Python para Hackers

CloudLamb
E-Learning

CloudLamb E-Learning

Python para Hackers

Contenido del curso

Capítulo 1. Introducción

Capítulo 2. Primeros pasos

Capítulo 3. **Python** *Tema de hoy*

Tipos de datos y variables | Operadores | Cadenas | Condicionales | Bucles | Funciones | Clases y Objetos | Módulos | Archivos | Sockets

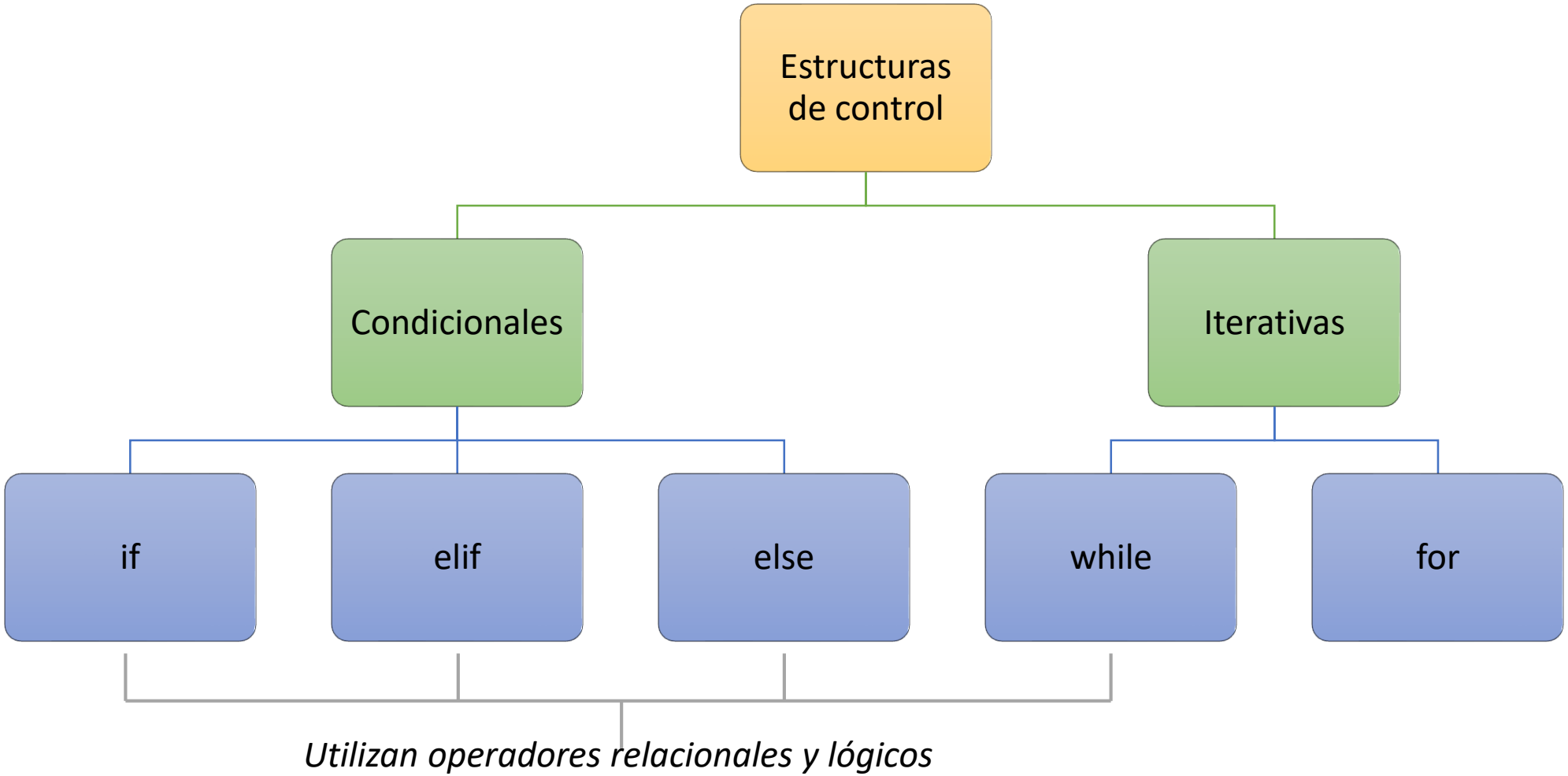
Capítulo 4. Hands-On

CloudLamb E-Learning

Estructuras de control

Estructuras de control

Resumen



Operador relacional
==
!=
>
<
>=
<=

Operador lógico
and
or
xor

Laboratorio

Estructuras de control

FOR *(por cada)*

```
for variable in lista:
    código
```

Código:

```
print("\nFOR")
lista = ["Azul", "Amarillo", "Rojo", "Verde"]
for i in lista:
    print("Color: " + i)
```

Ejecución:

```
FOR
Color: Azul
Color: Amarillo
Color: Rojo
Color: Verde
```

Estructuras de control

Break y continue

En Python, **break** y **continue**, permiten alterar el flujo normal de un ciclo. Se utilizan cuando se requiere romper la iteración actual.

