# Estructuras de datos y funciones

## Introducción a los diccionarios (Parte I)

{desafío} latam_

***Utilizar estructuras de datos apropiadas para la elaboración de un algoritmo que resuelve un problema acorde al lenguaje Python.***

***Codificar un programa utilizando funciones para la reutilización de código acorde al lenguaje Python.***

- Unidad 1:
  Introducción a Python
- Unidad 2:
  Sentencias condicionales e iterativas
- Unidad 3:
  Estructuras de datos y funciones

***Te encuentras aquí***

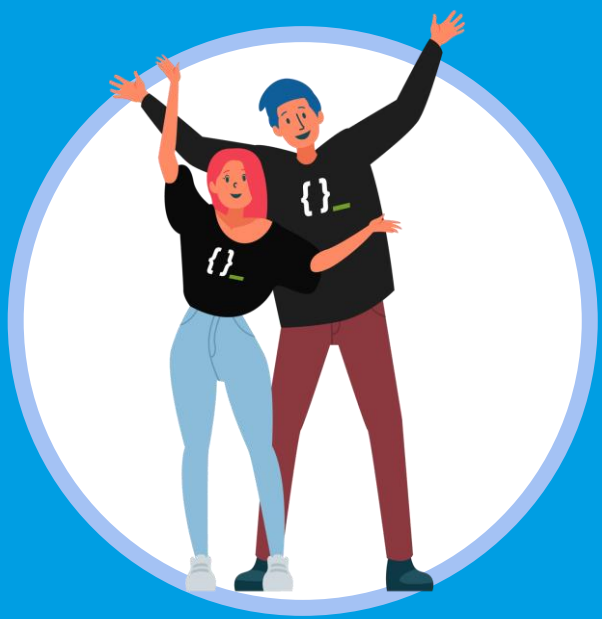
{desafío} latam_

# ¿Qué aprenderás en esta sesión?

- *Utiliza operaciones de creación y acceso a los elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*
- *Utiliza operaciones para la agregación, modificación y eliminación de elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*

{desafío} latam_

# ¿Qué son las listas?



{desafío}
latam_

# /* Diccionarios */

# ¿Qué son?

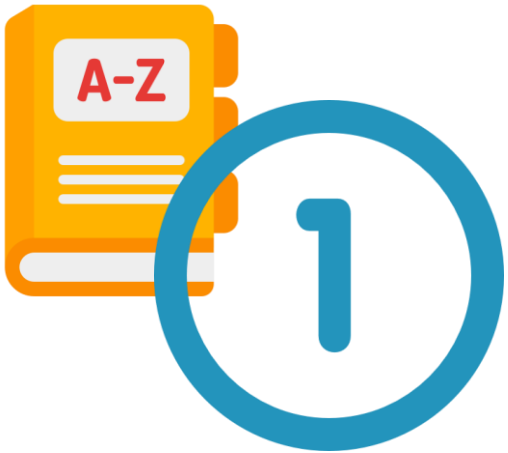
- Estructura de datos compuesta por pares de **`clave:valor`**, donde:
  - Cada clave se asocia con un elemento del diccionario
  - Como toda estructura de datos, permiten almacenar una gran cantidad de datos en una sola variable.

Reciben este nombre porque se leen igual que uno de la vida real, ya que en un diccionario buscamos una palabra y esta nos lleva a la definición:

- La **clave** es equivalente a la **palabra**
- El **valor** es equivalente a su **definición**

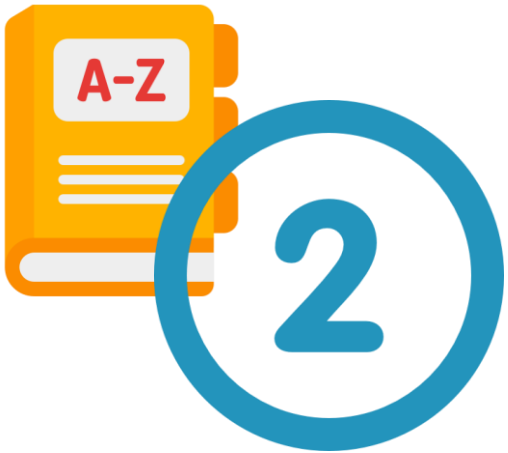
# Listas vs Diccionarios

Para acceder a los elementos de una lista, se hace a través de la posición o el índice asociado al elemento, mientras que en un diccionario, se hace por medio de la clave (o llave).



