power-and-multiplication

April 9, 2024

[ ]:
```
from functools import partial
def power(a,b):
  return a**b
power_2=partial(power,a=2)
print("The powers of 2 from 1 to 10 are:")
for i in range(1,11):
  print("2 ^",i,":",power_2(b=i))
```

```
The powers of 2 from 1 to 10 are:
2 ^ 1 : 2
2 ^ 2 : 4
2 ^ 3 : 8
2 ^ 4 : 16
2 ^ 5 : 32
2 ^ 6 : 64
2 ^ 7 : 128
2 ^ 8 : 256
2 ^ 9 : 512
2 ^ 10 : 1024
```

[ ]:
```
def mul(a,b):
  return a*b
k=1
n=int(input("Enter no of times the tables gets printed:"))
while k<=n:
  for i in range(int(input("Enter 1st factor initial range:")),int(input("Enter␣
 ↪1st factor end range:"))):
    print("The",i,"table is:")
    for j in range(int(input("Enter 2nd factor inital range:
 ↪")),int(input("Enter 2nd factor end range:"))):
      print(i,"*",j,":",mul(a=i,b=j))
  k+=1
```

```
Enter no of times the tables gets printed:2
Enter 1st factor initial range:1
Enter 1st factor end range:2
The 1 table is:
Enter 2nd factor inital range:1
```

```
Enter 2nd factor end range:11
1 * 1 : 1
1 * 2 : 2
1 * 3 : 3
1 * 4 : 4
1 * 5 : 5
1 * 6 : 6
1 * 7 : 7
1 * 8 : 8
1 * 9 : 9
1 * 10 : 10
Enter 1st factor initial range:10
Enter 1st factor end range:12
The 10 table is:
Enter 2nd factor inital range:1
Enter 2nd factor end range:11
10 * 1 : 10
10 * 2 : 20
10 * 3 : 30
10 * 4 : 40
10 * 5 : 50
10 * 6 : 60
10 * 7 : 70
10 * 8 : 80
10 * 9 : 90
10 * 10 : 100
The 11 table is:
Enter 2nd factor inital range:1
Enter 2nd factor end range:11
11 * 1 : 11
11 * 2 : 22
11 * 3 : 33
11 * 4 : 44
11 * 5 : 55
11 * 6 : 66
11 * 7 : 77
11 * 8 : 88
11 * 9 : 99
11 * 10 : 110
```

simple-calculator

April 9, 2024

[ ]:
```
#Simple Calculator
def add(a,b):
  return a+b
def sub(a,b):
  return a-b
def mul(a,b):
  return a*b
def div(a,b):
  if b!=0:
    return a/b
  else:
    return "Zero Division is not possible"
def percent(a):
  return (a/100)*100
def power(a,b):
  return a**b

def generate_solution(choose,a,b):
  if choose=='add':
    print(add(a,b))
  elif choose=='sub':
    print(sub(a,b))
  elif choose=='mul':
    print(mul(a,b))
  elif choose=='div':
    print(div(a,b))
  elif choose=='percent':
    print(percent(a,b))
  elif choose=='power':
    print(power(a,b))
  else:
    print("Entered Invalid Operator")
num_1=int(input("Enter 1st Value:"))
num_2=int(input("Enter 2nd Value:"))
choose_operator=input("|Add|Sub|Mul|Div|Percent|Power|:").lower()
if __name__=="__main__":
  generate_solution(choose_operator,num_1,num_2)
```

```
Enter 1st Value:2
Enter 2nd Value:0
|Add|Sub|Mul|Div|Percent|Power|:div
Zero Division is not possible
```

special-numbers

April 9, 2024

[15]:
```python
#printing even numbers using list comprehension from 1 to 50 numbers
numbers=[x for x in range(1,51)if x%2==0]
print(numbers)
#to print odd numbers just need to replace the condition
#x%2==0
#  to
#x%2==1
#Use case: Not only printing even or odd numbers but
#we can find multiples of the specific number
#Just need to replace the condition x%2==0 to x%required number==0
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50]
```

