Fundamentos básicos del lenguaje Python

Condicionales

Universidad Privada Boliviana

Hugo Condori Quispe, Ph.D.

hugo.condori@fulbrightmail.org

Control de flujo en Python

Bloque: If... Else

Primero se prueba la condición

```
if (condition):
    declaración1
    declaración2
else:
    declaración3
Resto del Código
```

si la condición es verdadera, se ejecutan las declaraciones dadas después de los dos puntos (:)

else si la condición no es verdadera, se ejecutan las declaraciones después del **bloque else**.

Bloque: If... Else

las condiciones se pueden agrupar usando operadores lógicos

```
if (condition):
    declaración1
    declaración2
else:
    declaración3
Resto del Código
```

si la condición es verdadera, se ejecutan las declaraciones dadas después de los dos puntos (:)

else si la condición es verdadera, se ejecutan las declaraciones después del **bloque else**.

Bloque: If... Else

las condiciones se pueden agrupar usando operadores lógicos

ndición

```
if (condition):
    declaración1
else:
    declaración3
Resto del código
```

El bloque else no es obligatorio

si la condición es verdadera, se ejecutan las declaraciones dadas después de los dos puntos (:)

else si la condición es verdadera, se ejecutan las declaraciones después del **bloque else**.

Bloque: If... Else

La indentación es obligatoria en python (estándar de 4 espacios/1 tab)

```
if (condition):
    statement1
    statement2
else:
    statement3
Rest of the Code
```

Sintaxis

¿Cuál es la sintaxis correcta para una declaración if en Python?

a) `if (condition)`
b) `if [condition]`
c) `if condition`
d) `if condition:`

- ¿Qué ocurre si la condición en un if statement es falsa y no hay un bloque else?

a) El programa genera un error.
b) El programa se detiene y no se ejecuta más código.
c) El programa continúa ejecutando el código después del if statement.
d) Ninguna de las anteriores.

- ¿Cuál es el resultado de la siguiente condición en Python?

```
if 5 > 10:
    print("Verdadero")
else:
  print("Falso")
```

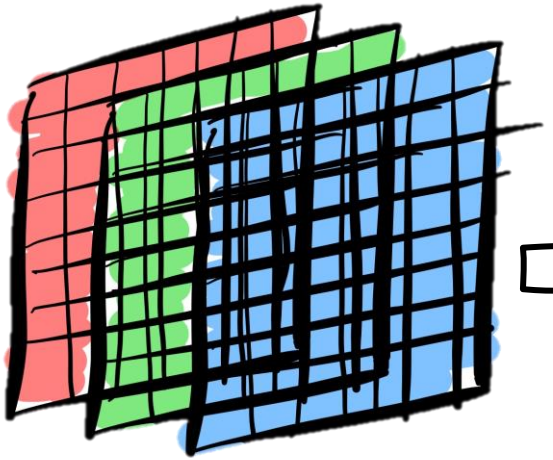
a) Verdadero
b) Falso
c) Error de sintaxis
d) Ninguna de las anteriores.

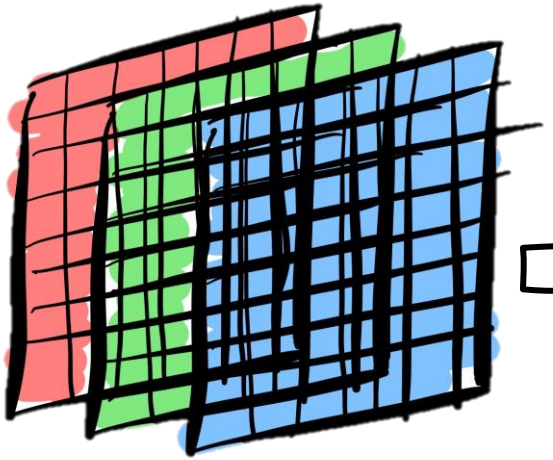
- ¿Qué bloque de código se ejecuta si la condición en un bloque if es falsa y hay un bloque else asociado?

a) Bloque if
b) Bloque else
c) Ambos bloques if y else
d) Ninguno de los bloques se ejecuta
e) Depende de la condición en el else

Listas

Imágenes representadas por pixeles





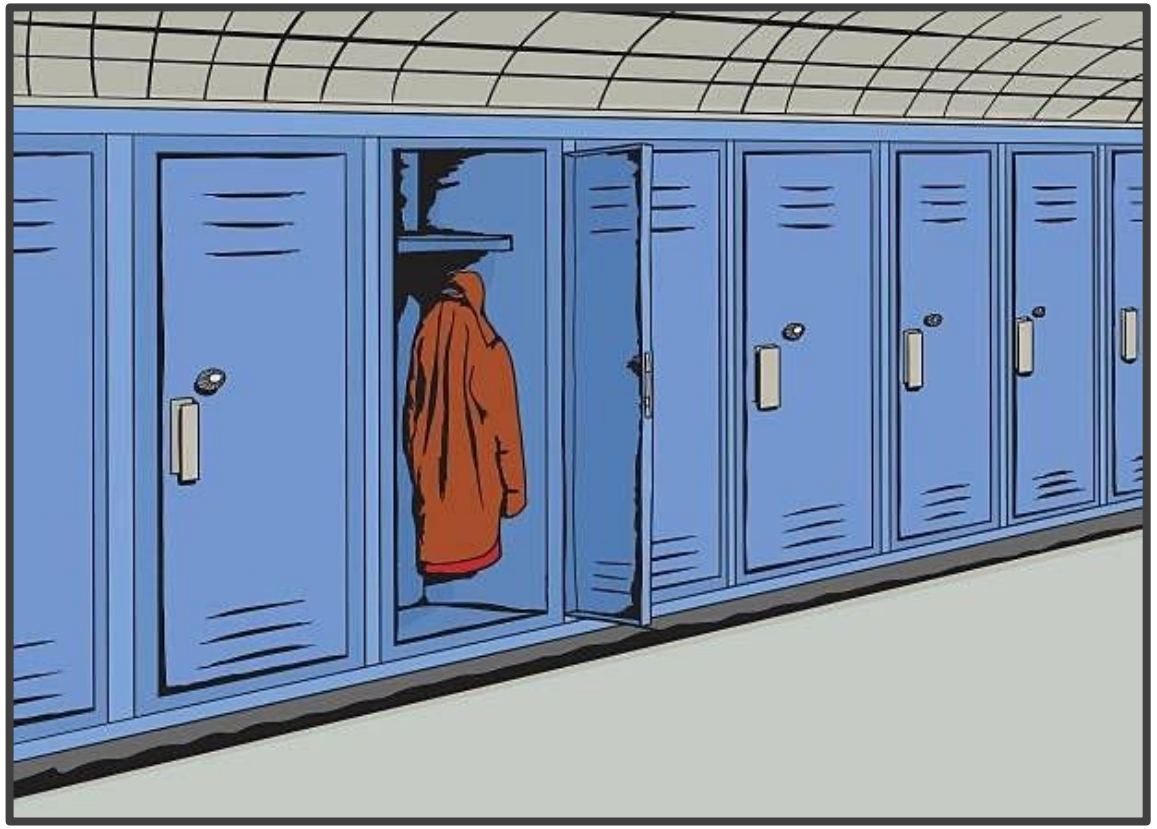
```
array([[[ 63,  95, 208, 231, 190, 186, 196, 208,  91, 110,  31,  66,
         163,  75,  96,  84,  86, 106,   0,  57,  90, 147,  31, 148,
          63,  10,  61, 231,  80, 127, 174, 240,  45,  56, 153, 193,
         238,  85,  38, 118,  94, 145, 144, 192,  82,  48, 145,  17,
         176, 150,  43, 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,
```

List [ ]

¿Qué es una lista de Python?

1. Una colección ordenada
2. eso es re-dimensionable
3. Y contiene elementos de diferentes tipos.

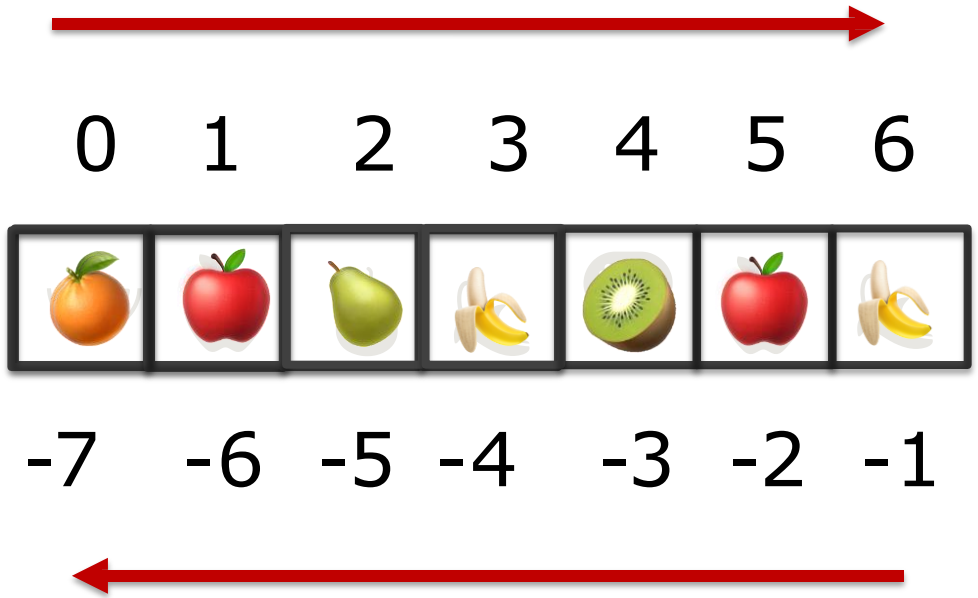


```
#Create a list
num_list = [1,2,3]

#A list can have a mixture of all types
mix_list = ['hello’, 1, True]
```

List indexing

¿Cómo accedemos a los elementos de una lista?

```
# Make a list of fruits
>>> frutas = ['mandarina', 'manzana',
'pera', 'plátano', 'kiwi', 'manzana',
'plátano']
```

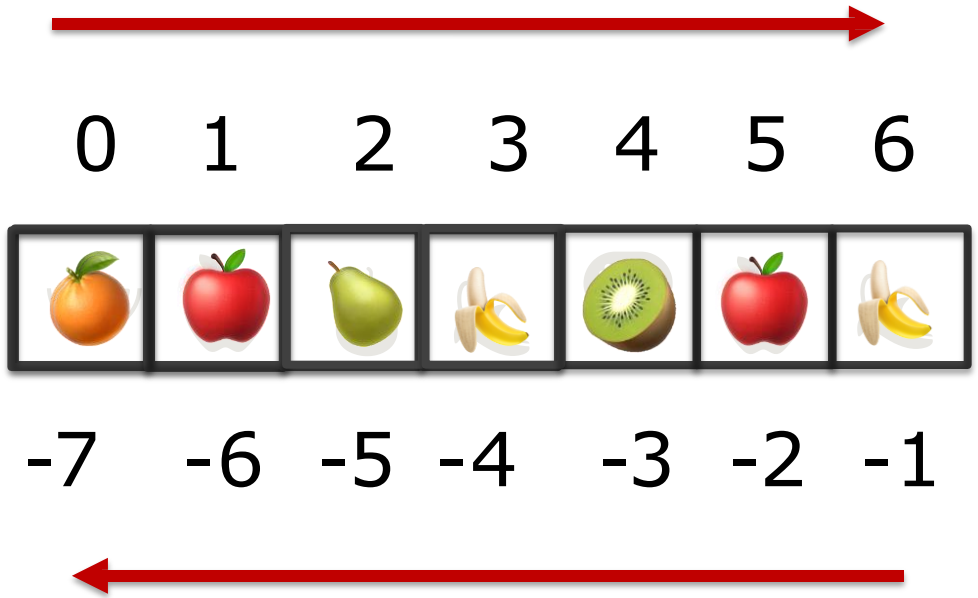



```
List[start:stop:(Optional)step]
```

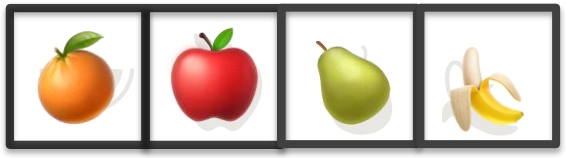
List indexing

¿Cómo accedemos a los elementos de una lista?

```
# Make a list of fruits
>>> frutas = ['mandarina', 'manzana',
'pera', 'plátano', 'kiwi', 'manzana',
'plátano']
```



`fruits[0:4] =`

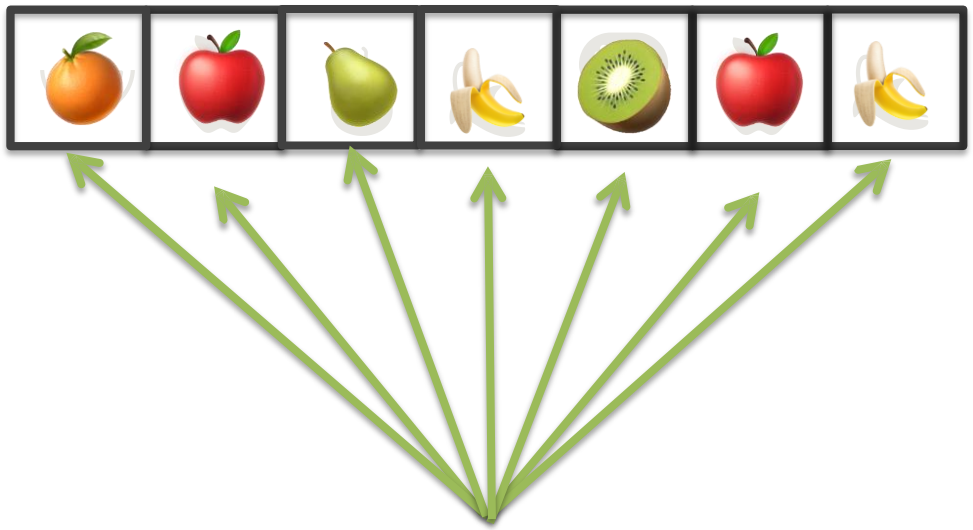


List methods

```
# Make a list of fruits
>>> fruits = ['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

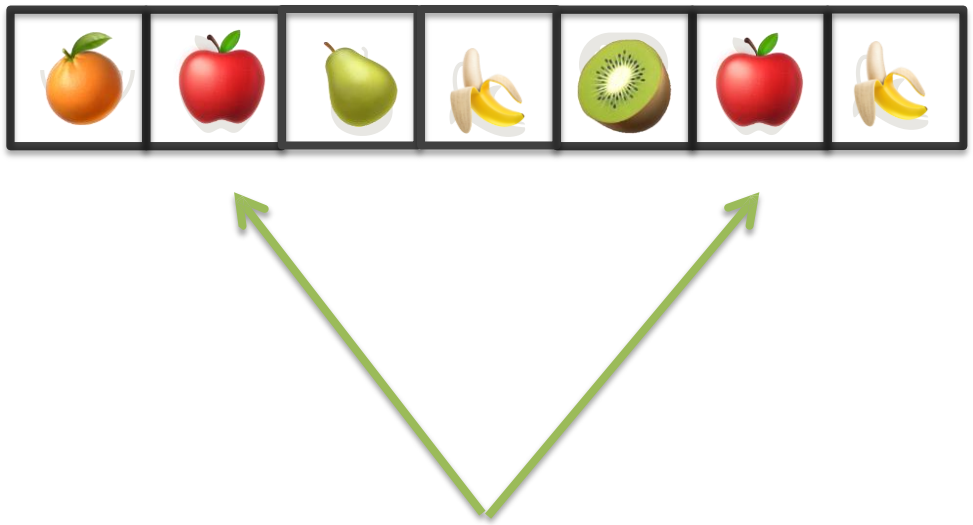
len()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']

>>> len(fruits)
7
```



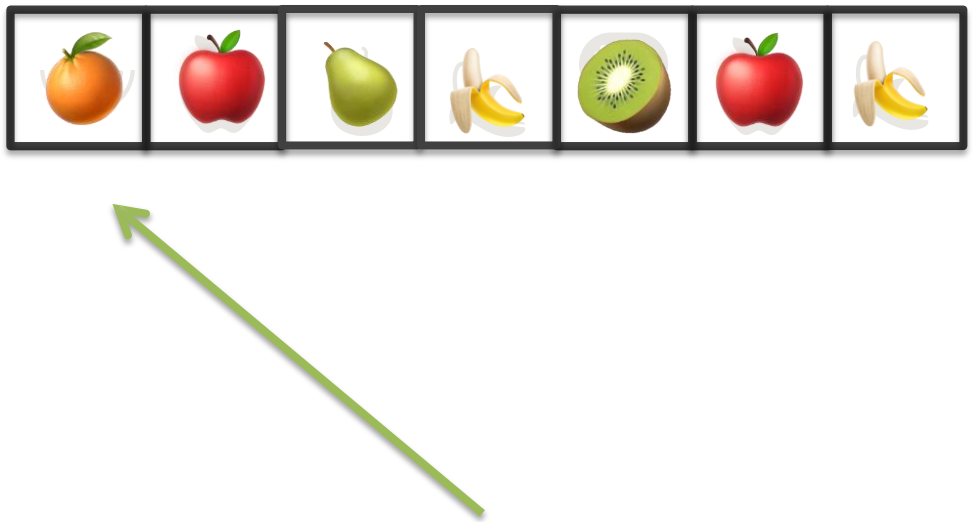
Count()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']

>>> fruits.count('manzana')
2
```



Count()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']

>>> fruits.count(mandarina')
 1
```



- ¿Cuál es la forma correcta de crear una lista vacía en Python?
    a) lista = []
    b) lista = {}
    c) lista = ()
    d) lista = [None]
    e) Ninguna de las anteriores

- ¿Cuál de las siguientes afirmaciones es cierta acerca de las listas en Python?

  a) Las listas pueden contener elementos de diferentes tipos de datos.

  b) Las listas deben contener solo números enteros.

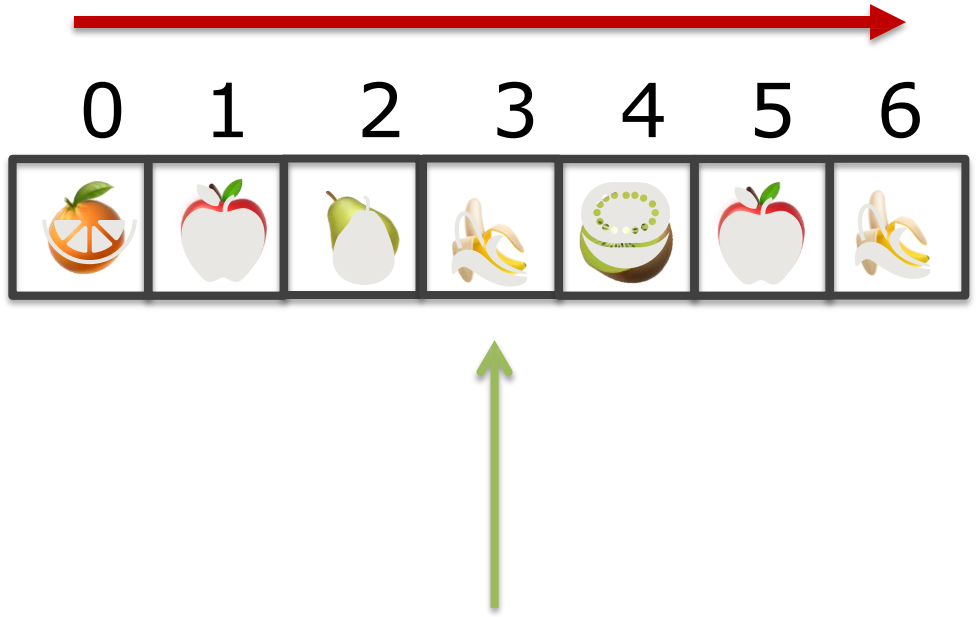
  c) Las listas no pueden contener más de 10 elementos.

  d) Las listas siempre se ordenan automáticamente de forma ascendente.

  e) Ninguna de las anteriores

Index()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']


>>> fruits.index('platano')

 3
```



- ¿Cuál es la sintaxis correcta para acceder al tercer elemento de una lista llamada mi_lista?

  a) mi_lista[2]

  b) mi_lista[3]

  c) mi_lista(2)

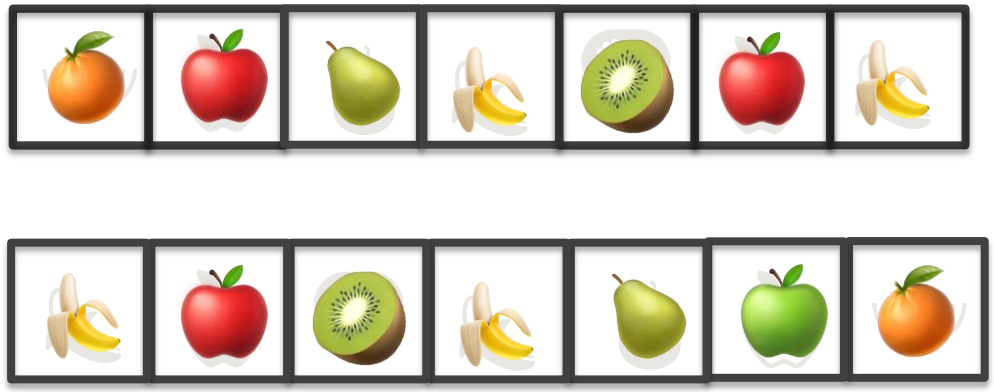
  d) mi_lista(3)

  e) Ninguna de las anteriores

- ¿Cuál es la sintaxis correcta para acceder al penúltimo elemento de una lista llamada mi_lista?
    a) mi_lista[-1]
    b) mi_lista[-2]
    c) mi_lista(6)
    d) mi_lista(7)
    e) Ninguna de las anteriores

Reverse()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']
>>> fruits.reverse()
>>> fruits
['plátano', 'manzana', 'kiwi', 'plátano', 'pera', 'manzana', 'mandarina']
```



List methods – sort()

```
# Make a list of fruits
>>> fruits = ['plátano', 'manzana', 'kiwi', 'plátano', 'pera', 'manzana', 'mandarina']
```

- ¿Cuál de los siguientes métodos de lista en Python se utiliza para obtener el índice de la primera aparición de un elemento en la lista?
  a) `find`
  b) `search`
  c) `index`
  d) `position`

List methods – sort()

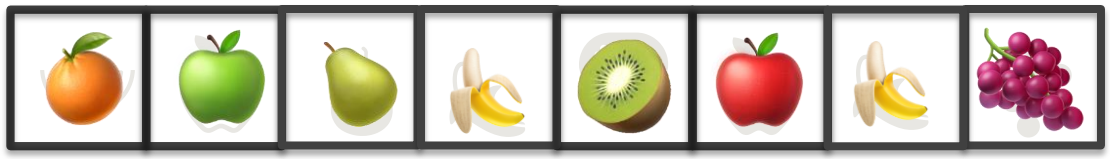
```
# Make a list of fruits
>>> fruits = ['plátano', 'manzana', 'kiwi', 'plátano', 'pera', 'manzana',
'mandarina']

>>> fruits.sort()
>>> fruits
['kiwi', 'mandarina', 'manzana', 'manzana', 'pera', 'plátano', 'plátano']
```

List methods – append()

```
# Make a list of fruits
>>> fruits = ['mandarina', 'manzana', 'pera', 'plátano', 'kiwi', 'manzana', 'plátano']
```

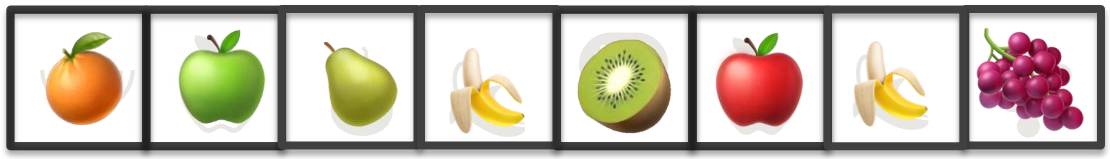
List methods – append()

```
# Make a list of fruits
>>> fruits = ['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']

>>> fruits.append('grape')
>>> fruits
['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana', 'grape']
```



List methods – pop()

```
>>> fruits.pop()
```



List methods – pop()

```
>>> fruits.pop()
>>> fruits
['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

List methods

```
# Make a list of fruits
>>> fruits = ['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

List methods – insert()

```
# Make a list of fruits
>>> fruits = ['tangerine', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']

>>> fruits.insert(1,'grape')
>>> fruits
['tangerine', 'grape', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

List methods – remove()

```
>>> fruits.remove('grape')
>>> fruits
['tangerine','apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

- Supongamos que tienes una lista llamada numeros con los siguientes valores: [3, 1, 4, 1, 5, 9, 2, 6, 5]. Si aplicas el método sort a esta lista, ¿cuál será el resultado?

  a) [1, 1, 2, 3, 4, 5, 5, 6, 9]

  b) [9, 6, 5, 5, 4, 3, 2, 1, 1]

  c) [3, 1, 4, 1, 5, 9, 2, 6, 5]

  d) [1, 2, 3, 4, 5, 5, 6, 9]

- ¿Cuál es el propósito principal del método index en una lista en Python?

    a) Revertir el orden de la lista

    b) Encontrar el índice de un elemento en la lista

    c) Eliminar elementos duplicados de la lista

    d) Ordenar la lista en orden ascendente

- ¿Qué método se utiliza para eliminar el último elemento de una lista en Python?

  a) pop()

  b) remove()

  c) delete()

  d) del()

  e) Ninguna de las anteriores

- Si tienes una lista llamada frutas con los siguientes valores: ['manzana', 'plátano', 'naranja'], ¿qué hace el método pop cuando se llama sin argumentos en esta lista?

    a) Agrega un elemento a la lista

    b) Elimina el último elemento de la lista y lo devuelve

    c) Revierte el orden de los elementos en la lista

    d) Ordena la lista en orden ascendente

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 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'
```