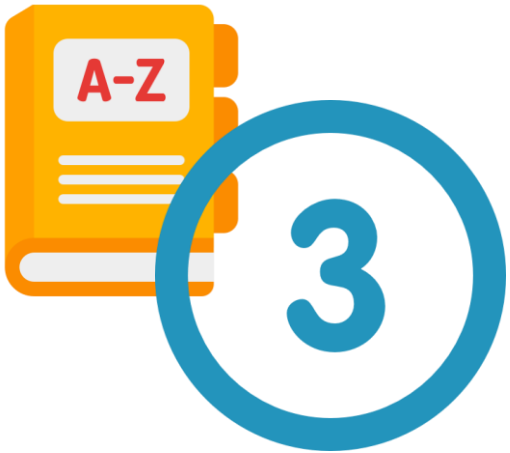
# Listas vs Diccionarios

En una lista, los índices se generan automáticamente para cada elemento, mientras que las claves de un diccionario no se generan automáticamente, sino que son definidas explícitamente al crear el diccionario o al asignar un nuevo elemento a la estructura.



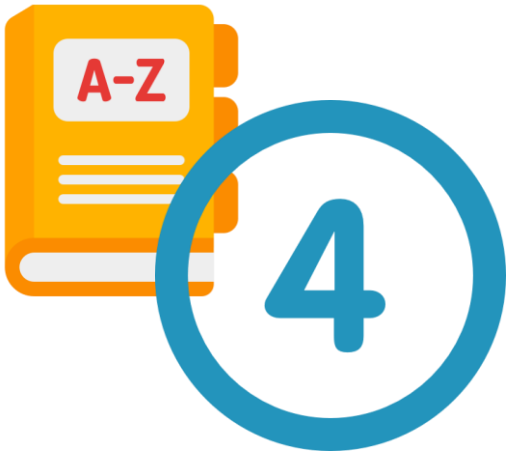
# Listas vs Diccionarios

La lista es un elemento ordenado donde el orden lo entrega el índice, en cambio, en un diccionario es un elemento no ordenado, no hay un elemento que va primero, segundo o último. La única manera de acceder a dicho elemento es mediante su clave.



# Listas vs Diccionarios

Las claves del diccionario pueden ser un string, un número, o incluso un booleano, pero con frecuencia se utilizan los strings.



# Listas vs Diccionarios

```
# En una lista usamos la posición para acceder a un elemento, y los índices se generan en forma implícita
lista = [25, 31, "hola"]
lista[2] # "hola"

# En un diccionario usamos la clave, y se deben definir de forma explícita
diccionario = {"a": 25, "b": 31, "c": "hola"}
diccionario["c"] # "hola"
```

# Crear un diccionario

En Python, un diccionario (vacío) se define con llaves: {}. Cada par de clave y valor se asocia mediante :, en la forma clave: valor.

```
notas = {"Camila": 7, "Antonio": 5, "Felipe": 6, "Antonia": 7}
```

Si al momento de definir un diccionario tenemos muchas llaves, podemos definirlo en múltiples líneas para facilitar la lectura de código.

```
notas = {
    "Camila": 7,
    "Antonio": 5,
    "Felipe": 6,
    "Daniela": 5,
    "Vicente": 7,
}
```

# Acceder a un elemento

## *dentro de un diccionario*

Se accede a un elemento específico utilizando la **clave** de ese **valor**, donde necesariamente se debe utilizar la misma clave en la que está almacenado el elemento, sino podríamos recibir un valor no deseado, o incluso tener un error si la clave no existe.

```
notas["Felipe"] # 6
notas["felipe"] # KeyError: 'felipe'
```

# La clave tiene que ser única

En un diccionario, solo puede haber un valor asociado a una clave.

```
duplicados = {"clave": 1, "clave": 2}
print(duplicados) # {"clave": 2}
```

Al usar dos veces la misma clave, Python se quedará con la última que definimos, ignorando completamente la existencia del primer valor.