[29]:
```python
#to find factorial of the number using recursion function
#ex: 5!=5*4*3*2*1=120
def factorial(n):
  if n==1:
    return 1
  else:
    return n*factorial(n-1)
for i in range(int(input("Enter range:"))):
  print(factorial(n=int(input("Enter the number:"))))
```

```
Enter range:5
Enter the number:1
1
Enter the number:2
2
Enter the number:5
120
Enter the number:10
3628800
Enter the number:1
1
```

[49]:
```python
#to find factorial of the number without recursion
def fact(n):
```

```
  if n==1:
    return 1
  else:
    i=1
    f=1
    while i<=n:
      f*=i
      i+=1
    return f
num=int(input("Enter the value:"))
fact(num)
```

```
Enter the value:5
```

[49]:
```
120
```

[46]:
```
#program to check whether the given number is Harshad Number or not
#Use case: Harshad Number is the number which is divisible by its
#sum of digits
#type 1
def harshad(n):
  n=int(n)
  temp=n
  sum=0
  while n>0:
    digit=n%10
    sum+=digit
    n=n//10
  if temp%sum==0:
    return f"Given number {temp} is Harshad Number"
  else:
    return f"Given number {temp} is not Harshad number"
num=input('Enter the number:')
harshad(num)
```

```
Enter the number:11
```

[46]:
```
'Given number 11 is not Harshad number'
```

[57]:
```
#type 2
def harshad_number(n):
  sum_value=sum(int(digits) for digits in str(n))
  return n%sum_value==0
number=int(input("Enter the value:"))
if harshad_number(number):
  print(f"{number} is Harshad Number")
else:
```

```
  print(f"{number} is not a Harshad Number")
```

```
Enter the value:48
48 is Harshad Number
```

[64]:
```
#program to check whether the given number is prime or not
#use case: Prime numbers are the numbers which can divisible
#only by its number and 1
def is_prime(n):
  i=1
  fc=0
  while i<=n:
    if n%i==0:
      fc+=1
    i+=1
  if fc==2:
    return f"{n} is a prime number"
  else:
    return f"{n} is not a prime number"
is_prime(2)
```

[64]:
```
'9 is not a prime number'
```

[69]:
```
#program to reverse the number
def rev(n):
  temp=n
  rev=0
  while n>0:
    r=n%10
    rev=(rev*10)+r
    n=n//10
  return f"The reverse of {temp} is {rev}"
num=int(input("Enter the number:"))
rev(num)
```

```
Enter the number:234
```

[69]:
```
'The reverse of 234 is 432'
```

[71]:
```
#program to calculate the sum of digits of a number
def sum_of_digits():
  n=int(input("Enter the number:"))
  temp=n
  sum=0
  while n>0:
    digits=n%10
    sum+=digits
```

```
    n=n//10
  return f"The sum of digits of number {temp} is {sum}"
sum_of_digits()
```

```
Enter the number:234
```

[71]:
```
'The sum of digits of number 234 is 9'
```

[78]:
```
#program to calculate the product of digits of a number
def prod_of_digits():
  n=int(input("Enter the number:"))
  temp=n
  prod=1
  while n>0:
    digits=n%10
    prod*=digits
    n=n//10
  return f"The product of digits of number {temp} is {prod}"
prod_of_digits()

#We can use even for finding factorial of number
#if the numbers are in sequence from 1,2,3,4...,9
#we can find until factorial of 9
```

```
Enter the number:135
```

[78]:
```
'The product of digits of number 135 is 15'
```

[90]:
```
#program to calculate whether the given number is
#An Armstrong Number or not
def armstrong_num(n):
  v=str(n)
  power_value=len(v)
  sum=0
  n=int(n)
  for i in v:
    prod=int(i)**power_value
    sum+=prod
  if sum==n:
    return f"The number {n} is an Armstrong Number"
  else:
    return f"The number {n} is not an Armstrong Number"
num=int(input('Enter the number:'))
armstrong_num(num)
```

```
Enter the number:2345
```

[90]:
```
'The number 2345 is not an Armstrong Number'
```

[95]:
```
#program to print perfect number
#Use Case:A number is said to be perfect number when the sum of its proper
#divisors(excluding its number) is equal to the given number
#ex: 6->the divisors are 1,2,3
#it is perfect number as 6->1+2+3=6
def perfect():
  n=int(input("Enter number:"))
  sum=0
  for i in range(1,n):
    if n%i==0:
      sum+=i
  if sum==n:
    return f"{n} is a Perfect Number"
  else:
    return f"{n} is not a Perfect Number"
perfect()
```

```
Enter number:12
```

[95]:
```
'12 is not a perfect number'
```

[103]:
```
#Program to find whether the given number is palindrome or not
#Use Case:Palindrome number is satisfies when the reversed is also the same
#original number
#ex: taking number 121 if we reverse it we get the same original number 121
def palindrome(n):
  temp=n
  rev=0
  while n>0:
    r=n%10
    rev=(rev*10)+r
    n=n//10
  if temp==rev:
    return f"{temp} is a palindrome number"
  else:
    return f"{temp} is not a palindrome number"
num=int(input('enter number:'))
palindrome(num)
```

```
enter number:123321
```

[103]:
```
'123321 is a palindrome number'
```

[111]:
```
#Program to create a fibonacci series
#Use case: Fibonacci series is the series of numbers where
#the next value is the sum of its preceeding 2 values
def fibonacci(a,b,limit):
```

```
  fibo_series=[]
  while a<limit:
    fibo_series.append(a)
    a,b=b,a+b
  return fibo_series
a=int(input('Enter Initial Value:'))
b=int(input('Enter Second Value:'))
limit=int(input("Enter the limit:"))
print(f"The fibonacci series from {a} to {limit} is:")
fibonacci_series=fibonacci(a,b,limit)
print(*fibonacci_series)
```

```
Enter Initial Value:0
Enter Second Value:1
Enter the limit:20
The fibonacci series from 0 to 20 is:
0 1 1 2 3 5 8 13
```

[114]:
```
#Program to create an arithmetic progression series
#Use Case: Arithmeticprogression (AP) is a sequence of numbers in which the
#difference between any two consecutive terms is constant.
#This difference is called the "common difference" and is denoted by d
def arithmetic_progression(a,d,n):
  progression=[a+(i*d) for i in range(n)]
  return progression
a=int(input('Enter Initial term:'))
d=int(input('Enter Common Difference:'))
n=int(input('Enter the nth term:'))
print('The Arithmetic Progression is:')
print(arithmetic_progression(a,d,n))
```

```
Enter Initial term:1
Enter Common Difference:3
Enter the nth term:10
The Arithmetic Progression is:
[1, 4, 7, 10, 13, 16, 19, 22, 25, 28]
```

[115]:
```
#Program to generate geometric progression
#Use case: In a geometric progression (or sequence), each term is found by
#multiplying the previous term by a constant value.
#This constant value is called the "common ratio" and is denoted by r
def geometric_progression(a,r,n):
  progression=[a*(r**i) for i in range(n)]
  return progression
a=int(input('Enter Initial term:'))
r=int(input('Enter Common Ratio:'))
n=int(input('Enter the nth term:'))
```

```
print('The Geomteric Progression is:')
print(geometric_progression(a,r,n))
```

```
Enter Initial term:1
Enter Common Ratio:2
Enter the nth term:10
The Geomteric Progression is:
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

rock-paper-scissors

April 9, 2024

Rock-Paper-Scissors game

Description:

1.Rock-Paper-Scissors game is 2 player game

2.The winning cases for user are:

User————-Computer

Scissors——-Paper

Rock————-Scissors

Paper————Rock

[ ]:
```
#Importing random module for random selections
import random as rn