`break`: Rompe el ciclo que lo contiene.
`continue`: Salta el resto del código de la iteración actual e inicia una nueva iteración.

Laboratorio

Estructuras de control

BREAK *(romper)*

Estructura de control:
    break

Código:

```
print("\nBREAK")
for s in "cadena":
    if s == "e":
        break
    print(s)
```

Ejecución:

```
BREAK
c
a
d
```

Laboratorio

Estructuras de control

CONTINUE *(continuar)*

Estructura de control:
    continue

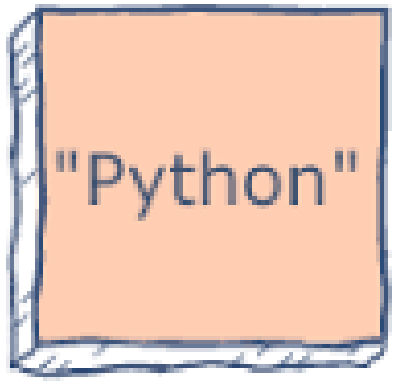
Código:

```
print("\nCONTINUE")
for c in "abcdefjhi":
    if c == "d":
        continue
    print(c)
```

Ejecución:

```
CONTINUE
a
b
c
e
f
j
h
i
```

Strings



Es un tipo de dato que contiene **secuencias de caracteres**.

Las cadenas en Python tienen que ir contenidas entre comillas dobles o sencillas:

```
“Hola mundo”
‘Hola mundo’
```

CloudLamb E-Learning

Strings

Escape

En Python cuando se utilizan *strings* es necesario utilizar un **escape (\)** para representar ciertos caracteres dentro de la cadena, como: “, ‘ ó \.

```
>>> print("Ella dijo: "Vamos a comer"")
SyntaxError: invalid syntax
```

```
>>> print("Ella dijo: \"Vamos a comer\"")
Ella dijo: "Vamos a comer"
```

```
>>> print('Ella dijo: "Vamos a comer"')
Ella dijo: "Vamos a comer"
```

```
>>> print("C:\\Users\Juan")
C:\Users\Juan
```

```
>>> print("C:\\\\Users\\Juan")
C:\\Users\Juan
```

Strings

Concatenar

Se pueden **concatenar dos o más cadenas** utilizando el operador “+”.

```
>>> a = "Hola"
>>> b = " mundo"
>>> print(a + b)
Hola mundo
>>>
```

```
>>> a = "La ruta de instalación es: "
>>> b = "c:\\\\Windows\\Program\\"
>>> print(a + b)
La ruta de instalación es: c:\\Windows\Program\
```

Strings
Métodos

El tipo de dato “string” forma parte de la librería estándar de Python que viene ya integrado en el interprete y cuenta con **métodos** que ayudan a manipular una cadena.

```
01. s = "Hola Mundo!"
02.
03. print(s.lower()) # Convierte todas las letras en minusculas
04. print(s.upper()) # Convierte todas las letras en mayusculas
05. print(s.capitalize()) # Capitaliza la primera letra de la cadena
06. print(s.count("a")) # Cuenta todas las letras 'a'
07. print(s.find("Mundo")) # Busca en la cadena un valor y devuelve su posicion
08.
09. # Separa la cadena en el separador indicado y devuelve una lista
10. print(s.split(" "))
11.
12. # Reemplaza texto en la cadena
13. print(s.replace("Mundo", "clase"))
14.
15. # Quita los espacios al inicio y final de la cadena
16. s = " Ejemplo    "
17. print("Esto es un " + s + " con cadenas")
18. print("Esto es un " + s.strip() + " con cadenas")
```

Strings
Métodos

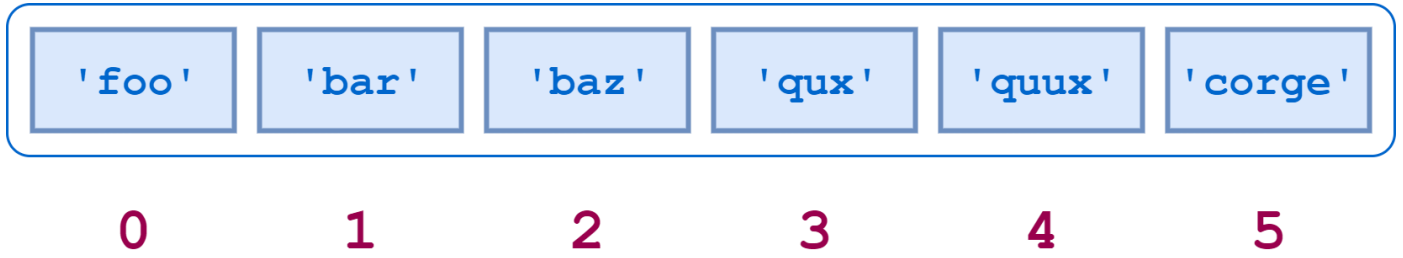
Ejecución del código

```
hola mundo!
HOLA MUNDO!
Hola mundo!
1
5
['Hola', 'Mundo!']
Hola clase!
Esto es un  Ejemplo     con cadenas
Esto es un Ejemplo con cadenas
```

CloudLamb
E-Learning

Listas

Es un tipo de dato que **contiene diferentes tipos de datos** ordenados y que pueden ser modificados.



Listas

Obtención de elementos

Los elementos de una lista se pueden **obtener** utilizando su posición numérica.

```
01. # Listas
02. print("\nLISTAS\n")
03. abc = ["a","b","c","d","e","f"]
04.
05. print(abc[0]) # Primer elemento de la lista
06. print(abc[0:5]) # Primeros 5 elementos de la lista
07. print(abc[:5]) # Otra version del comando anterior
08. print(abc[-3:]) # Ultimos 3 elementos de la lista
09. print(abc[:]) # Todos los elementos de la lista
```

```
a
['a', 'b', 'c', 'd', 'e']
['a', 'b', 'c', 'd', 'e']
['d', 'e', 'f']
['a', 'b', 'c', 'd', 'e', 'f']
```

Ejecución del código

Listas

Obtención de elementos

También se pueden utilizar la estructura de control iterativa **for** para obtener los elementos de una lista.

```
01. # Obtencion de elementos de una lista con estructuras de control
02. for elemento in abc:
03.     print(elemento)
```

```
a
b
c
d
e
f
```

Ejecución del código

CloudLamb E-Learning

Listas

Métodos

El tipo de dato lista cuenta con una serie de **métodos para manipular los elementos de una lista**.

```
01. # Metodos de las listas
02. abc.append("b") # Agrega un elemento al final de la lista
03. print(abc)
04.
05. abc.remove("a") # Elimina el elemento especificado de la lista
06. print(abc)
07.
08. print(abc.count("b")) # Cuenta los elementos que tienen el valor especificado
09.
10. abc.sort() # Ordena los elementos de la lista
11. print(abc)
12.
13. abc.insert(2,"s") # Agrega un nuevo elemento en la posicion indicada
14. print(abc)
```

Listas

Métodos

Ejecución del código

```
['a', 'b', 'c', 'd', 'e', 'f', 'b']
['b', 'c', 'd', 'e', 'f', 'b']
2
['b', 'b', 'c', 'd', 'e', 'f']
['b', 'b', 's', 'c', 'd', 'e', 'f']
```

CloudLamb
E-Learning

Funciones

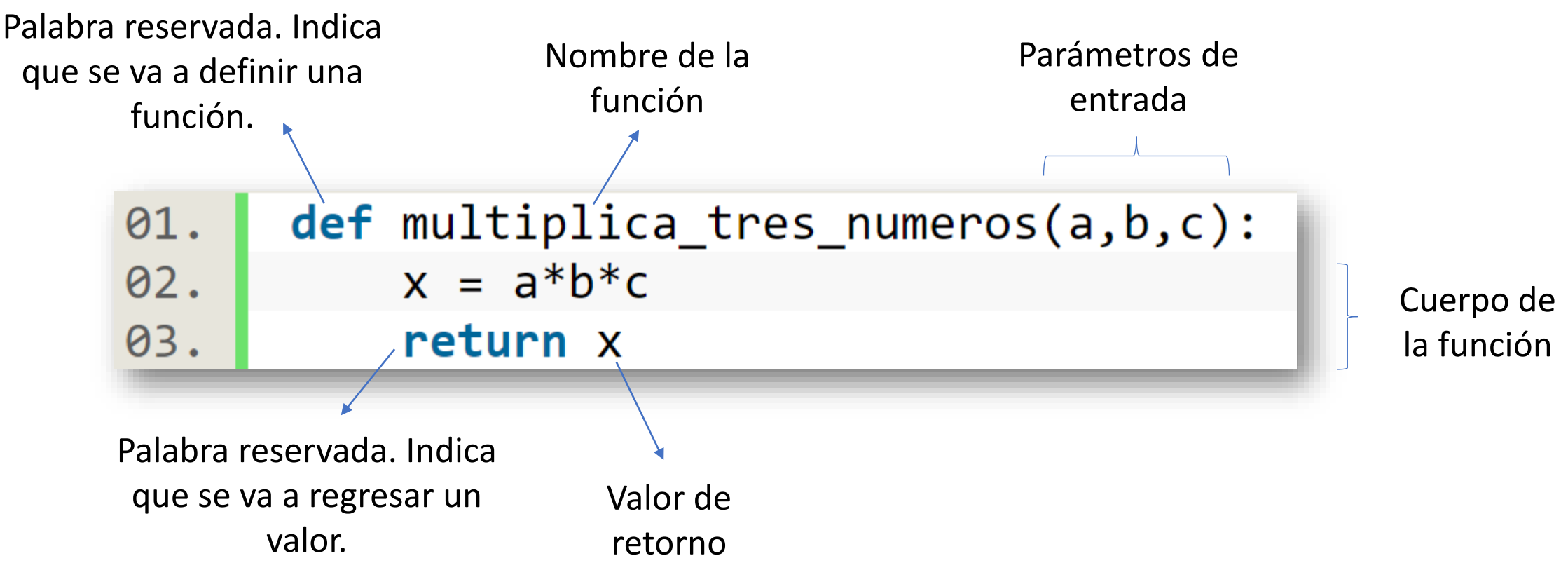
Es un **bloque de código** que sólo se ejecuta cuando es solicitado. Se le puede pasar datos de entrada (**parámetros**) y puede regresar algún tipo de dato (**retorno**).

```
def celsius_to_fahr(temp):
    return 9/5 * temp + 32
```

CloudLamb E-Learning

Funciones

Anatomía de una función



CloudLamb E-Learning

Funciones

Laboratorio

Definición de funciones

```
01. # Funciones
02. print("\nFUNCIONES\n")
03. # Funcion
04. def imprime_lista(lista):
05.     print("\nElementos de la lista:")
06.     for i in lista:
07.         print(i)
08.
09. def buscar_elemento(lista,elemento):
10.     print("\nBuscando el elemento: " + elemento)
11.     e = ""
12.     for i in lista:
13.         if i == elemento:
14.             return "El elemento <" + elemento + "> EXISTE en la lista"
15.     return "El elemento <" + elemento + "> NO existe en la lista"
16.
```

Funciones

Laboratorio

Llamada de funciones

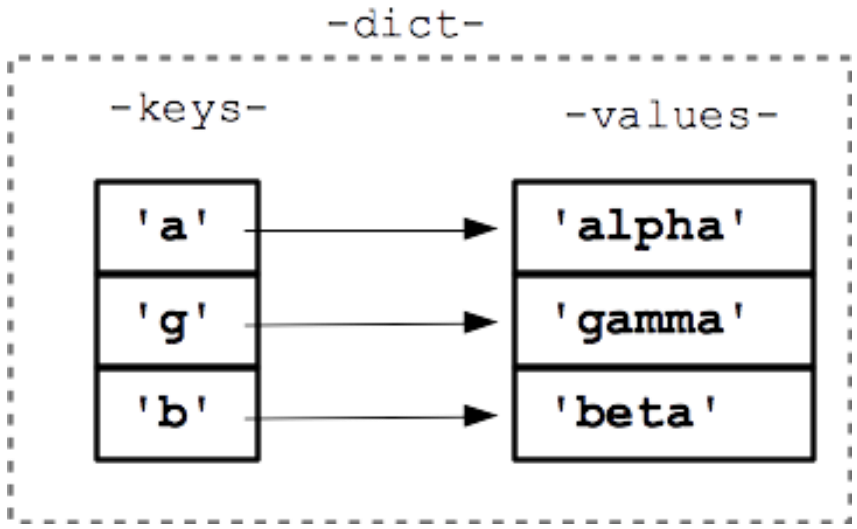
```
18.  a = ["Pedro", "Antonio", "Alma", "Perla"]
19.  imprime_lista(a)
20.  print(buscar_elemento(a,"Alma"))
```

Ejecución del código:

```
Elementos de la lista:
Pedro
Antonio
Alma
Perla

Buscando el elemento: Alma
El elemento <Alma> EXISTE en la lista
```

Diccionarios

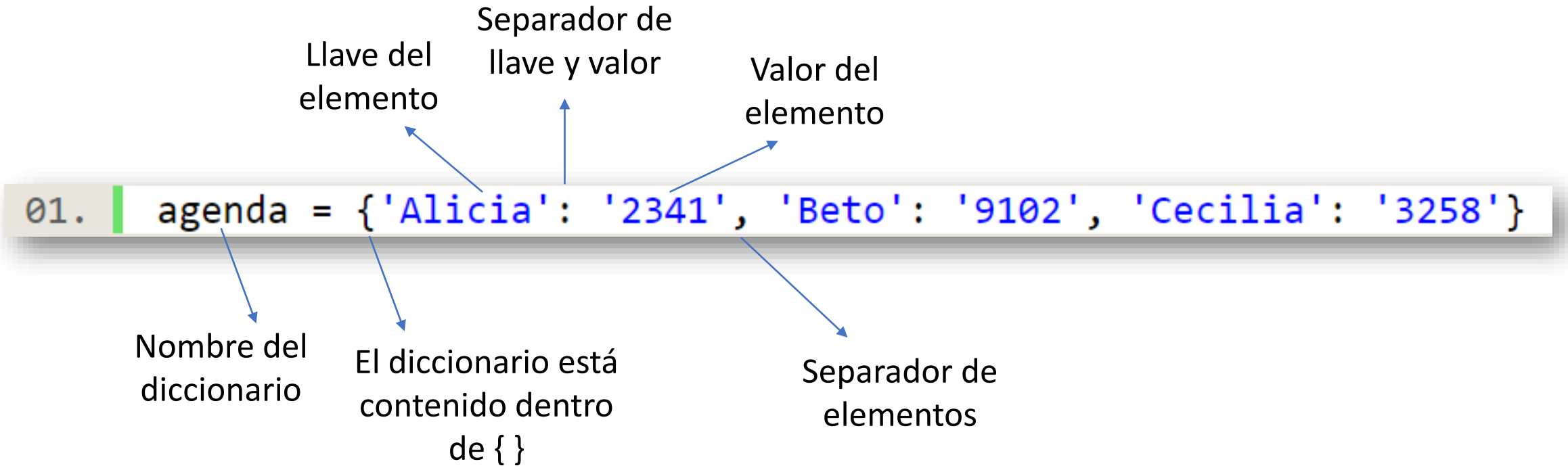


Es un tipo de dato en Python donde sus **elementos no están ordenados** pero se almacenan bajo una **llave** (número, cadena o una tupla)

Diccionarios

Anatomía de un diccionario

Los elementos de un diccionario consisten pares de **llaves** y su **valor** correspondiente.



CloudLamb E-Learning

Diccionarios

Operaciones básicas

```
01. #Diccionarios
02. print("\nDICCIONARIOS\n")
03. agenda = {'Alicia': '2341', 'Beto': '9102', 'Cecilia': '3258'}
04. 
05. print(agenda)
06. print(len(agenda)) # Determina la longitud del diccionario
07. print(agenda["Alicia"]) # Devuelve el valor de la llave indicada
08. agenda["Beto"] = '9120' # Modifica el valor de la llave indicada
09. print(agenda)
10. 
11. #Imprime las llaves del diccionario
12. for x in agenda:
13.   print(x)
14. 
15. # Imprime todos los valores del diccionario
16. for x in agenda:
17.   print(agenda[x])
18. 
19. # Agrega un nuevo elemento al diccionario
20. agenda["Pedro"] = '3456'
21. print(agenda)
22. 
23. # Elimina el elemento con la llave indicada
24. agenda.pop("Cecilia")
25. print(agenda)
```

Diccionarios

Operaciones básicas

Ejecución del código:

```
DICCIONARIOS

{'Alicia': '2341', 'Beto': '9102', 'Cecilia': '3258'}
3
2341
{'Alicia': '2341', 'Beto': '9120', 'Cecilia': '3258'}
Alicia
Beto
Cecilia
2341
9120
3258
{'Alicia': '2341', 'Beto': '9120', 'Cecilia': '3258', 'Pedro': '3456'}
{'Alicia': '2341', 'Beto': '9120', 'Pedro': '3456'}
```

Próxima clase...

- **Primera Evaluación**
- **Capítulo 4:** Hands-On

CloudLamb
E-Learning

CloudLamb E-Learning

CloudLamb

¡Muchas gracias por su atención!