successfully.