#creating a user defined function for user
def user():
  while True:
    user_choice=input("Enter your choice(Rock|Paper|Scissors):").lower()
    if user_choice in ['rock','paper','scissors']:
      return user_choice
    else:
      print("Invalid choice|||| Please choose the above options")

#creating a user defined function for computer
def computer():
  return rn.choice(['rock','paper','scissors'])

#creating a function for determining the winner at each round
def determine_winner(user_choice,computer_choice):
  if user_choice==computer_choice:
    return "Its a tie!!!"
  elif (user_choice=='rock'and computer_choice=='scissors') or\
   (user_choice=='paper' and computer_choice=='rock') or\
    (user_choice=='scissors' and computer_choice=='paper'):
    return "YOU WIN!!!"
  else:
```

```
    return "You Loose!!!"

#creating main function to play the game by adjoining the user defined functions
def play_game():
  user_score=0
  computer_score=0
  while True:
    user_choice=user()
    computer_choice=computer()
    print(f"Your Choice is {user_choice}&computer choice is {computer_choice}")
    result=determine_winner(user_choice,computer_choice)
    print(result)
    if result=='YOU WIN!!!':
      user_score+=1
    elif result=='You Loose!!!':
      computer_score+=1
    print('-'*10,'Score','-'*10)
    print(f'User {user_score} : {computer_score} Computer')
    play_again=input('Do you want to play again (Yes/No):').lower()
    if play_again=="no":
      break
  if user_score>computer_score:
    print("User wins")
  elif user_score==computer_score:
    print("Both are tie")
  else:
    print("Computer wins")
  print("Thanks for playing")

#calling the main function
if __name__=="__main__":
  play_game()
```

```
Enter your choice(Rock|Paper|Scissors):scissors
Your Choice is scissors and computer choice is rock
You Loose!!!
---------- Score ----------
User 0 : 1 Computer
Do you want to play again (Yes/No):yes
Enter your choice(Rock|Paper|Scissors):paper
Your Choice is paper and computer choice is rock
YOU WIN!!!
---------- Score ----------
User 1 : 1 Computer
Do you want to play again (Yes/No):no
Both are tie
Thanks for playing
```

operations-on-python-lists

April 9, 2024

[75]:
```
51#Program to insert the elements in a list
a=list() #or we can use a=[]
n=int(input('Enter no of elements:'))
for i in range(n):
  x=int(input('Enter the elements:'))
  a=a+[x] #Its adds each element in list
print(a)
```

```
Enter no of elements:5
Enter the elements:1
Enter the elements:2
Enter the elements:4
Enter the elements:3
Enter the elements:5
[1, 2, 4, 3, 5]
```

[76]:
```
#Accessing elements in lists
#using index values we can access elements
#In the above program we already created a list
#so no need to create again
print(a[0]) #Accessing 1st element
print(a[1]) #Accessing 2nd element
print(a[2]) #Accessing 3rd element
print(a[:3]) #Accessing from 1st to 3rd elements
print(a[::2]) #Accessing the elements using step count
print(*a[::2]) #Cheat code to print the elements without list
```

```
1
2
4
[1, 2, 4]
[1, 4, 5]
1 4 5
```

[77]:
```
#Opeartions on list
#1. append(x): The method used to add each element at end of list
print("Before appending element")
print(a)
```

```
print("After appending element")
a.append(6)
print(a)
print('-'*20)
```

```
Before appending element
[1, 2, 4, 3, 5]
After appending element
[1, 2, 4, 3, 5, 6]
--------------------
```

[78]:
```
#2. extend([x]): The method used to add a set of elements of list
#note:the elements must be in list
print('Before Extending')
print(a)
print('After extending')
a.extend([7,8,9])
print(a)
print('-'*20)
```

```
Before Extending
[1, 2, 4, 3, 5, 6]
After extending
[1, 2, 4, 3, 5, 6, 7, 8, 9]
--------------------
```

[79]:
```
#3. insert(i,x): The method used to add the element at particular index
print('After Inserting')
a.insert(1,2)
print(a)
print('-'*20)
```

```
After Inserting
[1, 2, 2, 4, 3, 5, 6, 7, 8, 9]
--------------------
```

[80]:
```
#4. remove(x): The method to remove first occured element in list
print('After removing')
a.remove(2)
print(a)
print('-'*20)
```

```
After removing
[1, 2, 4, 3, 5, 6, 7, 8, 9]
--------------------
```

[81]:
```
#5. pop(i): The method used to remove the element at given index
#if we do not specified index then it will remove the last element
print('Pop the element at specified index')
a.pop(5)
print(a)
print('-'*20)
print('Pop the element without specifying index')
a.pop()
print(a)
print('-'*20)
```

```
Pop the element at specified index
[1, 2, 4, 3, 5, 7, 8, 9]
--------------------
Pop the element without specifying index
[1, 2, 4, 3, 5, 7, 8]
--------------------
```

[82]:
```
#6. count(): It counts the no of occurances of the element
print(a.count(7))
print('-'*20)
```

```
1
--------------------
```

[83]:
```
#7. sort(): It sorts the list in ascending order
a.sort()
print(a)
print('-'*20)
```

```
[1, 2, 3, 4, 5, 7, 8]
--------------------
```

[84]:
```
#8. sort(reverse=True): It sorts the list in descending order
a.sort(reverse=True)
print(a)
print('-'*20)
#9. sorted(): It is also sorting method but it requires to store
#in new variable
#ex: b=sorted(a)
```

```
[8, 7, 5, 4, 3, 2, 1]
--------------------
```

[85]:
```
#10. index(): It returns the index of element
print(a.index(7))
print('-'*20)
```

```
1
--------------------
```

[86]:
```
#11. reverse(): It reverses the list
a.reverse()
print(a)
print('-'*20)
```

```
[1, 2, 3, 4, 5, 7, 8]
--------------------
```

[87]:
```
#12. del[start:end]: It is a statement used to
#delete elements from specified indices
del a[1:4]
print(a)
print('-'*20)
```

```
[1, 5, 7, 8]
--------------------
```

[88]:
```
#13. copy(): It copies the list to another variable
b=a.copy()
print('A is:')
print(a)
print('B is:')
print(b)
print('-'*20)
```

```
A is:
[1, 5, 7, 8]
B is:
[1, 5, 7, 8]
--------------------
```

[89]:
```
#14. clear(): It clears all the elements in list
b.clear()
print('A is:')
print(a)
print('B is:')
print(b)
print('-'*20)
```

```
A is:
[1, 5, 7, 8]
B is:
[]
--------------------
```

programs-on-lists

April 10, 2024

[ ]:
```
#Program to reverse a list without using reverse()
a=[1,2,3,4,5] #defining statically
print('Before reversing')
print(a)
a=a[::-1] #It starts accessing from end of the list
print('After reversing')
print(a)
```

```
Before reversing
[1, 2, 3, 4, 5]
After reversing
[5, 4, 3, 2, 1]
```

[ ]:
```
#Program to find minimum and maximum values at a list
#without using max() and min()
def find_max(a):
  max_value=0
  for num in a:
    if num>max_value:
      max_value=num
  return max_value
def find_min(a):
  min_value=a[0]
  for num in a:
    if num<=min_value:
      min_value=num
  return min_value
c=[10,2,3,4,1]
print("The list is:")
print(c)
print("The maximum value is:",find_max(c))
print("The minimum value is:",find_min(c))
```

```
The list is:
[10, 2, 3, 4, 1]
The maximum value is: 10
The minimum value is: 1
```

[ ]:
```
#Program to find maximum and minumum of list
#using max() and min()
d=[2,1,4,3,7,5,9,22,3,1]
print(max(d)) #prints maximum value
print(min(d)) #prints minimum value
```

