



## ALGORITMOS Y ESTRUCTURAS DE DATOS (TSDS)

ASIGNATURA:

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROFESOR:

Ing. Lorena Chulde

FECHA:

20/06/2025

PERÍODO ACADÉMICO:

2025-A

### TALLER

(Individual)

## TÍTULO: TUPLAS Y DICCIONARIOS ALGORITMOS

Nombre del estudiante

Claudia Coello

```
>>> a_tuple = ('value1', 'value2', 'value3')  
>>> var1, var2, var3 = a_tuple  
También es una tupla
```



# PROPÓSITO DE LA TAREA

- Recorrer tuplas mediante estructuras cíclicas para leer sus elementos y visualizar sus valores en pantalla.
- Desarrollar algoritmos usando diccionarios
- Aplicar diccionario mediante funciones para implementar un CRUD

## Parte 1: Tuplas

### 1. Iterar una tupla

```
tupla = 1, 2, 3
```

```
print("Claudia Coello")  
lista = [1,2,3]  
tupla = tuple(lista)
```

```
for i in tupla:  
    print(i)
```

```
Claudia Coello  
1  
2  
3
```

### 2. Anidar tuplas. Acceder al elemento 'a'

```
tupla = 1, 2, ('a', 'b'), 3
```

```
print("Claudia Coello")  
tupla = 1, 2, ('a', 'b'), 3  
elemento = tupla[2][0]  
print(elemento)
```

```
Claudia Coello  
a
```

### 3. Asignar el valor de una tupla con n elementos a n variables.

```
tupla = (1, 2, 3)
```

```
print("Claudia Coello")  
tupla = (1, 2, 3)  
a,b,c = tupla  
print(a)  
print(b)  
print(c)
```

```
Claudia Coello
1
2
3
```

## Métodos tuplas

4. **count(<obj>):** Contar el numero de veces que se repite un elemento en la tupla

```
tupla = (1, 1, 1, 3, 5)
```

```
print("Claudia Coello")
```

```
tupla = (1, 1, 1, 3, 5)
```

```
print(tupla.count(1))
```

```
Claudia Coello
3
```

5. Verificar la posición de un elemento:

**index(<obj>[,index]):** permite buscar el **índice** del primer elemento que coincida con un valor específico, a partir de una posición determinada.

Con un parámetro

```
tupla = (7, 7, 7, 3, 5)
```

```
print("Claudia Coello")
```

```
tupla = (7, 7, 7, 3, 5)
```

```
print(tupla.index(5))
```

```
Claudia Coello
4
```

6. Con dos parámetros

```
tupla = (7, 7, 7, 3, 5)
```

```
print("Claudia Coello")
```

```
tupla = (7, 7, 7, 3, 5)
```

```
print(tupla.index(7,2))
```

```
Claudia Coello
2
```

7. Imprimir el apellido y el nombre

```
nombres = ("Lorena", "Chulde")
```

```
print("Claudia Coello")
nombres = ("Claudia", "Coello")
nombre, apellido = nombres
print(f'{nombre}, {apellido}')
```

```
Claudia Coello
Claudia, Coello
```

#### 8. Imprimir el orden de llegada de los atletas (enumerate)

```
atletas = ("Lorena Chulde", "Juan Perez", "Maria Mera", "Pedro Robayo")
```

```
print("Claudia Coello")
atletas = ("Lorena Chulde", "Juan Perez", "Maria Mera", "Pedro Robayo")

for atleta in enumerate(atletas, + 1):
    print("Orden", atleta[0], "atleta:", atleta[1])
```

```
Claudia, Coello
(0, 'Lorena Chulde')
(1, 'Juan Perez')
(2, 'Maria Mera')
(3, 'Pedro Robayo')
```

#### 9. Verificar que es un objeto enumerate

```
atletas = ("Lorena Chulde", "Juan Perez", "Maria Mera", "Pedro Robayo")
posicion = enumerate(atletas)
print(posicion)
```

Salida:

```
<enumerate object at 0x000001799C055EE0>
```

#### 10. Crea una tupla con números, pide un numero por teclado e indica cuantas veces se repite. Usando `count`

```
print("Claudia Coello")
numeros = (5,4,3,2,1,6,45,6,7,8,9,2,6,3,2,1,6,7)
numero = int(input("Ingrese un numero: "))
print(numeros.count(numero))
```

```
Claudia Coello
Ingrese un numero: 5
1
```

11. Crea una tupla con números e indica el numero con mayor valor y el que menor tenga. Usando max, min

```
print("Claudia Coello")
numeros = (5,4,3,2,1,6,45,6,7,8,9,2,6,3,2,1,6,7)
print(max(numeros))
print(min(numeros))
```

```
Claudia Coello
45
1
```

```
print("Claudia Coello")
numeros = (5,4,3,2,1,6,45,6,7,8,9,2,6,3,2,1,6,7)
numMayor = numeros[0]
numMenor = numeros[0]
for i in range(1, len(numeros)):
    if numeros[i] > numMayor:
        numMayor = numeros[i]

for i in range(1, len(numeros)):
    if numeros[i] < numMenor:
        numMenor = numeros[i]
print("El mayor es: ", numMayor)
print("El menor es: ", numMenor)
```

```
Claudia Coello
El mayor es: 45
El menor es: 1
```

12. Desarrolle un algoritmo que cree una tupla  
13. Crea una tupla con valores ya predefinidos del 1 al 10, pide un índice por teclado y muestra el valor de la tupla.

```
print("Claudia Coello")
tupla = (1,2,3,4,5,6,7,8,9,10)
indice = int(input("Ingrese un indice: "))
print(tupla[indice])
```

```
Ingrese un indice: 4
5
```

14. Convertir una lista en tupla haciendo uso de la función tuple().

```
lista = [6, 7, 8, 9, 10]
```

```
print("Claudia Coello")
```

```
lista = [6, 7, 8, 9, 10]
listaTupla = tuple(lista)
print(listaTupla)
```

```
Claudia Coello
(6, 7, 8, 9, 10)
```

15. Crea una tupla con los meses del año, pide números al usuario, si el numero esta entre 1 y la longitud máxima de la tupla, muestra el contenido de esa posición sino muestra un mensaje de error.

El programa termina cuando el usuario introduce un cero.

```
print("Claudia Coello")
meses_del_year = ('Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto',
'Septiembre', 'Octubre', 'Noviembre', 'Diciembre')
def mostrarMeses():
    numero = 0
    numeroMeses = len(meses_del_year)
    while True:
        try:
            numero = int(input(f"Ingrese un numero entre 1 y {numeroMeses} o 0 para
salir: "))

            if numero > len(meses_del_year) or numero < 0:
                raise ValueError(f"Error ingrese un numero entre 1 y {numeroMeses} o 0
para salir: ")

            if numero == 0:
                break

            print("Mes: ",meses_del_year[numero - 1])

        except Exception as e:
            print("Error:", e)
```

```
mostrarMeses()
```

```
cla@cla-inspiron-3542:~/Algoritmos/Semana 10$ ./usr/bin/pyt
Ingrese un numero entre 1 y 12 o 0 para salir: 15
Error: Error ingrese un numero entre 1 y 12 o 0 para salir
Ingrese un numero entre 1 y 12 o 0 para salir: 4
Mes: Abril
Ingrese un numero entre 1 y 12 o 0 para salir: -1
Error: Error ingrese un numero entre 1 y 12 o 0 para salir
Ingrese un numero entre 1 y 12 o 0 para salir: 0
cla@cla-inspiron-3542:~/Algoritmos/Semana 10$ ./usr/bin/pyt
```

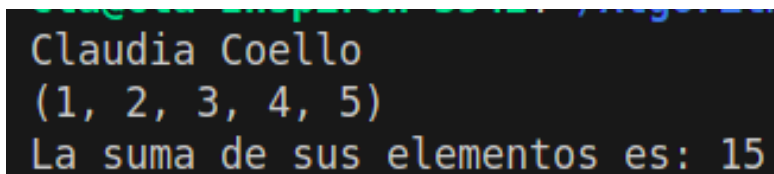
16. Crea una función que reciba una tupla de números y devuelva la suma de todos sus elementos.

```
print("Claudia Coello")
tupla = (1,2,3,4,5)

def sumaElementosTupla(tupla):
    suma = 0
    for i in tupla:
        suma += i

    return suma

print(tupla)
print(f'La suma de sus elementos es: {sumaElementosTupla(tupla)}')
```



```
Claudia Coello
(1, 2, 3, 4, 5)
La suma de sus elementos es: 15
```