# Agregar un elemento
## *a un diccionario*

```
diccionario = {"llave 1": 5}
```

Para agregar un elemento a un diccionario, hace falta especificar una clave nueva, de forma que el nuevo valor ingresado sea identificado con dicha clave.

```
diccionario["llave 2"] = 9
print(diccionario) # {"llave 1": 5, "llave 2": 9}
```

*Presta atención a la sintaxis, donde para definir un diccionario usamos llaves ({}), pero para acceder a sus elementos usamos corchetes [].*

# Cambiar un elemento

## *dentro de un diccionario*

De forma similar a como agregamos un elemento a un diccionario, podemos actualizar el valor de uno. Para esto, simplemente tenemos que redefinir el valor asociado a la clave.

```
diccionario = {"llave 1": 5, "llave 2": 7}
diccionario["llave 2"] = 9
print(diccionario) # {"llave 1": 5, "llave 2": 9}
```

# Eliminar elementos

## *de un diccionario*

Podemos eliminar una llave de un diccionario, junto a su valor, de dos formas: usando el método **pop** del diccionario o utilizando **del**. La principal diferencia entre ambas formas es que al utilizar **pop** obtendremos el valor del elemento eliminado.

```
diccionario = {"celular": 140000, "notebook": 489990, "tablet": 120000, "cargador": 12400}
del diccionario["celular"]
print(diccionario)
```

```
{'notebook': 489990, 'tablet': 120000, 'cargador': 12400}
```

# Eliminar elementos
## *de un diccionario*

Para usar pop se usa la clave, no la posición.

```
eliminado = diccionario.pop("tablet")

print(eliminado)
print(diccionario)
```

```
120000
{'notebook': 489990, 'cargador': 12400}
```

# Unir diccionarios

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta"}

# Union de diccionario_a y diccionario_b
diccionario_a.update(diccionario_b)

# Notar que la unión queda en el primer diccionario
print(diccionario_a)
```

```
{'nombre': 'Alejandra', 'apellido': 'López', 'edad': 33, 'altura': 1.55, 'mascota':
'miti', 'ejercicio': 'bicicleta'}
```

# ¡Cuidado con las colisiones!

***Cuando ambos diccionarios tienen una clave en común, el valor del segundo diccionario sobreescribe al del primero.***

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta", "altura": 155}

# Union de diccionario_a y diccionario_b
diccionario_a.update(diccionario_b)

# Se sobreescribió el valor de altura por el del diccionario_b
print(diccionario_a)
```

```
{'nombre': 'Alejandra', 'apellido': 'López', 'edad': 33, 'altura': 155, 'mascota': 'miti',
'ejercicio': 'bicicleta'}
```

# ¡Cuidado con las colisiones!

***Cuando ambos diccionarios tienen una clave en común, el valor del segundo diccionario sobreescribe al del primero.***

Por lo tanto, no es lo mismo hacer **`diccionario_a.update(diccionario_b)`** que **`diccionario_b.update(diccionario_a)`**

```
diccionario_a = {"nombre": "Alejandra", "apellido": "López", "edad": 33, "altura": 1.55}
diccionario_b = { "mascota":"miti", "ejercicio":"bicicleta", "altura": 155}
diccionario_b.update(diccionario_a)
print(diccionario_b)
```

```
{'mascota': 'miti', 'ejercicio': 'bicicleta', 'altura': 1.55, 'nombre': 'Alejandra',
'apellido': 'López', 'edad': 33}
```

# ¡Practiquemos!

# ¡Practiquemos!

***Realicemos la siguiente actividad en grupos***

1. Crea un diccionario

1. Agrega un elemento

1. Cambia un elemento

1. Elimina un elemento

{desafío} latam_

# ¿Cuál es la diferencia entre listas y diccionarios?

# ¿Para qué sirven los diccionarios?

# Próxima sesión...

- *Utiliza operaciones de creación y acceso a los elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*
- *Utiliza operaciones para la agregación, modificación y eliminación de elementos de una estructura de datos acorde al lenguaje Python para resolver un problema.*

*(continuación)*

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales