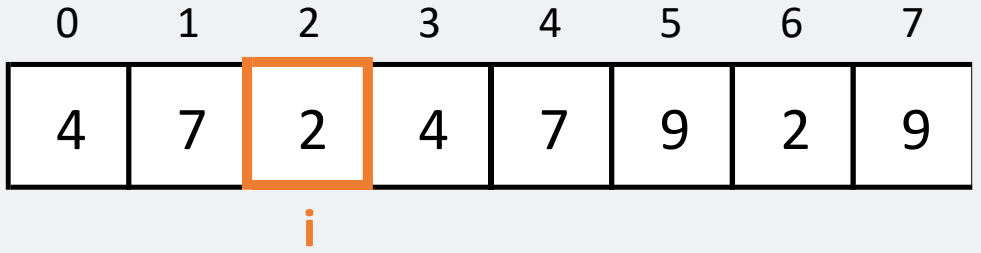
NIVEL 4

TUPLAS

Universidad de los Andes

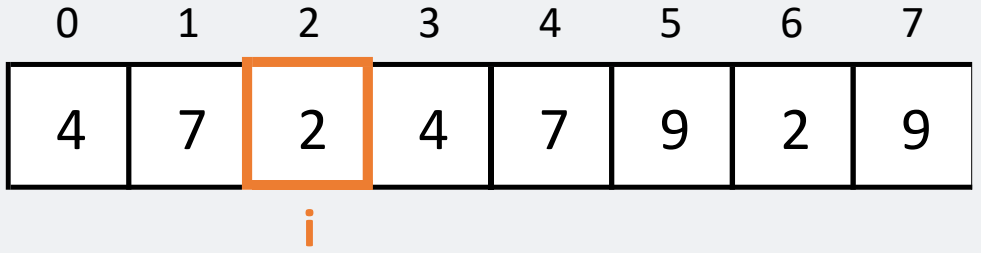
RECORDEMOS LAS LISTAS



- Estructura de datos lineal
- Cada elemento se identifica con un índice o posición
- Mutable: se pueden agregar, eliminar o modificar elementos

```
In [1]: lista = [4, 7, 2, 4, 7, 9, 2, 9]

In [2]: lista
Out[2]: [4, 7, 2, 4, 7, 9, 2, 9]

In [3]: lista[2] = 99

In [4]: lista.append(-1)

In [5]: lista
Out[5]: [4, 7, 99, 4, 7, 9, 2, 9, -1]
```

TUPLAS EN PYTHON

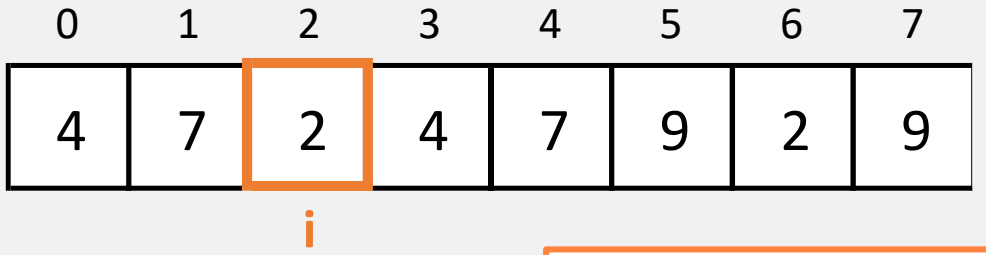


- Estructura de datos lineal
- Cada elemento se identifica con un índice o posición
- Inmutables: no se pueden modificar los elementos después de creados

```
In [6]: tupla = (4, 7, 2, 4, 7, 9, 2, 9)

In [7]: tupla
Out[7]: (4, 7, 2, 4, 7, 9, 2, 9)

In [8]: type(tupla)
Out[8]: tuple
```

El nombre del tipo de dato en Python es tuple

TUPLAS EN PYTHON



- Estructura de datos lineal
- Cada elemento se identifica con un índice o posición
- Inmutables: no se pueden modificar los elementos después de creados

