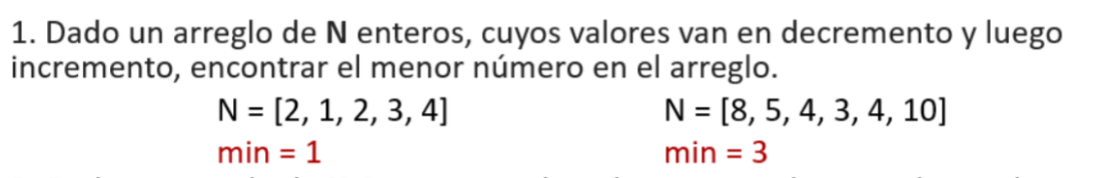
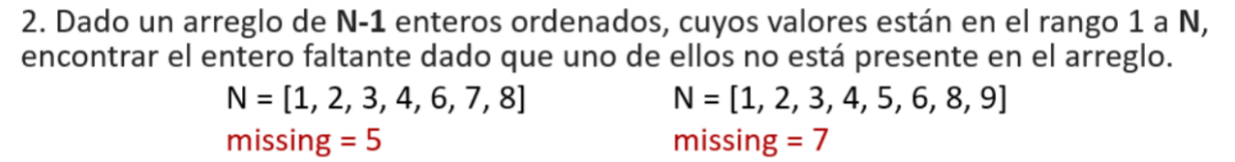
Juan Camilo Bazurto Arias



def lessElement(a):  
 index = 0  
 if len(a) == 1:  
 return a[index]  
 else:  
 if a[index] < a[index + 1]:  
 a[index], a[index + 1] = a[index + 1], a[index]  
 return lessElement(a[index + 1:])  
  
  
N1 = [2, 1, 2, 3, 4]  
print(N1, 'min:', lessElement(N1))  
N2 = [8, 5, 4, 34, 10, 3]  
print(N2, 'min:', lessElement(N2))

Resultados:





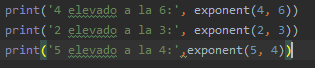
def missingElement(a, i=0):  
 if not a:  
 return 1  
 if i + 1 == a[i]:  
 return missingElement(a, i + 1)  
 else:  
 return i + 1

Resultados:

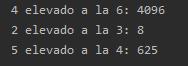




def mult(num, i, num2, j=1):  
 if j == i:  
 return num  
 else:  
 return mult(num \* num2, i, num2, j + 1)  
  
def exponent(num, e):  
 if e == 0:  
 return 1  
 elif e == 1:  
 return num  
 else:  
 halfl, halfr = mult(num, e//2, num), mult(num, (e+1)//2, num)  
 return halfl \* halfr



Resultados:





def merge(s1, s2): #--> O(n) why ?  
 result = []  
 s1\_index, s2\_index = 0,0  
 while s1\_index < len(s1) and s2\_index < len(s2):  
 s1\_element, s2\_element = s1[s1\_index], s2[s2\_index]  
 if s1\_element <= s2\_element:  
 s1\_index += 1  
 result.append(s1\_element)  
 else:  
 s2\_index += 1  
 result.append(s2\_element)  
 # Remaining elements on left-side  
 if s1\_index < len(s1):  
 result += s1[s1\_index:]  
 # Remaining elements on right-side  
 if s2\_index < len(s2):  
 result += s2[s2\_index:]  
 return result  
  
def merge\_sort( s ):  
 # 0 - La secuencia es divisible ?  
 size = len(s)  
 if size <= 1 :  
 return s[:] #--> subsequence from 0 --> len(s)  
 # 1 - Dividir la secuencia en mitades  
 # 2 - Ordenar Las mitades ( n//2 )  
 half\_1,half\_2 = merge\_sort(s[:(size//2)]), merge\_sort(s[(size//2):])  
 # 3 - Mezclar las mitades  
 result = merge(half\_1, half\_2)  
 return ''.join(result)

Resultado:

