<https://colab.research.google.com/drive/16cIXtyJx04x1YdrGLpGk2tB6rArWZytu?usp=sharing#scrollTo=ordR2Gc2bvDC>

1. Sortati numerele dintr-o lista folosind bubble sort. In aceasta sortare parcurgem numerele si daca se afla intr-o pozitie gresita (nr mare in stanga, nr mic in dreapta), le interschimbam. Continuam sa facem aceste parcurgeri cat timp au inca loc interschimbari. Daca intre doua parcurgeri nu au avut loc interschimbari, ne oprim.

def bubblesort(xs):  
 n = len(xs)  
  
 for i in range(n-1):  
 for j in range (0,n-i-1):  
 if xs[j] > xs[j+1]:  
 xs[j], xs[j+1] = xs[j+1], xs[j]  
  
xs = [1,5,2,7,4,9,10,0,8]  
bubblesort(xs)  
for i in range(len(xs)):  
 print(xs[i])

1. Sortati numerele dintr-o lista folosind insertion sort. Presupunem ca primele k elemente sunt sortate. Consideram elementul k+1 si il inseram la locul potrivit printre primele k elemente (mutand celelalte elemente la dreapta).

Exemplu (la pasul k = 3):

4 8 9 5 1 2 3    # 4 8 9 sunt sortate, consideram elementul 5  
4 5 8 9 1 2 3    # l-am inserat pe 5 la locul potrivit, continuam cu 1

def insertionsort(xs):  
 for i in range(1, len(xs)):  
 x = xs[i]  
 j = i-1  
 while j >= 0 and x < xs[j]:  
 xs[j+1] = xs[j]  
 j -= 1  
 xs[j+1] = x  
  
xs = [4, 8, 9, 5, 1, 2, 3]  
insertionsort(xs)  
for i in range(len(xs)):  
 print(xs[i])

1. Sortati numerele dintr-o lista folosind selection sort folosind selectia minimului. Parcurgem vectorul, gasim minimul si il punem pe prima pozitie. Cautam urmatorul minim (intre 1 si n-1) il punem pe a doua pozitie s.a.m.d.

def selectionsortmin(xs):  
 i = 0  
 for i in range(len(xs)):  
 pozmin = i  
 for j in range(i+1, len(xs)):  
 if xs[j] < xs[pozmin]:  
 pozmin = j  
 xs[i], xs[pozmin] = xs[pozmin], xs[i]  
  
xs = [1, 5, 2, 7, 4, 9, 10, 0, 8]  
selectionsortmin(xs)  
for i in range(len(xs)):  
 print(xs[i])

1. Sortati numerele dintr-o lista folosind merge sort. Trebuie sa scrieti algoritmul de interclasare care primeste doua liste sortate crescator si returneaza o lista sortata care contine toate elementele celor doua liste.

Aplicati apoi algoritmul merge sort: Cat timp vectorul are o lungime mai mare decat 1, impartim vectorul in doua parti egale (sau aproximativ egale), folosim merge\_sort pe cele doua parti, dupa cele doua apeluri avem doi vectori sortati, aplicam interclasarea si returnam vectorul rezultat.

def mergeSort(xs):  
 if len(xs) > 1:  
 mij = len(xs) // 2  
  
 L = xs[:mij]  
 R = xs[mij:]  
  
 mergeSort(L)  
 mergeSort(R)  
  
 i = j = k = 0  
  
 while i < len(L) and j < len(R):  
 if L[i] < R[j]:  
 xs[k] = L[i]  
 i += 1  
 else:  
 xs[k] = R[j]  
 j += 1  
 k += 1  
  
 while i < len(L):  
 xs[k] = L[i]  
 i += 1  
 k += 1  
  
 while j < len(R):  
 xs[k] = R[j]  
 j += 1  
 k += 1  
  
xs = [1, 5, 2, 7, 4, 9, 10, 0, 8]  
  
mergeSort(xs)  
  
for i in range (len(xs)):  
 print(xs[i], end=" ")  
print()

1. Sortati numerele dintr-o lista folosind quick sort. Trebuie sa alegeti un pivot (de exemplu primul element), sa gasiti numerele mai mici, egale si mai mari decat pivotul, si returnati quick\_sort(mai\_mici) + egale + quick\_sort(mai\_mari). Puteti folosi list comprehension daca vreti ([x for x in xs if x < p]).

def partition(xs, low, high):  
 i = (low-1)  
 pivot = xs[high]  
  
 for j in range(low, high):  
 if xs[j] <= pivot:  
 i = i+1  
 xs[i], xs[j] = xs[j], xs[i]  
 xs[i+1], xs[high] = xs[high], xs[i+1]  
 return (i+1)  
  
def quick\_sort(xs, low, high):  
 if len(xs) == 1:  
 return xs  
  
 if low < high:  
 pi = partition (xs, low, high)  
 quick\_sort(xs, low, pi-1)  
 quick\_sort(xs, pi+1, high)  
  
  
xs = [1, 5, 2, 7, 4, 9, 10, 0, 8]  
n = len(xs)  
quick\_sort(xs, 0, n-1)  
  
for i in range(n):  
 print(xs[i], end=" ")

15. Tot problema 13.

In loc sa introduceti sumele intr-o lista si sa o sortati, introduceti-le intr-un set/dict. Faceti cautarea direct cu operatorul "in".

m = 8  
xs = [1, 5, 2, 3, 10, 4]  
  
d = {}  
# O (n^2)  
for x in xs:  
 for y in xs:  
 d[x + y] = (x, y)  
  
# O(n^2)  
for k in d:  
 if m - k in d:  
 print(\*d[k], \*d[m - k])  
 break