Name : A Mallikarjuna Rao
email id: mritcse8@gmail.com
contact no:8074761485
Data Science and Gen AI LLM's
Academic year :2024-25

Assignment -1

Program 1:

```
importrandom
defcomputer_move(current_number):
    max_pick = min(3, 20 - current_number)
    computer_pick=random.randint(1,max_pick)
    move=[iforiinrange(current_number+1,current_number+computer_pick+1)] return
    move
defnumber_game():
    current_number=0
    print("GameStart!")
    whilecurrent_number<20: #
        Player's turn
        try:
            player_input=input(f"Yourturn(choose1,2,or3numbersstartingfrom{current_number+
1}):")
            player_numbers=list(map(int,player_input.split()))
        except ValueError:
            print("Invalidinput.Enternumbersseparatedbyspaces.")
            continue
        iflen(player_numbers)<1orlen(player_numbers)>3:
            print("Invalidinput.Youmustpickbetween1and3numbers.")
            continue
        ifplayer_numbers!=list(range(current_number+1,current_number+1+
len(player_numbers))):
            print(f"Invalidsequence.Numbersshouldbesequentialstartingfrom{current_number+1}.")
            continue
        current_number=player_numbers[-1]
        print(f"You played: {player_numbers}")
        if current_number >= 20:
            print("Youwin!")
```

```
break #Computer'sturn
        computer_numbers=computer_move(current_number)
        current_number = computer_numbers[-1]
        print(f"Computerplayed:{computer_numbers}")
        if current_number >= 20:
            print("Computerwins!")
            break
number_game()
```

Output:

Program 1

```
D:\PyProject>pymallik1.py
Game Start!
Yourturn(choose1,2,or3numbersstartingfrom1):1
Youplayed:[1]
Computerplayed:[2]
Yourturn(choose1,2,or3numbersstartingfrom3):3
Youplayed:[3]
Computerplayed:[4]
Yourturn(choose1,2,or3numbersstartingfrom5):5
Youplayed:[5]
Computerplayed:[6]
Yourturn(choose1,2,or3numbersstartingfrom7):7
Youplayed:[7]
Computerplayed:[8,9]
Yourturn(choose1,2,or3numbersstartingfrom10):
```

Program 2:

```
#FunctiontocalculatenCr def
ncr(n, r):
   # Basecase:ifr==0orr ==n,thevalueis1 if r ==
   0 or r == n:
      return1
   #Recursiveapproach:nCr=(n-1)C(r-1)+(n-1)Cr return
   ncr(n-1, r-1) + ncr(n-1, r)
#FunctiontoprintPascal'striangle
def print_pascals_triangle(rows):
   foriinrange(rows):
      #Printleadingspacesforformatting print(""
      * (rows - i), end="")
      #PrintthevaluesintherowusingnCr for j
      in range(i + 1):
         print(ncr(i,j),end="")
      print()#Movetothenextlineafterprintingeachrow #
Input: number of rows
rows=int(input("EnterthenumberofrowsforPascal'striangle:")) print_pascals_triangle(rows)
```

Output:

Program 2

```
D:\PyProject>pymallik2.py
EnterthenumberofrowsforPascal'striangle:3 1
 11
 121

D:\PyProject>pymallik2.py
EnterthenumberofrowsforPascal'striangle:6 1
```

```
 11
 121
 1331
 14641
15 10 1051
```

```
D:\PyProject>pymallik2.py
EnterthenumberofrowsforPascal'striangle:9 1
 11
 121
 1331
 14641
15 10 1051
1615201561
1721 35 35 217 1
18 2856705628 81
```

Program 3:

```
defprint_repeated_elements_with_count(lst):
    frequency_count={} for
    num in lst:
        ifnuminfrequency_count:
            frequency_count[num]+=1
        else:
            frequency_count[num]=1
    repeated_elements={num:countfornum,countinfrequency_count.items()ifcount>1} if
    repeated_elements:
        print("Repeatedelementswithfrequencycount:") for
        num, count in repeated_elements.items():
            print(f"Element{num}hascome{count}times.")
```

```
    else:
        print("Norepeatedelementsfound.") #
Example usage
input_list=[2,1,2,3,4,5,1,3,6,2,3,4]
print_repeated_elements_with_count(input_list)
```

Output:

Program 3

```
D:\PyProject>pymallik3.py
Repeatedelementswithfrequencycount:
Element2hascome3times.
Element1hascome2times.
Element3hascome3times.
Element4hascome2times.
```

Program 4:

First,let'sassumeyourfile**matrices.txt**containsthematricesinthefollowingformat: (Execute this program in your Jupiter IDE\Jupiter Note Book.)

```
A:
12
34
B:
56
78
```

```
defread_matrix_from_file(file_path):
    with open(file_path, 'r') as file:
        lines= file.readlines()
    A=[]
    B=[]
    matrix = None
    forlineinlines:
        line=line.strip()
```

```
        ifline=="A:":
            matrix=A
        elifline=="B:":
            matrix=B
        else:
            ifmatrixisnot None:
                matrix.append(list(map(int,line.split())))
    return A, B
defadd_matrices(A,B):
    result=[[0,0],[0,0]] for i
    in range(2):
        forjinrange(2):
            result[i][j]=A[i][j]+B[i][j]
    return result
defprint_matrix(matrix):
    forrowinmatrix:
        print("".join(map(str, row)))
if name_____== "main":
    file_path="matrices.txt"
    A,B=read_matrix_from_file(file_path)
    print("Matrix A:")
    print_matrix(A)
    print("\nMatrixB:")
    print_matrix(B)
    result=add_matrices(A,B)
    print("\nResult of A + B:")
    print_matrix(result)
```

Output:
Program 4

A: 1 2 3 4
B: 5 6 7 8

Program 5:

```
frommathimportgcd
class Fraction:
   definit(self,numerator,denominator):
      self.numerator=numerator
      self.denominator=denominator
      self.simplify()
   defadd(self,other):
      ifisinstance(other,Fraction):
         new_numerator = self.numerator * other.denominator + other.numerator * self.denominator
         new_denominator = self.denominator * other.denominator
         return Fraction(new_numerator, new_denominator)
      else:
         raiseTypeError("OperandsmustbeoftypeFraction") def
   simplify(self):
      common_divisor = gcd(self.numerator, self.denominator)
      self.numerator //= common_divisor
      self.denominator//=common_divisor
   def str(self):
      return
f"{self.numerator}/{self.denominator}"# Example
usage
fraction1=Fraction(1,2)
fraction2=Fraction(2,3)
result=fraction1+fraction2
print(f"{fraction1}+{fraction2}={result}")
```

Output:

Program 5

```
D:\PyProject>pymallik5.py
1/2 + 2/3 = 7/6
```