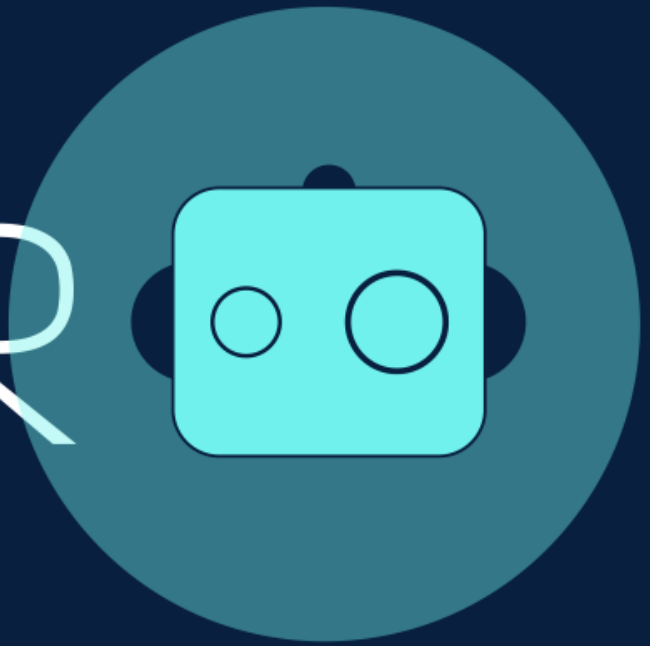
UMAKER



CLASE 2

Estructuras de control

¿Qué aprenderemos hoy?

- Manejo de las estructuras de Control
- Acerca de las estructuras de datos en Pyhton
- Estructuras de datos : Listas & Tuplas
- Metodos de las variables : Listas &Tuplas
- Slicing en Python
- Swaping en Python
- Cpython vs PyPy

Python If ... Else

Condiciones en Python y declaraciones If

- Python admite las condiciones lógicas habituales de las matemáticas:

igual	a == b
No es igual	a! = b
Menor que	a < b
Menos o igual que	a <= b
Mayor que	a > b
Mayor o igual que	a >= b

- Estas condiciones se pueden usar de varias maneras, más comúnmente en "declaraciones if" y bucles.

Se escribe una "declaración if" utilizando la palabra clave if .

Ejemplo

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

En este ejemplo, usamos dos variables, *a* y *b* , que se usan como parte de la instrucción if para probar si *b* es mayor que *a* . Como *a* es 33 y *b* es 200 , sabemos que 200 es mayor que 33, y así es la impresión de pantalla que a *"b is greater than a".*

Identación

Python se basa en la identación (espacio en blanco al comienzo de una línea) para definir el alcance en el código. Otros lenguajes de programación a menudo usan llaves para este propósito.

Ejemplo
Si la declaración, sin sangría (generará un error):

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

Elif

La palabra clave *elif* es la forma que tiene Python de decir "si las condiciones anteriores no eran ciertas, entonces intente esta condición".

Ejemplo

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

En este ejemplo, *a* es igual a *b* , por lo que la primera condición no es verdadera, pero la condición *elif* es verdadera, por lo que imprimimos en la pantalla que *"a and b are equal".*

Else

La palabra clave *else* captura cualquier cosa que no esté atrapada por las condiciones anteriores.

Ejemplo

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

En este ejemplo, *a* es mayor que *b* , por lo que la primera condición no es verdadera, tampoco la condición *elif* no es verdadera, así que pasamos a la condición *else* e imprimimos en la pantalla que *"a es mayor que b".*

También puede tener un *else* sin el *elif*:

Ejemplo

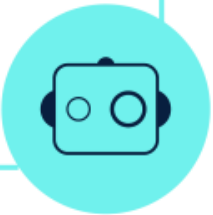
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Algo corto con *if*

Si solo tiene que ejecutar una instrucción, puede ponerla en la misma línea que la instrucción if.

Ejemplo

```
if a > b: print("a is greater than b")
```



Algo corto con if ... else

Si solo tiene que ejecutar una instrucción, una para if y otra para más, puede poner todo en la misma línea:

Ejemplo

```
a = 2
b = 330
print("A") if a > b else print("B")
```

También puede tener varias instrucciones *else* en la misma línea:

Ejemplo
Una línea si otra declaración, con 3 condiciones:

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

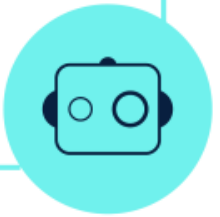
And

La palabra clave *and* es un operador lógico, y se usa para combinar declaraciones condicionales:

Ejemplo

Pruebe si *a* es mayor que *b*, Y si *c* es mayor que *a*:


```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```



Or

La palabra clave or es un operador lógico y se usa para combinar declaraciones condicionales:

Ejemplo
Pruebe si *a* es mayor que *b*, O si *a* es mayor que *c*:

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

If anidados

Puede tener declaraciones *if* dentro de las declaraciones *if*, esto se llama declaraciones anidadas *if*

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

La declaración pass

Las declaraciones *if* no pueden estar vacías, pero si por alguna razón tiene una declaración if sin contenido, escriba *pass* para evitar un error.

Ejemplo

```
a = 33
b = 200

if b > a:
  pass
```

Bucles en Python

Python tiene dos comandos de bucle primitivos:

- bucle `while`
- bucle `for`

El bucle while

Con el bucle `while` podemos ejecutar un conjunto de instrucciones, siempre y cuando se cumpla una condición.

Ejemplo

Imprima i siempre que sea menor que 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Nota: recuerde incrementar i, o el bucle continuará para siempre.

El bucle *while* requiere variables relevantes para estar listo, en este ejemplo, tenemos que definir una variable de indexación, `i` , lo que nos pone a 1.

La declaración *break*

Con la instrucción *break* podemos detener el ciclo incluso si la condición while es verdadera:

Ejemplo
Salga del bucle cuando sea 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

La declaración *continue*

Con la instrucción continue podemos detener la iteración actual y continuar con la siguiente:

Ejemplo
Continúe con la siguiente iteración si soy 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

La declaración *else*

Con la instrucción *else* podemos ejecutar un bloque de código una vez cuando la condición ya no es verdadera:

Ejemplo

Imprima un mensaje una vez que la condición sea falsa:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Listas de Python

Colecciones de Python (matrices)

Hay cuatro tipos de datos de recopilación en el lenguaje de programación Python:

- La lista es una colección ordenada y modificable. Permite miembros duplicados.
- Tuple es una colección ordenada e inmutable. Permite miembros duplicados.
- Set es una colección que no está ordenada ni indexada. No hay miembros duplicados.
- Dictionary es una colección desordenada, modificable e indexada. No hay miembros duplicados.

Al elegir un tipo de colección, es útil comprender las propiedades de ese tipo. Elegir el tipo correcto para un conjunto de datos en particular podría significar la retención de significado y podría significar un aumento en la eficiencia o la seguridad.

Lista

Una lista es una colección ordenada y modificable. En Python las listas se escriben entre corchetes.

Ejemplo
Crear una lista:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Índice

Accede a los elementos de la lista haciendo referencia al número de índice:

Ejemplo
Imprima el segundo elemento de la lista:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Indexación Negativa

La indexación negativa significa comenzar desde el final, se -1refiere al último elemento, se -2refiere al segundo último elemento, etc.

Ejemplo

Imprima el último elemento de la lista:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

Rango de índices

Puede especificar un rango de índices especificando dónde comenzar y dónde terminar el rango.
Al especificar un rango, el valor de retorno será una nueva lista con los elementos especificados.

Ejemplo

Devuelve el tercer, cuarto y quinto elemento:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

Nota: La búsqueda comenzará en el índice 2 (incluido) y finalizará en el índice 5 (no incluido).

Recuerde que el primer elemento tiene el índice 0.

Al omitir el valor inicial, el rango comenzará en el primer elemento:

Ejemplo
Este ejemplo devuelve los elementos desde el principio a "naranja":

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

Al omitir el valor final, el rango continuará hasta el final de la lista:

Ejemplo
Este ejemplo devuelve los elementos de "cereza" y hasta el final:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

Rango de índices negativos
Especifique índices negativos si desea comenzar la búsqueda desde el final de la lista:

Ejemplo
Este ejemplo devuelve los elementos del índice -4 (incluido) al índice -1 (excluido)

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

Cambiar valor del artículo

Para cambiar el valor de un elemento específico, consulte el número de índice:

Ejemplo

Cambiar el segundo elemento:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

Recorrer una lista

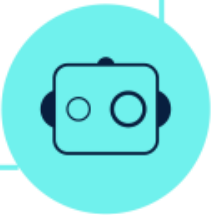
Puede recorrer los elementos de la lista utilizando un for ciclo:

Ejemplo

Imprima todos los elementos de la lista, uno por uno:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

El bucle *for* lo veremos mas adelante



Comprobar si el artículo existe

Para determinar si un elemento específico está presente en una lista, use la inpalabra clave:

Ejemplo
Compruebe si "apple" está presente en la lista:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

Longitud de la lista
Para determinar cuántos elementos tiene una lista, use la función *len()* :

Ejemplo
Imprima el número de elementos en la lista:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Agregar artículos

Para agregar un elemento al final de la lista, use el método append () :

Ejemplo

Usando el método *append()* para agregar un elemento:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Para agregar un elemento en el índice especificado, use el método *insert ()* :

Ejemplo

Insertar un elemento como segunda posición:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```


Remover el artículo

Existen varios métodos para eliminar elementos de una lista:

Ejemplo
El método remove() elimina el elemento especificado:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Ejemplo
El pop()método elimina el índice especificado (o el último elemento si no se especifica el índice):

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

Ejemplo
La delpalabra clave elimina el índice especificado:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```


Ejemplo
El clear()método vacía la lista:

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Copiar una lista

No se puede copiar una lista simplemente escribiendo list2 = list1, ya que: list2sólo será una referencia a list1, y los cambios realizados en list1Automáticamente, también sean por list2.

Hay formas de hacer una copia, una es usar el método de Lista incorporado copy().

Ejemplo
Haga una copia de una lista con el copy()método:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

Otra forma de hacer una copia es usar el método incorporado list().

Ejemplo
Haga una copia de una lista con el list()método:

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

Unir dos listas
Hay varias formas de unir o concatenar dos o más listas en Python.

Una de las formas más fáciles es mediante el uso del + operador.

Ejemplo
Unir dos listas:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```


Otra forma de unir dos listas es agregando todos los elementos de list2 a list1, uno por uno:

Ejemplo
Anexar list2 a list1:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)
```

O puede usar el extend() método, cuyo propósito es agregar elementos de una lista a otra lista:

Ejemplo
Use el extend()método para agregar list2 al final de list1:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

El constructor list ()

También es posible usar el constructor list () para hacer una nueva lista.

Ejemplo

Usando el list()constructor para hacer una Lista:

```
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```


Métodos de lista

Python tiene un conjunto de métodos integrados que puede usar en las listas.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Tuplas en Python

Tupla
Una tupla es una colección ordenada e inmutable . En Python, las tuplas se escriben entre corchetes.

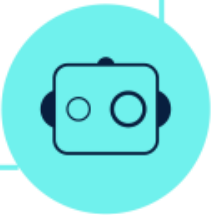
Ejemplo
Crea una tupla:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Acceso a elementos de tupla
Puede acceder a los elementos de tupla haciendo referencia al número de índice, entre corchetes:

Ejemplo
Imprima el segundo elemento en la tupla:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```



Indexación Negativa

La indexación negativa significa comenzar desde el final, se -1refiere al último elemento, se -2refiere al segundo último elemento, etc.

Ejemplo
Imprima el último elemento de la tupla:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

Rango de índices
Puede especificar un rango de índices especificando dónde comenzar y dónde terminar el rango.
Al especificar un rango, el valor de retorno será una nueva tupla con los elementos especificados.

Ejemplo
Devuelve el tercer, cuarto y quinto elemento:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

Nota: La búsqueda comenzará en el índice 2 (incluido) y finalizará en el índice 5 (no incluido).

Recuerde que el primer elemento tiene el índice 0.

Rango de índices negativos

Especifique índices negativos si desea comenzar la búsqueda desde el final de la tupla:

Ejemplo

Este ejemplo devuelve los elementos del índice -4 (incluido) al índice -1 (excluido)

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

Cambiar valores de tupla

Una vez que se crea una tupla, no puede cambiar sus valores. Las tuplas son inmutables o inmutables como también se le llama.

Pero hay una solución alternativa. Puede convertir la tupla en una lista, cambiar la lista y volver a convertir la lista en una tupla.

Ejemplo

Convierta la tupla en una lista para poder cambiarla:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Bucle a través de una tupla

Puede recorrer los elementos de la tupla utilizando un forbucle.

Ejemplo

Iterar a través de los elementos e imprimir los valores:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

El tema del bucle *for* lo veremos mas adelante

Comprobar si el artículo existe

Para determinar si un elemento específico está presente en una tupla, use la inpalabra clave:

Ejemplo

Compruebe si "manzana" está presente en la tupla:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

Longitud de la tupla

Para determinar cuántos elementos tiene una tupla, use el len()método:

Ejemplo

Imprima el número de elementos en la tupla:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Agregar artículos

Una vez que se crea una tupla, no puede agregarle elementos. Las tuplas son inmutables .

Ejemplo

No puede agregar elementos a una tupla:

```
thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange" # This will raise an error
print(thistuple)
```

Crear tupla con un artículo

Para crear una tupla con un solo elemento, debe agregar una coma después del elemento, a menos que Python no reconozca la variable como una tupla.

Ejemplo

Una tupla de artículo, recuerda la coma:

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Eliminar elementos

Nota: No puede eliminar elementos en una tupla.

Las tuplas no se pueden cambiar , por lo que no puede eliminar elementos de ellas, pero puede eliminarlas por completo:

Ejemplo

La palabra clave del puede eliminar la tupla por completo:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

Une dos tuplas

Para unir dos o más tuplas puede usar el + operador:

Ejemplo
Une dos tuplas:

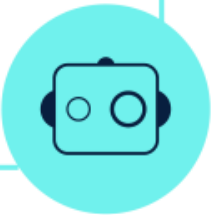
```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

El constructor tuple ()
También es posible usar el constructor tuple () para hacer una tupla.

Ejemplo
Usando el método tuple () para hacer una tupla:

```
thistuple = tuple(("apple", "banana", "cherry")) # note the double round-brackets
print(thistuple)
```



Métodos de tupla

Python tiene dos métodos integrados que puede usar en tuplas.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python para bucles

Un bucle for se usa para iterar sobre una secuencia (que es una lista, una tupla, un diccionario, un conjunto o una cadena).

Esto es menos como la palabra clave for en otros lenguajes de programación, y funciona más como un método iterador como se encuentra en otros lenguajes de programación orientados a objetos.

Con el bucle for podemos ejecutar un conjunto de declaraciones, una vez para cada elemento de una lista, tupla, conjunto, etc.

Ejemplo

Imprima cada fruta en una lista de frutas:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

El bucle for no requiere una variable de indexación para establecer de antemano.

Bucle a través de una cadena
Incluso las cadenas son objetos iterables, contienen una secuencia de caracteres:

Ejemplo
Recorre las letras de la palabra "banana":

```
for x in "banana":
  print(x)
```

La declaración *break*

Con la instrucción break podemos detener el ciclo antes de que haya pasado por todos los elementos:

Ejemplo
Salga del bucle cuando xsea "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Ejemplo

Salga del bucle cuando x sea "banana", pero esta vez el descanso se produce antes de la impresión:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

La declaración continue

Con la instrucción continue podemos detener la iteración actual del ciclo y continuar con la siguiente:

Ejemplo

No imprimir plátano:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

La función range ()

Para recorrer un conjunto de código un número específico de veces, podemos usar la función range () ,
La función range () devuelve una secuencia de números, comenzando desde 0 por defecto, y se incrementa en 1 (por defecto), y termina en un número específico.

Ejemplo
Usando la función range ():

```
for x in range(6):
  print(x)
```

Tenga en cuenta que el rango (6) no son los valores de 0 a 6, sino los valores de 0 a 5.

La función range () se establece por defecto en 0 como valor inicial, sin embargo, es posible especificar el valor inicial agregando un parámetro: range (2, 6) , que significa valores de 2 a 6 (pero sin incluir 6):

Ejemplo
Usando el parámetro de inicio:

```
for x in range(2, 6):
  print(x)
```

La función range () por defecto incrementa la secuencia en 1, sin embargo, es posible especificar el valor de incremento agregando un tercer parámetro: range (2, 30, 3) :

Ejemplo
Incremente la secuencia con 3 (el valor predeterminado es 1):

```
for x in range(2, 30, 3):
  print(x)
```

Lo demás en For Loop

La elsepalabra clave en un forbucle especifica un bloque de código que se ejecutará cuando finalice el bucle:

Ejemplo
Imprima todos los números del 0 al 5 e imprima un mensaje cuando finalice el ciclo:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Bucles anidados

Un bucle anidado es un bucle dentro de un bucle.

El "bucle interno" se ejecutará una vez por cada iteración del "bucle externo":

Ejemplo
Imprime cada adjetivo para cada fruta:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

La declaración de *pass*
forlos bucles no pueden estar vacíos, pero si por alguna razón tiene un forbucle sin contenido, passingrese la declaración para evitar un error.

Ejemplo

```
for x in [0, 1, 2]:
  pass
```