```
22
1
```

[ ]:
```
#Program to find the sum of elements in list
#without using sum()
def sum_list(a):
  sum=0
  for i in a:
    sum+=i
  return f"The sum of elements of list is {sum}"
d=[1,11,3,4,15,10]
print("The list is")
print(d)
sum_list(d)
```

```
The list is
[1, 11, 3, 4, 15, 10]
```

[ ]:
```
'The sum of elements of list is 44'
```

[ ]:
```
#Program to find sum of elements of list
#with using sum()
print('The list is')
print(d)
print('The sum of elements in list is:',sum(d))
```

```
The list is
[1, 11, 3, 4, 15, 10]
The sum of elements in list is: 44
```

[ ]:
```
#Program to find middle element of the list
#for odd number we can directly take middle value
#for even number we can have 2 middle values
a=[1,2,3,4,5,4,5]
print('The list is')
print(a)
print('The length of list is:',len(a))
if len(a)%2==1:
  print('The middle element is:') #Odd length
  print(a[len(a)//2])
else: #even length
  print('The middle elements are:')
```

```
  print(a[(len(a)-2)//2],a[(len(a))//2])
```

```
The list is
[1, 2, 3, 4, 5, 4, 5]
The length of list is: 7
The middle element is:
4
```

[ ]:
```
#Porgram to add 2 sorted lists
a=[1,2,3,4,5]
b=[2,4,6,8,10]
print('The A list is:')
print(a)
print('The B list is:')
print(b)
c=sorted(a+b)
print('The sorted list of 2 lists is:')
print(c)
```

```
The A list is:
[1, 2, 3, 4, 5]
The B list is:
[2, 4, 6, 8, 10]
The sorted list of 2 lists is:
[1, 2, 2, 3, 4, 4, 5, 6, 8, 10]
```

[ ]:
```
# Program to remove duplicates from the list
#Process 1 (using while loop)
a=[1,2,2,3,4,4,5,6]
print('The original list:')
print(a)
i=0
while i<len(a):
  j=i+1
  while j<len(a):
    if a[i]==a[j]:
      del a[j]
    else:
      j+=1
  i+=1
print('The filtered list:')
print(a)
```

```
The original list:
[1, 2, 2, 3, 4, 4, 5, 6]
The filtered list:
[1, 2, 3, 4, 5, 6]
```

[ ]:
```
#Process 2 (using set())
a=[1,2,2,3,4,4,5,6]
print(list(set(a))) #sets doesn't allow duplicates
```

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

[ ]:
```
#Program to perform intersection of 2 lists
#i.e it takes only common elements in both lists
def intersection(list1,list2):
  common_list=[]
  for items in list1:
    if items in list2 and items not in common_list:
      common_list.append(items)
  return common_list
list1=[1,2,3,4,5]
list_2=[1,2,4,6,7]
print('The intersection of 2 lists:')
intersection(list1,list_2)
```

```
The intersection of 2 lists:
```

[ ]:
```
[1, 2, 4]
```

[ ]:
```
#Program to find union of two lists
def union(list1,list2):
  union_list=list1[:]
  for items in list2:
    if items not in union_list:
      union_list.append(items)
  return union_list
list1=[1,2,3,4,5]
list2=[3,4,5,6,7,8]
print('the union of 2 lists is:')
union(list1,list2)
```

```
the union of 2 lists is:
```

[ ]:
```
[1, 2, 3, 4, 5, 6, 7, 8]
```

[26]:
```
#Porgram to remove an element in a particular index of a list
#without using pop()
def remove_element(a,n):
  if n>len(a):
    print("Index out of range so there is no change in list")
  else:
    for i in range(len(a)):
      if i==n:
        del a[i]
```

```
    print('After removing the element')
  return a
a=[1,2,3,4,5]
n=int(input("Enter index value:"))
print("Before removing element")
print(a)
print("Final list:")
print(remove_element(a,n))
```

```
Enter index value:2
Before removing element
[1, 2, 3, 4, 5]
Final list:
After removing the element
[1, 2, 4, 5]
```

[23]:
```
#Porgram to remove an element based on value in a list
#without using remove()
def remove_value(a,v):
  return [x for x in a if x!=v]
a=[2,3,4,5,7]
v=int(input("Enter remove value:"))
print(remove_value(a,v))
```

```
Enter remove value:3
[2, 4, 5, 7]
```

[33]:
```
#Program to replace the element with other element in the list
a=[]
n=int(input("Enter no of elements:"))
for i in range(n):
  x=int(input("Enter element:"))
  a.append(x)
print("The list is:")
print(a)
new_element=int(input("Enter new element:"))
replaced_element=int(input("Enter replaced element"))
for i in range(n):
  if a[i]==replaced_element:
    a[i]=new_element
print("The updated list:")
print(a)
```

```
Enter no of elements:5
Enter element:2
Enter element:3
Enter element:4
Enter element:4
```

```
Enter element:5
The list is:
[2, 3, 4, 4, 5]
Enter new element:1
Enter replaced element4
The updated list:
[2, 3, 1, 1, 5]
```

[37]:
```
#Program to check whether the given list is palindrome or not
def palindrome(a):
  if a==a[::-1]:
    return "The list is palindrome"
  else:
    return "The list is not a palindrome"
a=[1,2,3,4,1]
palindrome(a)
```

[37]:
```
'The list is not a palindrome'
```

[44]:
```
#Program to find the missing number in the list
#Use case: we find the missing value in a range of elements
#i.e the numbers in sequence order
#It works only for a single missing element
def find_missing_number(a):
  n=len(a)+1
  total_sum=(n*(n+1))//2
  a_sum=sum(a)
  return total_sum-a_sum
a=[1,2,3,4,5,6,7,9]
missing_value=find_missing_number(a)
print("The missing value is:",missing_value)
```

```
The missing value is: 8
```

stacks-and-queues-using-list

April 10, 2024

[5]:
```python
#Program to create a stack
#Stack is an advanced data structure which follows
#LIFO- Last In First Out principle
#Stack has 3 operations:
#1. Push(): It appends the element into stack
#2. Pop(): It removes the newly entered element into stack
#3. peek(): It displays the top-most element in  the stack
#other specifications:
#1. No of elements in stack
#2. Displaying the whole stack
#3. Restriction of appending of elements beyond the stack maximum size
def create_stack():
  stack=[]
  return stack
def is_empty(stack):
  return len(stack)==0
def push(stack,n,m,max):
  m=len(stack)
  if m==max:
    print("Stack is full!!!")
  else:
    stack.append(n)
    print("The pushed item is...",n)
def pop(stack):
  if is_empty(stack):
    print("The stack is empty!!!")
  else:
    print("The Popped item is...",stack.pop())
def peek(stack,m):
  top_element=stack[m-1]
  print(f"The top most element is...{top_element}")
def display(stack):
  print("The stack elements are...")
  for i in stack[::-1]:
    print(i)
a=[]
k=len(a)
```

```
size=int(input("Enter maximum size of stack..."))
while True:
  print("...Stack operations...")
  print('-'*35)
  print("Push <value>")
  print("Pop")
  print("Peek")
  print("Display")
  print("Quit")
  print('-'*35)
  do=input("Which operation do you want to perform...").split()
  operation=do[0].strip().lower()
  if operation=='push':
    push(a,int(do[1]),k,size)
  elif operation=='pop':
    pop(a)
  elif operation=='peek':
    peek(a,k)
  elif operation=='display':
    display(a)
  elif operation=='quit':
    break
  else:
    print("Enter above options")
```

```
Enter maximum size of stack…5
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…pop
The stack is empty!!!
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 1
The pushed item is… 1
The stack operations…
Push <value>
Pop
Peek
Display
```

```
Quit
Which operation do you want to perform…pop
The Popped item is… 1
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 1
The pushed item is… 1
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 2
The pushed item is… 2
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 3
The pushed item is… 3
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…display
The stack elements are…
3
2
1
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…peek
The top most element is…3
The stack operations…
Push <value>
```

```
Pop
Peek
Display
Quit
Which operation do you want to perform…pop
The Popped item is… 3
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…display
The stack elements are…
2
1
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 3
The pushed item is… 3
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 4
The pushed item is… 4
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 5
The pushed item is… 5
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…push 6
Stack is full!!!
```

```
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…display
The stack elements are…
5
4
3
2
1
The stack operations…
Push <value>
Pop
Peek
Display
Quit
Which operation do you want to perform…quit
```

[8]:
```
#Program to create a queue
#Queue is an adavanced data structure which follows
#FIFO- First In First Out principle
#Queue has 4 operations:
#1. Enqueue(): Adds the element into the queue
#2. Dequeue(): Removes the first added element from queue
#3. Rear(): It displays first added element in queue
#4. Front(): It displays newly added element in queue
#Other specifications:
#Maximum size of queue
#Displaying the queue
#Removes the first added element and then adds new element when the;
#size of queue reaches its maximum size
def create_queue():
  queue=[]
  return queue
def is_empty(queue):
  return len(queue)==0
def enqueue(queue,n,m,max):
  m=len(queue)
  if m==max:
    print("The queue is full so we perform dequeue operation and then enqueue")
    queue.pop(0)
    queue.append(n)
    print("The pushed item is...",n)
  else:
```

```
    queue.append(n)
    print("The pushed item is...",n)
def dequeue(queue):
  if is_empty(queue):
    print("The queue is empty")
  else:
    print("The popped element is...",queue.pop(0))
def rear(queue):
  print("The rear element is...",queue[0])
def front(queue,m):
  print("The front element is...",queue[m-1])
def display(queue):
  print("The queue is...")
  for i in queue:
    print(i)
a=[]
k=len(a)
size=int(input("Enter maximum size:"))
while True:
  print("...Queue Operations...")
  print('-'*35)
  print('Enqueue <value>')
  print('Dequeue')
  print('Rear')
  print('Front')
  print('Display')
  print('Quit')
  print('-'*35)
  do=input("Which operation do you want to perform...").split()
  operation=do[0].strip().lower()
  if operation=='enqueue':
    enqueue(a,int(do[1]),k,size)
  elif operation=='dequeue':
    dequeue(a)
  elif operation=='rear':
    rear(a)
  elif operation=='front':
    front(a,k)
  elif operation=='display':
    display(a)
  elif operation=='quit':
    break
  else:
    print("Enter valid operation...")
```

```
Enter maximum size:5
…Queue Operations…
```

```
----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
----------------------------------
Which operation do you want to perform…enqueue 1
The pushed item is… 1
…Queue Operations…
----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
----------------------------------
Which operation do you want to perform…enqueue 2
The pushed item is… 2
…Queue Operations…
----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
----------------------------------
Which operation do you want to perform…enqueue 3
The pushed item is… 3
…Queue Operations…
----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
----------------------------------
Which operation do you want to perform…enqueue 4
The pushed item is… 4
…Queue Operations…
----------------------------------
Enqueue <value>
Dequeue
Rear
```

```
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…enqueue 5
The pushed item is… 5
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…display
The queue is…
1
2
3
4
5
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…enqueue 6
The queue is full so we perform dequeue operation and then enqueue
The pushed item is… 6
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…display
The queue is…
2
3
4
```

```
5
6
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…rear
The rear element is… 2
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…front
The front element is… 6
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…dequeue
The popped element is… 2
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…display
The queue is…
3
4
```

```
5
6
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…rear
The rear element is… 3
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…enqueue 8
The pushed item is… 8
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…display
The queue is…
3
4
5
6
8
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
```

```
-----------------------------------
Which operation do you want to perform…quioo
Enter valid operation…
…Queue Operations…
-----------------------------------
Enqueue <value>
Dequeue
Rear
Front
Display
Quit
-----------------------------------
Which operation do you want to perform…quit
```