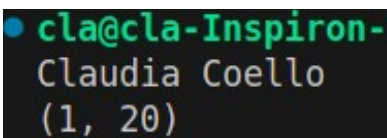
17. Crea una función que reciba una tupla de números y devuelva una nueva tupla con el valor mínimo y máximo.

```
print("Claudia Coello")
tupla = (10,1,20,13,4,5)

def minMaxTupla(tupla):
    tuplaMinMax = (min(tupla), max(tupla))

    return tuplaMinMax

tuplaMinMax = minMaxTupla(tupla)
print(tuplaMinMax)
```



```
• cla@cla-Inspiron-
Claudia Coello
(1, 20)
```

18. Crea una función que reciba una tupla y un valor, y retorne cuántas veces aparece ese valor en la tupla.

```
def aparicionesEnTupla(tupla, numero):
    contador = 0
    for i in tupla:
```

```

if i == numero:
    contador += 1
return contador

```

```

tupla = (1,1,2,5,8,5,8)
numeroApariciones = aparicionesEnTupla(tupla, int(input("Ingrese un numero
para contar la cntidad de veces que aparece: ")))
print(tupla)
print(f"El numero de apariciones en la tupla es: {numeroApariciones}")

```

```

cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$ /usr/bin/python3 "/"
Ingrese un numero para contar la cntidad de veces que aparece: 1
(1, 1, 2, 5, 8, 5, 8)
El numero de apariciones en la tupla es: 2
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$

```

## Parte 2: Diccionarios

1. Declarar el diccionario con clave y valor de los países y capitales de América Latina

```

print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
print(países)

```

```

Claudia Coello
{'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota'}

```

2. Acceder a un valor del diccionario de países y capitales.

```

print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
print(países["Ecuador"])

```

```

Claudia Coello
Quito

```

3. Agregar un elemento al diccionario de países y capitales.

```

print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
países["Bolivia"] = "La paz"
print(países)

```

```

Claudia Coello
{'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota', 'Bolivia': 'La paz'}

```



#### 4. Modificar un elemento del diccionario de países y capitales.

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
países["Bolivia"] = "Montevideo"
print(países)
```

```
Claudia Coello
{'Ecuador': 'Quito', 'Peru': 'Lima', 'Colombia': 'Bogota', 'Bolivia': 'Montevideo'}
```

#### 5. Eliminar un elemento del diccionario de países y capitales.

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
del países["Peru"]
print(países)
```

```
Claudia Coello
{'Ecuador': 'Quito', 'Colombia': 'Bogota'}
```

#### 6. Eliminar un elemento con la función pop

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
países.pop("Colombia")
print(países)
```

```
Claudia Coello
{'Ecuador': 'Quito', 'Peru': 'Lima'}
```

#### 7. Acceder un elemento en concreto

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
print(países["Peru"])
```

```
Claudia Coello
Lima
```

#### 8. Crear diccionarios con diferentes tipos de datos

```
print("Claudia Coello")
persona = {"Nombre" : "Claudia", "Apellido" : "Coello", "edad":21,
"Pais":"Ecuador"}
for elemento in persona:
    print(elemento)
for elemento in persona.values():
    print(elemento)

for elemento in persona.items():
    print(elemento)
```

```
Claudia Coello
Nombre
Apellido
edad
Pais
Claudia
Coello
21
Ecuador
('Nombre', 'Claudia')
('Apellido', 'Coello')
('edad', 21)
('Pais', 'Ecuador')
```

## 9. Con valores de tipo lista

```
print("Claudia Coello")
semestre = {"materias" : ["Algoritmos", "Redes", "Arquitectura"], "notas" :
[10,9,8],"pasatiempos" : ["bailar","pintar","leer"]}
print(semestre["pasatiempos"][1])
```

```
Claudia Coello
pintar
```

## 10. Un diccionario puede contener otro diccionario

```
print("Claudia Coello")
diccionario = {'diccionario anidado':{'saludo':'hola'}, 'diccionario
normal':'hola'}
print(diccionario)
print(diccionario['diccionario anidado'])
```

```
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$ /usr/bin/python3 "/home/cla/
Claudia Coello
{'diccionario anidado': {'saludo': 'hola'}, 'diccionario normal': 'hola'}
{'saludo': 'hola'}
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
```

### 11. Consultar las claves del diccionario

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
claves = países.keys()
print('Las claves son : ', claves)

for i in países:
    print('Otro metodo es: ', i)
```

```
Claudia Coello
Las claves son : dict_keys(['Ecuador', 'Peru', 'Colombia'])
Otro metodo es: Ecuador
Otro metodo es: Peru
Otro metodo es: Colombia
```

### 12. Consultar los valores del diccionario

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
valores = países.values()
print('Las valores son : ', valores)
```

```
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$ /usr/bin/python
Claudia Coello
Las valores son : dict_values(['Quito', 'Lima', 'Bogota'])
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
```

### 13. Consultar la longitud del diccionario

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
longitud = len(países)
print('La longitud del diccionario es: ', longitud)
```

```
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
Claudia Coello
La longitud del diccionario es: 3
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
```

### 14. Recorrer el diccionario con for e imprimir claves

```
print("Claudia Coello")
países = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}
```

```
for i in paises.keys():
    print(i)
```

```
cla@cla-Inspiron
Claudia Coello
Ecuador
Peru
Colombia
cla@cla-Inspiron
```

### 15. Recorrer el diccionario con for e imprimir clave y valor

```
print("Claudia Coello")
paises = {"Ecuador": "Quito", "Peru": "Lima", "Colombia": "Bogota"}

for i in paises.items():
    print(i)
```

```
cla@cla-Inspiron-3542:~/
Claudia Coello
('Ecuador', 'Quito')
('Peru', 'Lima')
('Colombia', 'Bogota')
cla@cla-Inspiron-3542:~/
```

### 16. Escribir un programa que guarde en una variable el diccionario {'Euro':'€', 'Dollar':'\$', 'Yen':'¥'}, pregunte al usuario por una divisa y muestre su símbolo o un mensaje de aviso si la divisa no está en el diccionario.

```
print("Claudia Coello")
monedas = {'Euro':'€', 'Dollar':'$', 'Yen':'¥'}
divisa = input('Ingrese una divisa: ')

if divisa in monedas:
    print('Su símbolo es:', monedas[divisa])
else:
    print('La divisa no está en el diccionario')
```

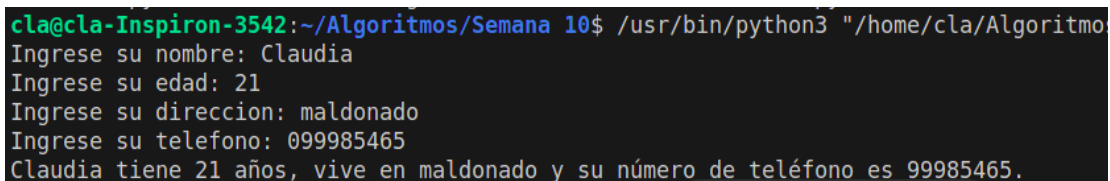
```
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
Claudia Coello
Ingrese una divisa: Dollar
Su símbolo es: $
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
Claudia Coello
Ingrese una divisa: sucre
La divisa no está en el diccionario
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$
```

17. Escribir un programa que pregunte al usuario su nombre, edad, dirección y teléfono y lo guarde en un diccionario. Después debe mostrar por pantalla el mensaje <nombre> tiene <edad> años, vive en <dirección> y su número de teléfono es <teléfono>.

```
nombre = input("Ingrese su nombre: ")
edad = int(input("Ingrese su edad: "))
direccion = input("Ingrese su direccion: ")
telefono = int(input("Ingrese su telefono: "))

persona = {'nombre' : nombre, 'edad' : edad, 'direccion' : direccion, 'telefono' :
telefono}

print(f'{nombre} tiene {edad} años, vive en {direccion} y su número de teléfono es
{telefono}.')
```



```
cla@cla-Inspiron-3542:~/Algoritmos/Semana 10$ /usr/bin/python3 "/home/cla/Algoritmo
Ingrese su nombre: Claudia
Ingrese su edad: 21
Ingrese su direccion: maldonado
Ingrese su telefono: 099985465
Claudia tiene 21 años, vive en maldonado y su número de teléfono es 99985465.
```

## CRUD CON DICCIONARIOS

18. Diccionario que permite gestionar notas  
Realiza lo siguiente:

- Crear una función para añadir las notas (agregarNota), con los parámetros: nota, título, contenido.
- Crear una función para ver las notas (verNotas)
- Crear una función para editar las notas (editarNota)
- Crear una función para eliminar las notas (eliminarNota)
- Crear una función que muestre el menú

## PARTE 3: Algoritmos

### Bubble Sort (Burbuja)

Crear una fn para recorrer una lista.

**Método burbuja:** Crear una función para ordenar elementos de un array de menor a mayor.

Cree una función llamada ordenar()

```
def ordenar(array):
    tamaño = len(array)
    #for para recorrer las posiciones de 0 a 5
    for i in range(0,tamaño-1):
        #con este bucle comparamos
        for j in range(0,tamaño-1):
            if array[j] > array[j+1]:
                aux = array[j]
                array[j] = array[j+1]
                array[j+1] = aux
        return array

numeros = [6,3,8,2,7]
print("la lista original es: ", numeros)
#ordenar(numeros)
print("la lista ordenada es: ", ordenar(numeros))
```

```
def ordenar(lista):
    size = len(lista)
    for i in range(0,size - 1):
        for j in range(0, size - 1):
            if lista[j] > lista[j + 1]:
                aux = lista[j]
                lista[j] = lista[j+1]
                lista[j+1] = aux
    return lista

numero = [6,3,8,2,7]
print("La lista original es: ", numero)
print("La lista ordenada es: ",ordenar(numero))
```

```
cla@cla-Inspiron-3542:~/Algoritmos/Sema
La lista original es:  [6, 3, 8, 2, 7]
La lista ordenada es:  [2, 3, 6, 7, 8]
```

## Selection Sort (Selección)

Analizar y explicar paso a paso en qué consiste el Algoritmo de ordenamiento por Selección.

Realizar la prueba de escritorio.

```
algoritmos-ordenamiento > seleccion-sort.py > ...
1  #Declara la función (Lorena Chulde)
2  def seleccion_sort(lista):
3      for i in range(len(lista)):
4          minimo=i;
5          for j in range(i,len(lista)):
6              if(lista[j] < lista[minimo]):
7                  minimo=j;
8              if(minimo != i):
9                  aux=lista[i];
10                 lista[i]=lista[minimo];
11                 lista[minimo]=aux;
12         print (lista);
13
14 #Programa Principal
15 lista =[6,5,3,1,8,7,2,4];
16 print("Antes")
17 print (lista)
18 print("Después")
19 seleccion_sort(lista);
```

#Divide la funcion en dos, la parte ordenada y la desordenada, busca el elemento mas pequeño en esta ultima y lo coloca al final de la parte ordenada, asi va construyendo una #lista ordenada

```
def seleccion_sort(lista):
    for i in range(len(lista)):#Recorre toda la lista
        minimo = i#Supone que el primer elemento es el menor
        for j in range(i,len(lista)):#Recorre la lista desordenada buscando el menor
            if(lista[j] < lista[minimo]):#Si encuentra un numero menor guarda su posicion
                minimo = j
        if(minimo != i):#Si el actual no es el minimo, intercambian
            aux = lista[i]
            lista[i] = lista[minimo]
            lista[minimo] = aux
    print(lista)
```

```
lista = [6,5,3,1,8,7,2,4]
print("Antes")
print(lista)
print("Despues")
seleccion_sort(lista)
```

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

1	<p>Al inicio  lista = [6,5,3,1,8,7,2,4]  Al final  lista = [1,5,3,6,8,7,2,4]</p>	<pre> i = 0 minimo = 0 j = 0 lista[0] = 6 lista[0] = 6 ----- i = 0 minimo = 0 j = 1 lista[1] = 5 lista[0] = 6 minimo = 1 ----- i = 0 minimo = 1 j = 2 lista[2] = 3 lista[1] = 5 minimo = 2 ----- i = 0 minimo = 2 j = 3 lista[3] = 1 lista[2] = 3 minimo = 3 ----- i = 0 minimo = 3 j = 4 lista[4] = 8 lista[3] = 1 ----- i = 0 minimo = 3 j = 5 lista[5] = 7 lista[3] = 1 ----- i = 0 minimo = 3 j = 6 lista[6] = 2 lista[3] = 1 ----- i = 0 minimo = 3 j = 7 lista[7] = 4 </pre>	<pre> minimo=3 i = 0 aux = 6 lista[0] = 1 lista[3] = 6 </pre>
---	--	--	---



		lista[3] = 1	
2	Al inicio lista = [1,5,3,6,8,7,2,4] Al final lista = [1,2,3,6,8,7,5,4]	<pre> i = 1 minimo = 1 j = 1 lista[1] = 5 lista[1] = 5 ----- i = 1 minimo = 1 j = 2 lista[2] = 3 lista[1] = 5 minimo = 2 ----- i = 1 minimo = 2 j = 3 lista[3] = 6 lista[2] = 3 ----- i = 1 minimo = 2 j = 4 lista[4] = 8 lista[2] = 3 ----- i = 1 minimo = 2 j = 5 lista[5] = 7 lista[2] = 3 ----- i = 1 minimo = 2 j = 6 lista[6] = 2 lista[2] = 3 minimo = 6 ----- i = 1 minimo = 6 j = 7 lista[7] = 4 lista[6] = 2 </pre>	minimo=6 i = 1 aux = 5 lista[1] = 2 lista[6] = 5
3	Al inicio lista = [1,2,3,6,8,7,5,4] Al final lista = [1,2,3,6,8,7,5,4]	<pre> i = 2 minimo = 2 j = 2 lista[2] = 3 lista[2] = 3 ----- </pre>	

		<pre> i = 2 minimo = 2 j = 3 lista[3] = 6 lista[2] = 3 ----- i = 2 minimo = 2 j = 4 lista[4] = 8 lista[2] = 3 ----- i = 2 minimo = 2 j = 5 lista[5] = 7 lista[2] = 3 ----- i = 2 minimo = 2 j = 6 lista[6] = 5 lista[2] = 3 ----- i = 2 minimo = 2 j = 7 lista[7] = 4 lista[2] = 3 </pre>	
4	<p>Al inicio  lista = [1,2,3,6,8,7,5,4]  Al final  lista = [1,2,3,4,8,7,5,6]</p>	<pre> i = 3 minimo = 3 j = 3 lista[3] = 6 lista[3] = 6 ----- i = 3 minimo = 3 j = 4 lista[4] = 8 lista[3] = 6 ----- i = 3 minimo = 3 j = 5 lista[5] = 7 lista[3] = 6 ----- i = 3 minimo = 3 j = 6 </pre>	<pre> i = 3 aux = 6 lista[3] = 4 lista[7] = 6 </pre>

		lista[6] = 5 lista[7] = 6 minimo = 6 ----- i = 3 minimo = 6 j = 7 lista[7] = 4 lista[6] = 5 minimo = 7	
5	Al inicio lista = [1,2,3,4,8,7,5,6] Al final lista = [1,2,3,4,5,7,8,6]	i = 4 minimo = 4 j = 4 lista[4] = 8 lista[4] = 8 ----- i = 4 minimo = 4 j = 5 lista[5] = 7 lista[4] = 8 minimo = 5 ----- i = 4 minimo = 5 j = 6 lista[6] = 5 lista[5] = 7 minimo = 6 ----- i = 4 minimo = 6 j = 7 lista[7] = 6 lista[6] = 5 minimo = 6	minimo = 6 i = 4 aux = 8 lista[4] = 5 lista[6] = 8
6	Al inicio lista = [1,2,3,4,5,7,8,6] Al final lista = [1,2,3,4,5,6,8,7]	i = 5 minimo = 5 j = 5 lista[5] = 7 lista[5] = 7 ----- i = 5 minimo = 5 j = 6 lista[6] = 8 lista[5] = 7 minimo = 5 ----- i = 5	if(minimo != i):#Si el actual no es el minimo, intercambian aux = lista[i] lista[i] = lista[minimo] lista[minimo] = aux minimo = 7 i = 5 aux = 7 lista[5] = 6 lista[7] = 7

		minimo = 5 j = 7 lista[7] = 6 lista[5] = 7 minimo = 7	
7	Al inicio lista = [1,2,3,4,5,6,8,7] Al inicio lista = [1,2,3,4,5,6,7,8]	i = 6 minimo = 6 j = 6 lista[6] = 8 lista[6] = 8 ----- i = 6 minimo = 6 j = 7 lista[7] = 7 lista[6] = 8 minimo = 7	if(minimo != i):#Si el actual no es el minimo, intercambian aux = lista[i] lista[i] = lista[minimo] lista[minimo] = aux minimo = 7 i = 6 aux = 8 lista[6] = 7 lista[7] = 8

## Insertion Sort (Inserción)

- Recorre la lista de izquierda a derecha.
- En cada paso, **toma un elemento y lo inserta** en la posición correcta dentro de la parte ya ordenada de la lista.

```
def insertion_sort(lista):
    for i in range(1, len(lista)):
        valor_actual = lista[i]
        posicion = i - 1
        # Mover los elementos mayores a la derecha
        while posicion >= 0 and lista[posicion] > valor_actual:
            lista[posicion + 1] = lista[posicion]
            posicion -= 1
        # Insertar el valor actual en su posición correcta
        lista[posicion + 1] = valor_actual
    return lista
```

```
numeros = [8, 4, 2, 9, 5]
print('Lista desordenada: ', numeros)
ordenados = insertion_sort(numeros)
print("Lista ordenada:", ordenados)
```

```
def insertion_sort(lista):
    for i in range(1, len(lista)):# Empieza desde el segundo elemento (índice 1)
        valorActual = lista[i]# Guarda el valor actual para compararlo
        posicion = i - 1# Posición del elemento anterior (parte ordenada)

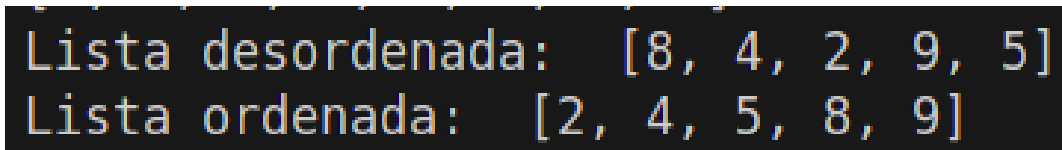
        while posicion >= 0 and lista[posicion] > valorActual:# Mueve los elementos mayores que
`valorActual` una posición adelante
            lista[posicion + 1] = lista[posicion] # Desplaza el elemento mayor
```

```
    posicion -= 1                # Retrocede para comparar con el anterior

    lista[posicion + 1] = valorActual # Inserta `valorActual` en su posición correcta

return lista

numeros = [8,4,2,9,5]
print('Lista desordenada: ',numeros)
ordenados = insertion_sort(numeros)
print('Lista ordenada: ',ordenados)
```



```
Lista desordenada: [8, 4, 2, 9, 5]
Lista ordenada: [2, 4, 5, 8, 9]
```

## ENTREGABLES:

- Una vez culminada tu tarea, capturar las pantallas de la ejecución de cada ejercicio con tus datos y súbela en el apartado del aula virtual.
- Subir los ejercicios al git o al drive y entrega la url de los archivos .py o, a su vez, entregue el archivo.
- Recordar que el nombre del archivo deberá ser: S10\_Taller\_ApellidoNombre

## RECURSOS NECESARIOS

- Acceso a Internet.
- Imaginación.
- VSC