

```
# -*- coding: utf-8 -*-  
"""
```

```
Created on Fri Oct 1 01:10:40 2021
```

```
@author: Guillermo Arredondo  
Abraham Guerrero  
Luisa Saloma  
"""
```

```
"""El método clásico de ordenamiento que consta de comparar cada dato con respecto al de la casilla a ocupar,  
así, se busca el dato menor a partir de la posición que se desea rellenar y hacia el final, una vez encontrado, se sustituye  
por la posición actual"""
```

```
def seleccionDirecta(A):  
    for i in range(len(A)):  
        minimo = i  
        for j in range(i+1, len(A)):  
            if (A[j] < A[minimo]):  
                minimo = j  
        if (minimo != i):  
            aux = A[i]  
            A[i] = A[minimo]  
            A[minimo] = aux
```

```
"""El método de inserción directa, similar al caso de selecciónDirecta, compara los datos con respecto a los anteriores, colocandol  
en su posición ordenada mediante intercambio de datos 2 a 2"""
```

```
def insercionDirecta(A):  
    for i in range(len(A) - 1):  
        limite = i + 1  
        while (limite > 0 and A[limite] <= A[limite - 1]):  
            A[limite], A[limite - 1] = A[limite - 1], A[limite] #esto es una asignación de variable por comas.  
            limite = limite - 1
```

```
"""Se generan dos subcolecciones, una desde el inicio hasta el pivote (calculado en la mitad del arreglo)  
y otro del pivote en adelante se hace uso de llamadas recursivas para volver a dividirlo hasta que la longitud sea 1  
Nótese que en el caso de ser una colección vacía saldrá del método inmediatamente"""
```

```
def combinacionSubColecc(A):  
    if (len(A) > 1):  
        pivote = len(A) // 2  
        """se divide (virtualmente) el arreglo en una
```

parte izquierda y una parte derecha"""

```
sub1 = A[:pivote]
sub2 = A[pivote:]
combinacionSubColecc(sub1) #A[0:pivote]
combinacionSubColecc(sub2) #A[pivote:max]
mezcla(A, sub1, sub2)
```

"""El método mezcla recibe como parámetros la colección original, y las dos subcolecciones que se desean mezclar de esta forma, no es necesario regenerar otra colección auxiliar, pues se rellenan los datos dentro de la misma colección original."""

```
def mezcla(original, A, B):
    i = j = k = 0
    while (i < len(A)):
        if (j >= len(B) or A[i] < B[j]):
            original[k] = A[i]
            i += 1
        else:
            original[k] = B[j]
            j += 1
        k += 1
    while (j < len(B)):
        original[k] = B[j]
        j += 1
        k += 1
```