Herramientas Básicas de Visualización de Datos

Listas, loops y diccionarios

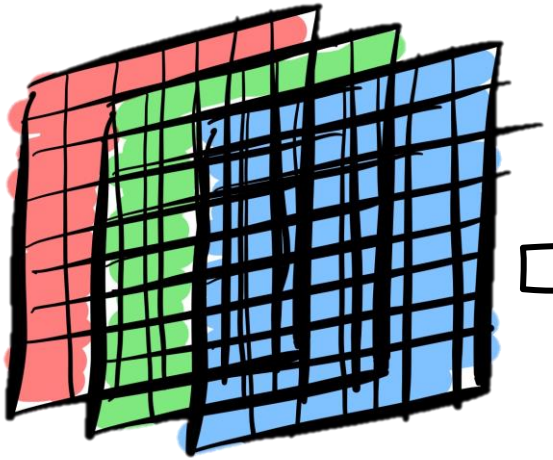
Universidad Privada Boliviana

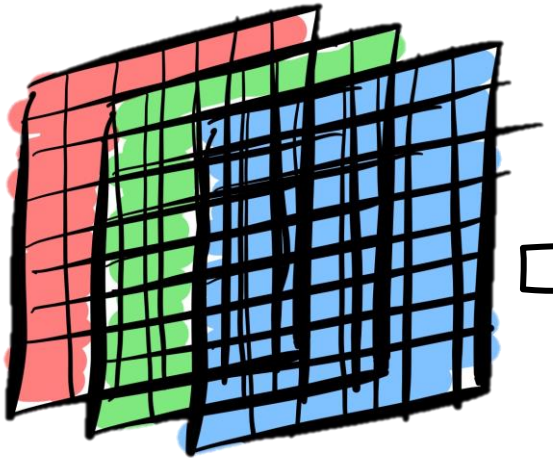
Hugo Condori Quispe, Ph.D.

hugo.condori@fulbrightmail.org

Listas

Imágenes representadas por pixeles





```
array([[[ 63,  95, 208, 231, 190, 186, 196, 208,  91, 110,  31,  66,
         163,  75,  96,  84,  86, 106,   0,  57,  90, 147,  31, 148,
          63,  10,  61, 231,  80, 127, 174, 240,  45,  56, 153, 193,
         238,  85,  38, 118,  94, 145, 144, 192,  82,  48, 145,  17,
         176, 150,  43, 111,  55, 104, 249,  42,  25, 145,  88, 102,
         115,  12,  91, 238, 235, 198, 141,  96, 118, 167, 222, 123,
          43, 235,  10, 205,  68, 249, 208,  71, 112,  94, 146,   1,
          16, 134,   7,  94,  28, 148,  77,  58, 122, 232,  59, 194,
          16,   7, 204, 120, 174, 192,  72, 197,  95,  18, 188, 243,
          83, 201, 137, 250, 237, 149, 222, 150,  96,   2, 191,  15,
         164, 167, 147,  70, 193,  30,  33, 108,  65,  77, 192, 237,
         189,  47, 166, 116, 141, 107,  39, 158, 241,  33, 170,  37,
         102, 173,  82, 181, 229, 189, 161,  18, 251, 187,  95,  28,
          45, 132,  66, 201, 166, 243,  18, 135,   9, 171, 110,  74,
          99, 207, 185, 228, 243,  17, 236,  60,  61, 142,  12,  54,
         203,  33,  30,  97, 202, 231,  34, 137, 162, 172, 100, 233,
         116,  58, 172,  35,  91, 100,   2,   6, 221,  86, 176,  15,
          15, 117,  57,  54, 108, 158, 145, 155,  89, 116,  73,  28,
         169,  33, 175, 120, 129, 187,  18,  82],
        [ 36,  94,  73, 107, 127, 205, 232, 139, 189, 180,  94, 192,
          64, 218, 138,  38, 249, 243,  36,  31,  41, 116,  81, 208,
         106, 176, 108,   2,  38,  86, 152, 185,  43, 145, 217, 109,
          93, 144, 250,  56,  83,  52, 248, 115, 129, 236, 102, 117,
         211, 200,   9,  11,  49, 120, 251, 101, 177,  39,  43,  25,
         217, 108, 121, 217, 225, 201, 124,  10,  80, 102, 161, 100,
```

```
array([[[154, 187,  54,  33,  40,  57,  67, 196,  48,  77, 223,  36,
          70,  36, 173, 229, 131,  53,  15,  68,  39, 241, 210,   3,
           7, 135,  11,  35,  63,  19, 183,  65, 214, 194, 156, 177,
           5, 238, 241, 112, 183,  30, 234,  50, 201,  67,   1, 174,
           2, 190, 161, 210,  90,  44,  66,   1, 145, 176,  22, 232,
          57, 133, 234,  52, 195, 104,  18, 180,  65,  67, 244,  72,
         188, 104, 206, 186, 224,  95, 247, 145, 212, 126,  40,  54,
         170, 251, 131, 157,   9, 132, 221, 194, 205,  52,  97, 106,
         145,  64,  12, 244,  57, 110,   5, 126, 106, 121, 145,  31,
         185,  10,   5,  25,   8,  99,  17,  24, 228, 169, 112, 154,
          50, 207, 215, 211,  92,  25, 230,  40,  70, 213, 226, 218,
          72, 161,  32,  90,   7, 250,  79, 221, 127, 187, 175, 228,
           4,  56, 122, 206, 130, 104, 231, 155,  60, 144, 106, 242,
          21, 237, 217, 120, 137, 191,  87, 218, 207,  22,  11,   0,
          38,  28,  19,  21, 253, 205,  28, 181,  40, 236, 204, 233,
         244, 167,  10, 191,  28,   3,  53, 102, 157, 148, 143, 105,
         210, 195,  83, 101, 235, 204, 166,  58, 241, 170,  26, 162,
         110, 162,  43, 200,   5, 213,  80, 130, 237, 193, 241, 103,
          68,  17, 115,  15,  81, 171,  34,  33],
        [172, 181,  55, 176,  99, 100,   8, 105, 252,  17, 212, 233,
         214, 174,  91, 212,   5,  13,  18,  21, 183,  99,  78,  44,
         100, 175, 241, 135,  37, 135, 191, 215, 129, 138, 221,  25,
          90, 184,  69,  95,  58, 133, 217, 188,  65, 160, 180, 198,
          70,   1, 104, 136,  39, 197,  68, 220, 239,  26, 104,  96,
          79,  44,  97, 131, 214,  19, 240, 247,  36,  23, 205,  85,
```

method - Sum()

valor opcional que se agregará a la suma de la secuencia; cero si no se especifica

```
sum(listlike,[,start])
```

List methods

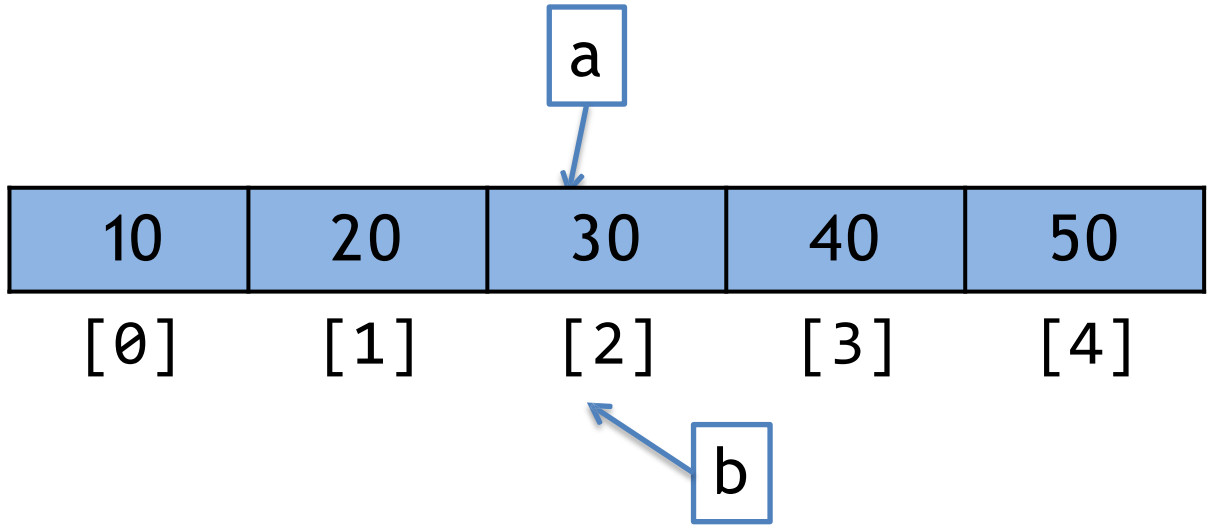
```
>>> fruits = ['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
1
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4) # Find next banana starting a position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'tangerine']
>>> fruits.append('grape')
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'tangerine', 'grape']
>>> fruits.sort()
>>> fruits
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'pear', 'tangerine']
>>> fruits.pop()
'tangerine'
```

List Aliasing

'*Aliasing*' significa dar otro nombre al objeto existente. No significa copiar.

```
a = [10, 20, 30, 40, 50]
b = a
```

la modificación en 'a' afectará a 'b' y viceversa

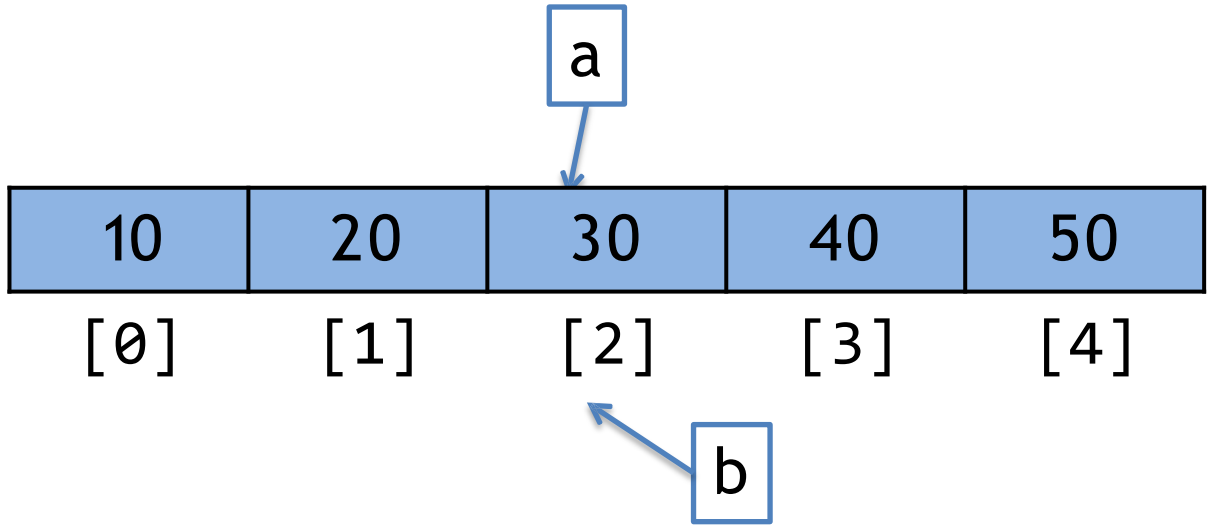


Copia de una lista

Cuando copiamos una lista, se almacena una copia separada de todos los elementos en otra lista. Ambas listas son independientes.

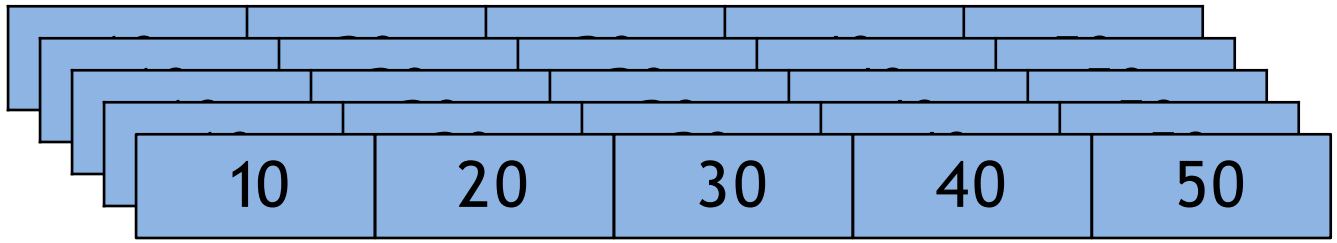
```
a = [10, 20, 30, 40, 50]
b = a.copy()
```

la modificación en a no afectará a b y viceversa

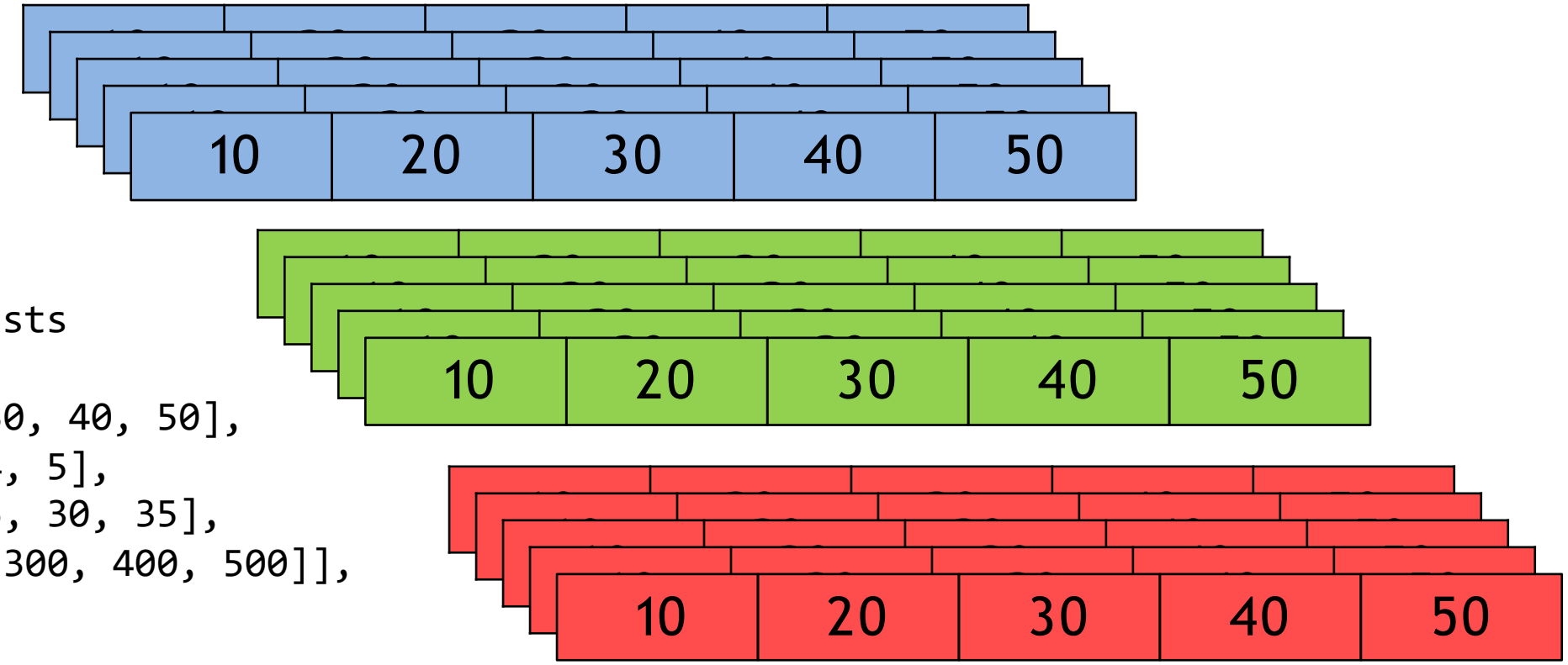


Listas anidadas

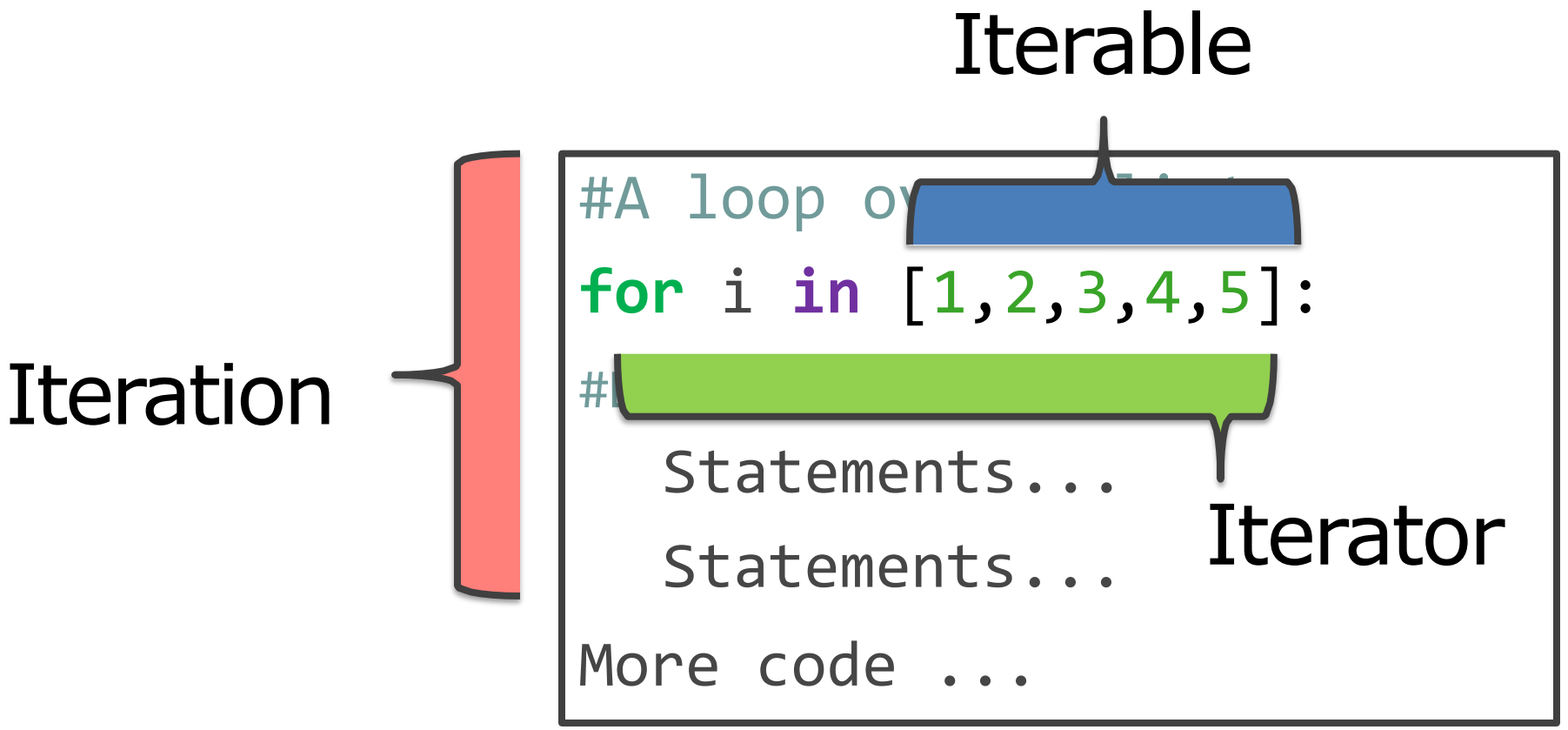
```
# Nest multiple lists
nested_a = [
           [10, 20, 30, 40, 50],
           [1, 2, 3, 4, 5],
           [15, 20, 25, 30, 35],
           [100, 200, 300, 400, 500]
           ]
```



```
# Nest 3-D multiple lists
nested_a = [
           [[10, 20, 30, 40, 50],
           [1, 2, 3, 4, 5],
           [15, 20, 25, 30, 35],
           [100, 200, 300, 400, 500]],
           ...
           ...]
```



Iteration, Iterator, Iterable


Iterable
Iteration
#A loop ov
for i in [1,2,3,4,5]:
#
Statements...
Statements...
Iterator
More code ...

Range function

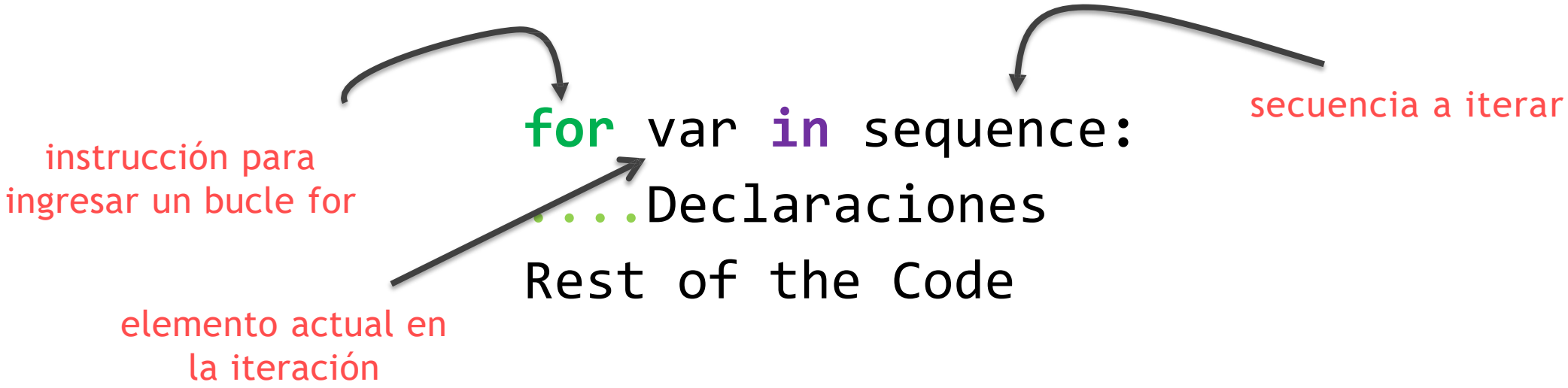
```
# Create a list of integers
new_list = list(range(10))
print(new_list)
>>> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```


```
range[start:stop:(Optional)step]
```

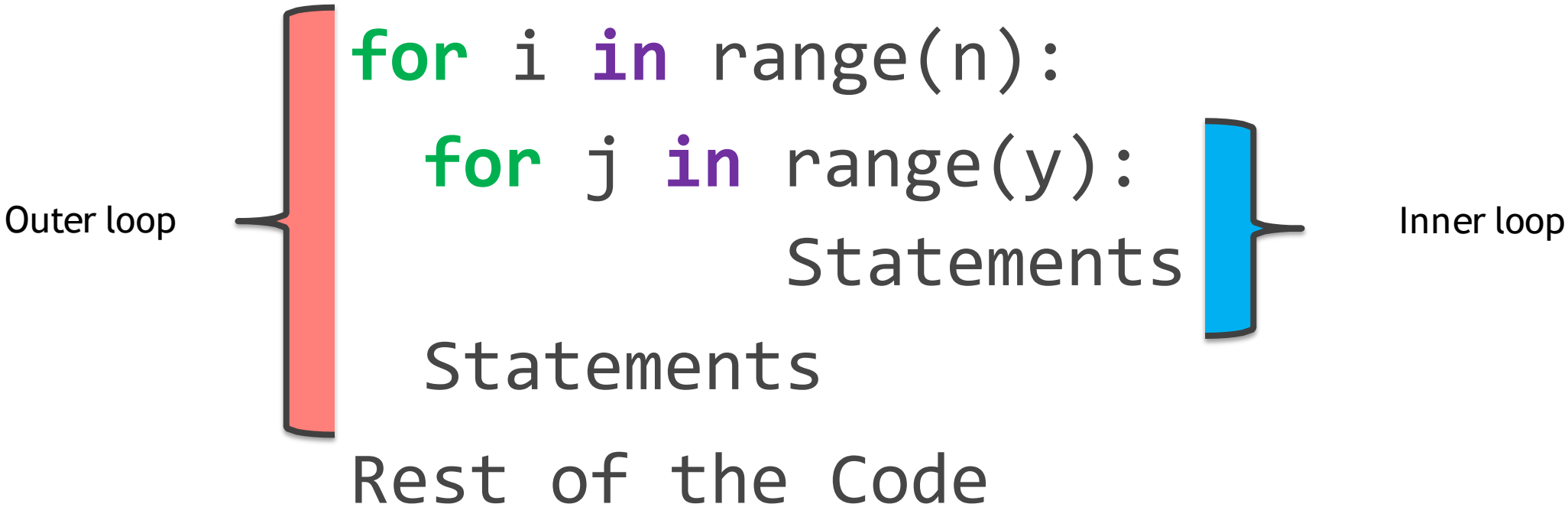
Loops

For loop

El bucle for es útil para iterar sobre los elementos de una secuencia, como string, list, tuple, range (), etc.



Nested for loops



While loop

El bucle *while* sigue repitiendo una acción hasta que una condición asociada devuelve falso

Los bucles while no son muy populares en Python, pero pueden ser útiles si la condición de parada no está bien ordenada.

El intérprete de Python comprueba la condición

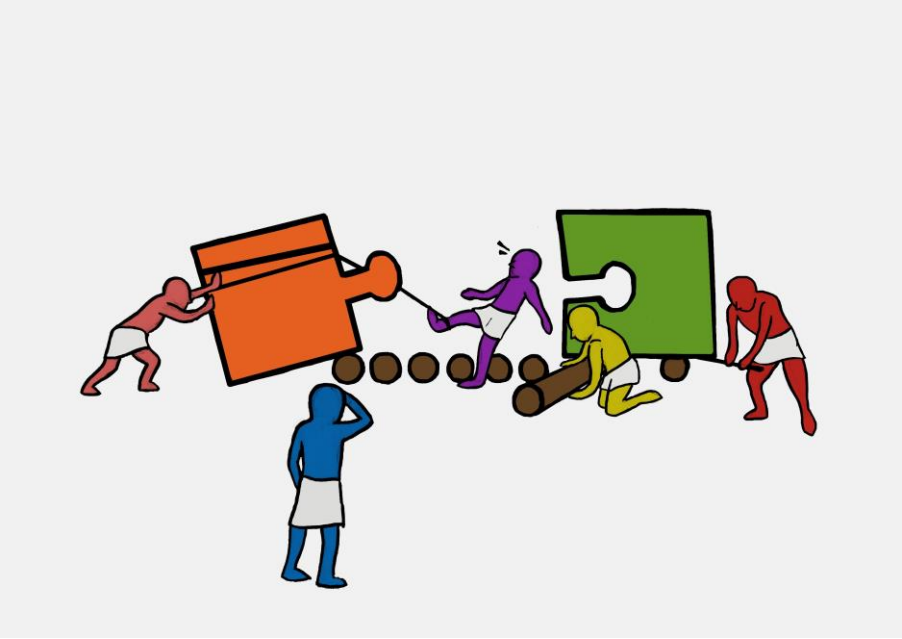
si es verdadero, ejecuta statements

```
while (condition):
        Statements
Rest of the Code
```

Infinite Loops

```
while True:
    print('Con un gran poder viene
    una gran responsabilidad ')
```

Ejercicio



Description

Exercise: Loops

The goal of this exercise is to learn to use `for` loops to manipulate lists.

Le dice a Python que queremos entrar en un bucle

La secuencia sobre la que queremos iterar

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

Representa el elemento actual en el que estamos en la iteración.

Proceso que queremos repetir una y otra vez

Data Structures

¿Por qué necesitamos estructuras de datos?

En una tierra lejana, donde abundaban los libros...

The Grand keeper de los libros

Oh no, ¿cómo voy a ordenar este caos?

Uh... Su Alteza, por supuesto. Sólo dame un momento.

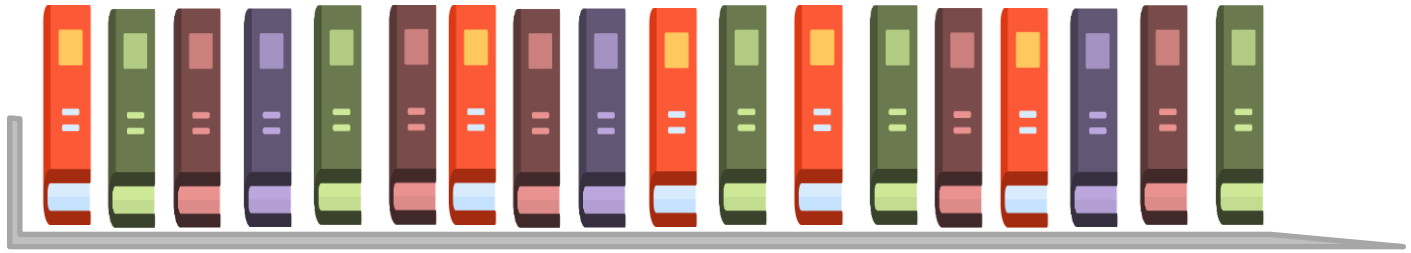
The Grand keeper, necesito el libro "Become a Data Scientist in 30 days"

¿Por qué necesitamos estructuras de datos?

The Great Keeper de los libros

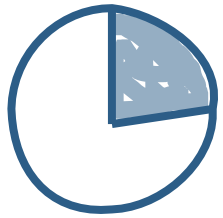
¡Eureka! Déjame usar mi magia para colocarlos en una fila – esto será ¡fácil!

Colocados como en una lista



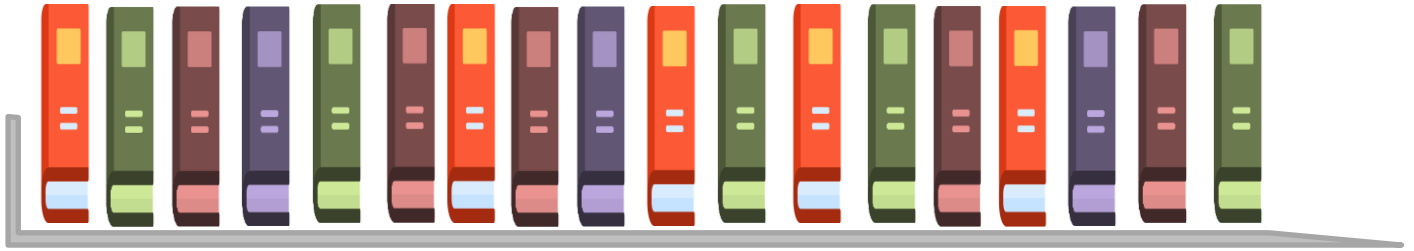
¿Por qué necesitamos estructuras de datos?

Buscando el libro



Tiempo de búsqueda

Colocados como en una lista



Encontré el libro!

¿Por qué necesitamos estructuras de datos?

The Grand Keeper of Books

¡Aquí tiene, su Alteza!

Colocados como en una lista

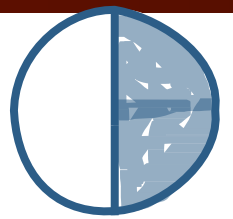
Necesito otro libro, "Zen of Python", ¿podría conseguírmelo ?

¿Por qué necesitamos estructuras de datos?

The Grand Keeper of Books

Colocados como en una lista

¿Por qué necesitamos estructuras de datos?



Colocados como en una lista

¡Encontré el libro!
Eso tomó mas tiempo

¿Por qué necesitamos estructuras de datos?

Colocados como en una lista

The Grand Keeper of Books

¡Aquí tiene, su Alteza!

¡Gracias otra vez!

...aunque, respetada Keeper, eso fue lento. ¡Padre te está enviando 10,000 libros más!

¿Por qué necesitamos estructuras de datos?

10,0000 libros!!!? ¿Cómo voy a organizarlos para que pueda obtener estos libros rápidamente?

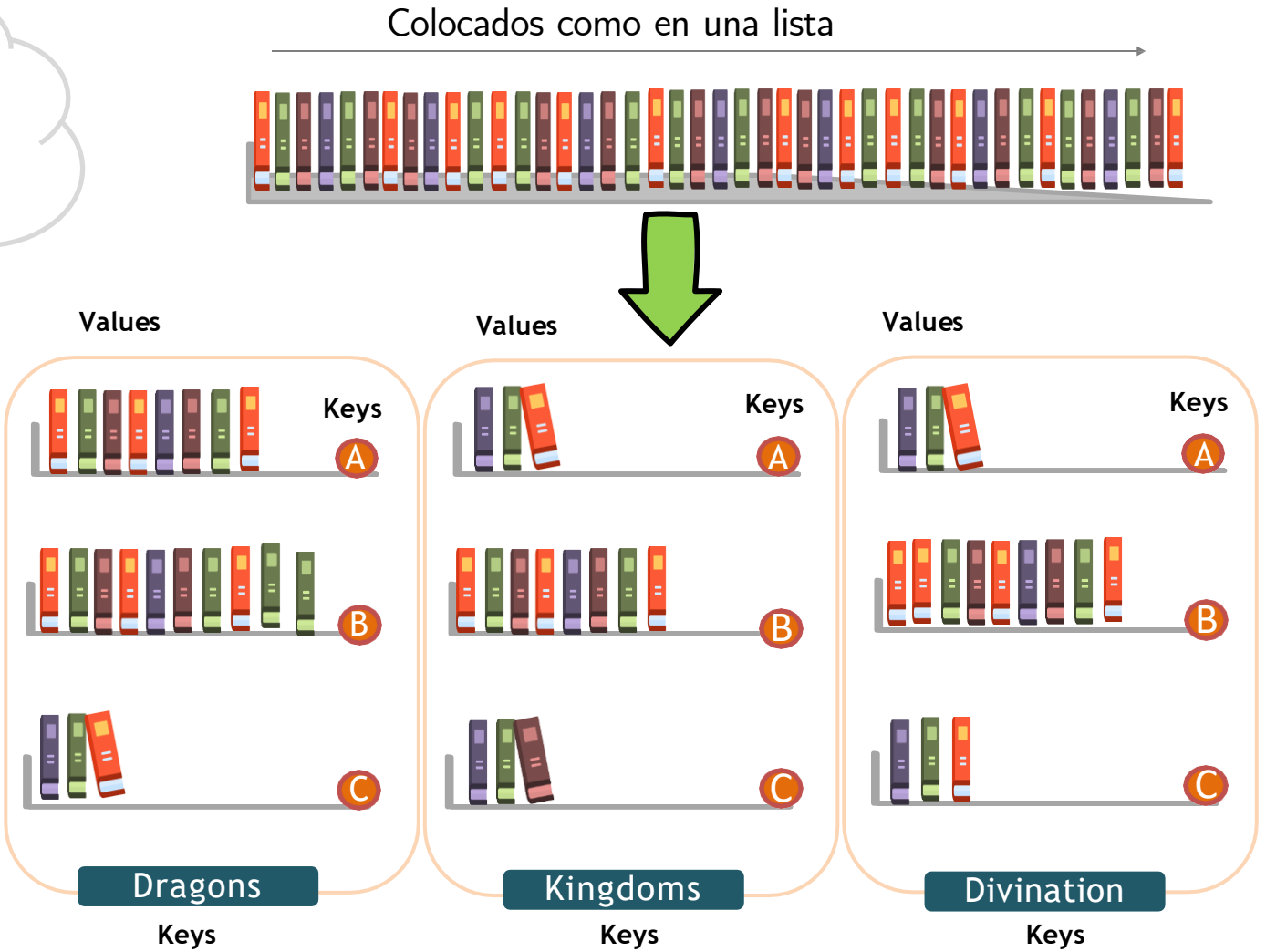
The Grand Keeper of Books

Colocados como en una lista

¿Por qué necesitamos estructuras de datos?

The Grand Keeper of Books

Déjame construir una biblioteca en su lugar

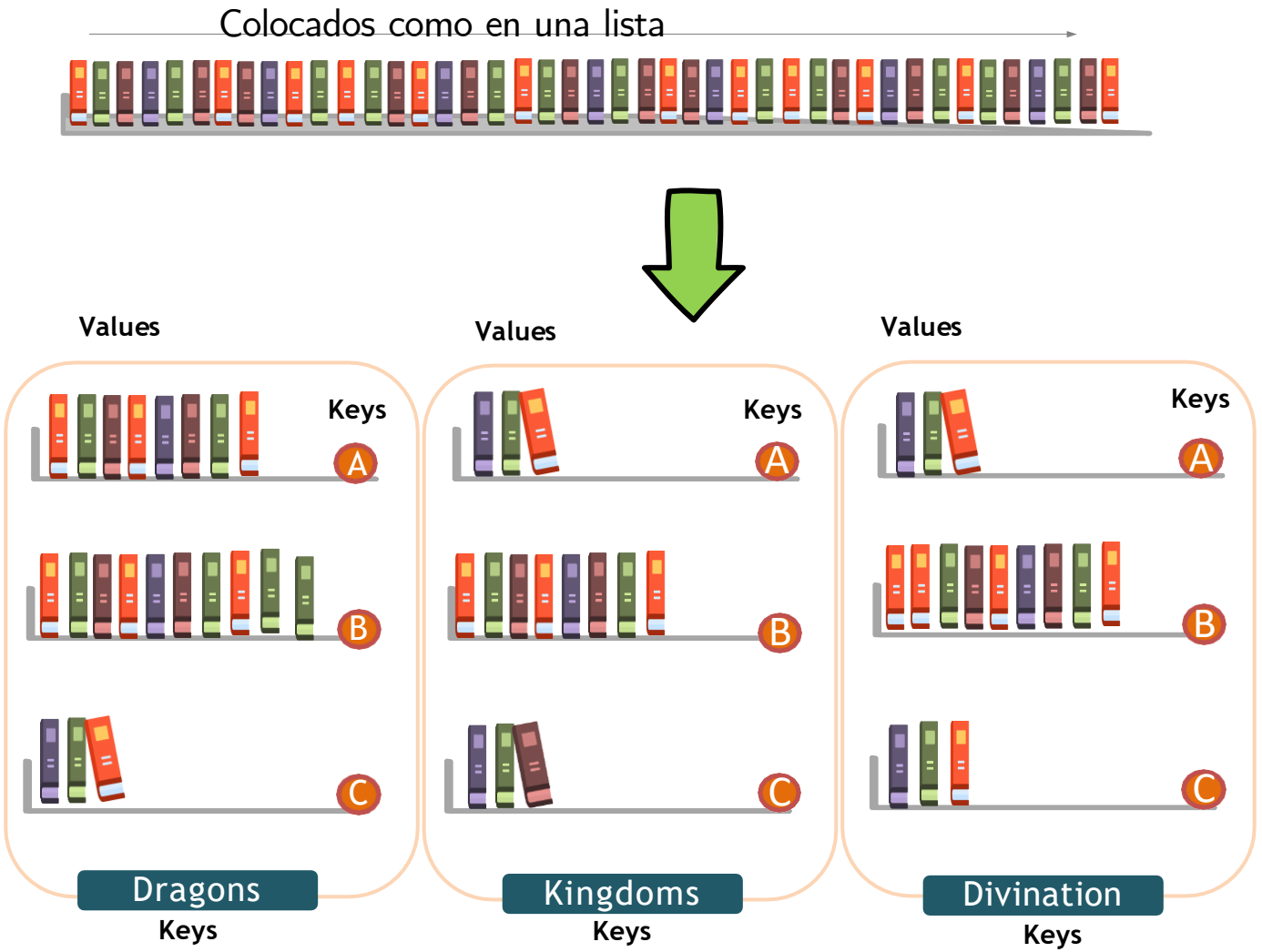


Este proceso se llama diccionarios en Python.

¿Por qué necesitamos estructuras de datos?

The Grand Keeper of Books

Déjame construir una biblioteca en su lugar



Este proceso se llama diccionarios en Python.


EXIT
VAIO

Diccionarios en Python

En Python, los diccionarios son una colección ordenada de objetos mutables con claves inmutables.
Los elementos dentro de un diccionario tienen una clave y un valor.

$$Dict = \{Key_1 : Value_1, ..., Key_n : Value_n\}$$

```
>>> dictA = {'India':'New Delhi', 'USA':'Washington
DC','Germany':'Berlin','Sri Lanka':'Colombo’}

>>> dictB = {'Apples':1, 'Pineapple':4, 'Grapes’:3}

>>> print(dictA)
{'India': 'New Delhi', 'USA': 'Washington DC', 'Germany': 'Berlin', 'Sri
Lanka': 'Colombo’}

>>> print(dictB)
{'Apples': 1, 'Pineapple': 4, 'Grapes': 3}
```

DictA

Key - Type	Value - Type
India<str>	New Delhi<str>
USA<str>	Washington DC<str>
Germany<str>	Berlin<str>

DictB

Key - Type	Value - Type
Apples<str>	1 <int>
Pineapple<str>	4 <int>
Grapes <str>	3 <int>

Diccionarios en Python

Métodos de diccionario importantes para recordar

Python Code	Function
`dictA = {'India':'Delhi','USA':'Washington'},` `dictA = {name='Sam', age=23}`	Crea un diccionario con el par key-value proporcionado

Diccionarios en Python

Métodos de diccionario importantes para recordar

Python Code	Function
`dictA = {'India':'Delhi','USA':'Washington'},` `dictA = {name='Sam', age=23}`	Crea un diccionario con el par key-value proporcionado
`dictA.items()`	Obtiene una lista de los pares "key-value" en el diccionario

Diccionarios en Python

Métodos de diccionario importantes para recordar

Python Code	Function
`dictA = {'India':'Delhi','USA':'Washington'},` `dictA = {name='Sam', age=23}`	Crea un diccionario con el par key-value proporcionado
`dictA.items()`	Obtiene una lista de los pares "key-value" en el diccionario
`dictA.values()`	Obtiene una lista de los valores en el diccionario

Diccionarios en Python

Métodos de diccionario importantes para recordar

Python Code	Function
`dictA = {'India':'Delhi','USA':'Washington'},` `dictA = {name='Sam', age=23}`	Crea un diccionario con el par key-value proporcionado
`dictA.items()`	Obtiene una lista de los pares "key-value" en el diccionario
`dictA.values()`	Obtiene una lista de los valores en el diccionario
`dictA.keys()`	Obtiene una lista de las keys en el diccionario

Diccionarios en Python

Métodos de diccionario importantes para recordar

Python Code	Function
`dictA = {'India':'Delhi','USA':'Washington'},` `dictA = {name='Sam', age=23}`	Crea un diccionario con el par key-value proporcionado
`dictA.items()`	Obtiene una lista de los pares "key-value" en el diccionario
`dictA.values()`	Obtiene una lista de los valores en el diccionario
`dictA.keys()`	Obtiene una lista de "keys" en el diccionario
`dictA['place of birth'] = 'Singapore'`	Establece el valor de "key" asociada a él
`del dictA['age']`	Elimina el par "key-value" del diccionario

Tuples en Python

En Python, las tuplas son colecciones ordenadas de objetos inmutables. Son como listas, pero la diferencia es que son inmutables y se encierran con ( ) en lugar de [ ].

$$\text{Tuple} = (e_1, e_2 \ldots e_N)$$

```
#Intialize a tuple
In [2]: tupleA = ('India','USA','Germany')

#Intialize another tuple
In [3]: tupleB = ('violet','indigo','blue','yellow','orange','red')

In [5]: tupleA + tupleB
Out[5]:
('India','USA','Germany','violet','indigo','blue','yellow','orange','red')
```

La ejecución del programa es más rápida cuando se manipula una tupla que cuando se manipula una lista equivalente.

Tuples en Python

Importante recordar: *dado que los paréntesis también se usan para definir la precedencia de los operadores en las expresiones, Python evalúa un número dentro de ( ) como un objeto de su tupla. Pero si quiere decirle a Python que defina una tupla singleton, debe poner una coma.*

```
In [7]: a = (2,3)

In [8]: b = (2)

In [9]: c = ('string')

In [10]: d = (2,)

In [11]: e,f = 2,3
```

Tipo de Variable

Type(a) - tuple

Type(b) - integer

Type(c) - string

Type(d) - tuple

Type(e) - integer

Type(f) - integer