Unidad 5:

- Listas
- Tuplas
- Diccionarios

Listas(`list`)

Las listas son estructuras de datos que pueden almacenar cualquier otro tipo de dato, inclusive una lista puede contener otra lista, además, la cantidad de elementos de una lista se puede modificar removiendo o añadiendo elementos. Para definir una lista se utilizan los corchetes, dentro de estos se colocan todos los elementos separados por comas:

```
calificaciones = [10,9,8,7.5,9]
nombres = ["Ana","Juan","Sofía","Pablo","Tania"]
mezcla = [True, 10.5, "abc", [0,1,1]]
```

Las listas son *iterables* y por tanto se puede acceder a sus elementos mediante indexación:

```
nombres[2]
```

```
'Sofía'
```

```
nombres[-1]
```

```
'Tania'
```

Se tiene la posibilidad de agregar elementos a una lista mediante el método append:

```
nombres.append("Antonio")
nombres.append("Ximena")
print(nombres)
```

```
uan', 'Sofía', 'Pablo', 'Tania', 'Antonio', 'Ximena']
```

El método remove elimina un elemento de una lista:

```
nombres.remove("Ana")
print(nombres)
```

```
['Juan', 'Sofía', 'Pablo', 'Tania', 'Antonio', 'Ximer
```

Tuplas(tuple)

```
colores=("Azul","Verde","Rojo","Amarillo","Blanco","Negro","Gris")
```

Las tuplas son secuencias de elementos similares a las listas, la diferencia principal es que las tuplas no pueden ser modificadas directamente, es decir, una tupla no dispone de los métodos como append o insert que modifican los elementos de una lista.

Para definir una tupla, los elementos se separan con comas y se encierran entre paréntesis.

```
colores[0]
```

```
'Azul'
```

```
colores[-1]
```

```
'Gris'
```

```
colores[3]
```

```
'Amarillo'
```

Diccionarios(dict)

Los diccionarios son estructuras que contienen una colección de elementos de la forma clave: valor separados por comas y encerrados entre llaves. Las claves deben ser objetos inmutables y los valores pueden ser de cualquier tipo. Necesariamente las claves deben ser únicas en cada diccionario, no así los valores.

Vamos a definir un diccionario llamado edades en el cual cada clave será un nombre y el valor una edad:

```
edades = {"Ana": 25, "David": 18, "Lucas": 35, "Ximena": 30, "Ale": 20}
```

Puede acceder a cada valor de un diccionario mediante su clave, por ejemplo, si quisieramos obtener la edad de la clave `Lucas` se tendría que escribir:

```
edades["Lucas"]
```

```
35
```

```
>>> d = {'uno': 1, 'dos': 2, 'tres': 3}
>>> for e in d:
...     print(e)
...
uno
dos
tres

# Recorrer las claves del diccionario
>>> for k in d.keys():
...     print(k)
...
uno
dos
tres

# Recorrer los valores del diccionario
>>> for v in d.values():
...     print(v)
...
1
2
3

# Recorrer los pares clave valor
>>> for i in d.items():
...     print(i)
...
('uno', 1)
('dos', 2)
('tres', 3)
```

Añadir elementos a un diccionario en Python

Como te decía, la clase dict es mutable, por lo que se pueden añadir, modificar y/o eliminar elementos después de haber creado un objeto de este tipo.

Para añadir un nuevo elemento a un diccionario existente, se usa el operador de asignación =. A la izquierda del operador aparece el objeto diccionario con la nueva clave entre corchetes [] y a la derecha el valor que se asocia a dicha clave.

```
>>> d = {'uno': 1, 'dos': 2}
>>> d
{'uno': 1, 'dos': 2}

# Añade un nuevo elemento al diccionario
>>> d['tres'] = 3
>>> d
{'uno': 1, 'dos': 2, 'tres': 3}
```

Modificar elementos de un diccionario

Para modificar un elemento en un diccionario simplemente escribimos el nombre del diccionario seguido de corchetes, dentro del corchete colocamos la llave que queremos modificar seguido del operador igual y luego el nuevo valor.

```
>>> d = {'uno': 1, 'dos': 2}
>>> d
{'uno': 1, 'dos': 2}
>>> d['uno'] = 1.0
>>> d
{'uno': 1.0, 'dos': 2}
```

Eliminar un elemento de un diccionario en Python

Existen varios métodos para eliminar un elemento de un diccionario en Python por ahora veremos el siguiente:

`del d[clave]:` Elimina el par clave: valor. Si no existe la clave, se lanza la excepción KeyError.

También podemos eliminar todos los datos de un diccionario con el método: `clear()`.

```
# Elimina un elemento con del
>>> del d['tres']
>>> d
{'dos': 2, 'cuatro': 4}

# Trata de eliminar una clave con del que no existe
>>> del d['seis']
Traceback (most recent call last):
  File "<input>", line 1, in <module>
KeyError: 'seis'
```

```
# Borra todos los elementos del diccionario
>>> d.clear()
>>> d
{}
```