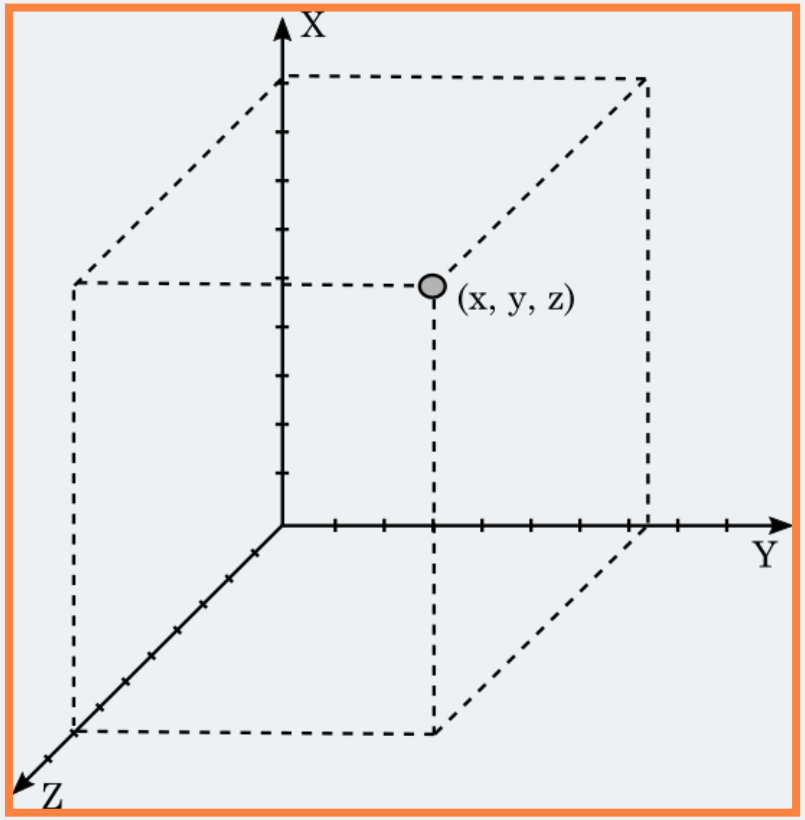
```
In [6]: tupla = (4, 7, 2, 4, 7, 9, 2, 9)

In [7]: tupla
Out[7]: (4, 7, 2, 4, 7, 9, 2, 9)

In [8]: type(tupla)
Out[8]: tuple
```

```
In [9]: tupla[1] = 99
Traceback (most recent call last):

  File "<ipython-input-9-249ea0827504>", line 1, in <module>
    tupla[1] = 99

TypeError: 'tuple' object does not support item assignment
```

PARA QUÉ LAS TUPLAS

- Agrupar una cantidad fija de ítems (datos) en un solo valor compuesto

Inmutabilidad == garantía



CREACIÓN DE TUPLAS

Las tuplas se crean de forma similar a las listas, pero se deben usar paréntesis redondos en lugar de cuadrados.

(,)

```
In [10]: coordenada = (3, 6, 3)

In [11]: coordenada
Out[11]: (3, 6, 3)

In [12]: type(coordenada)
Out[12]: tuple
```

CREACIÓN DE TUPLAS: EMPAQUETADO

```
In [13]: coordenada = 3, 6, 3

In [14]: coordenada
Out[14]: (3, 6, 3)

In [15]: type(coordenada)
Out[15]: tuple
```

- Los paréntesis en realidad son opcionales para crear una tupla.
- Si a una variable se le asigna una secuencia de valores separados por comas, el valor de esa variable será la tupla formada por todos los valores asignados.
- A esta operación se la denomina empaquetado.

TUPLAS COMPLEJAS

- Tuplas dentro de tuplas
- Tuplas con valores de tipos diferentes

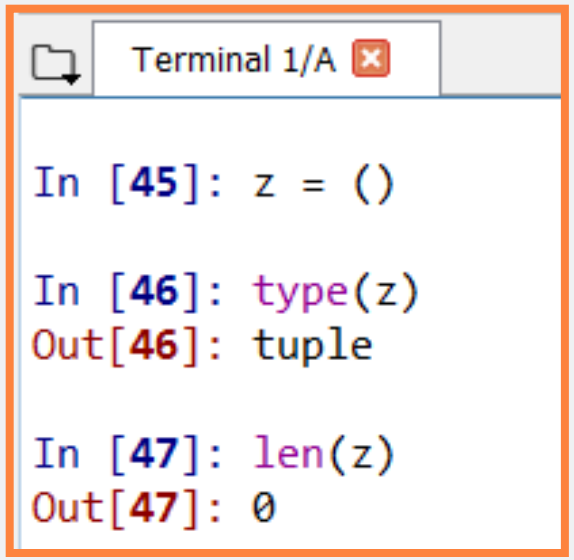
```
In [24]: x = 2

In [25]: y = 3

In [26]: ancho = 7

In [27]: alto = 4

In [28]: rectangulo = ((x, y), (ancho, alto), "rojo")

In [29]: rectangulo
Out[29]: ((2, 3), (7, 4), 'rojo')
```

N4

TUPLAS DE TAMAÑOS ESPECIALES

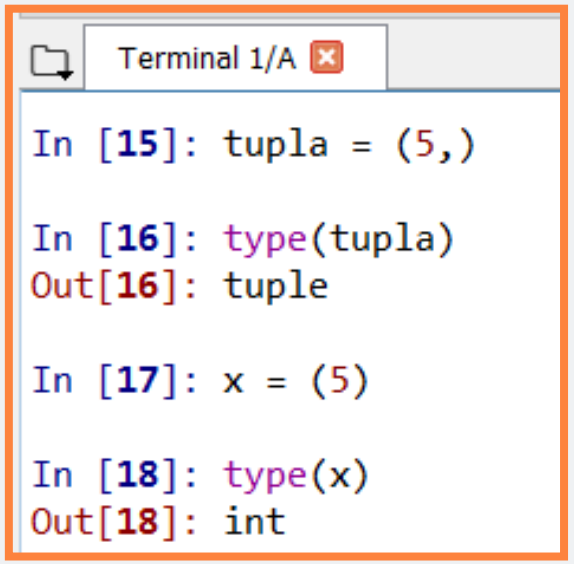
VACÍA

Una tupla vacía es una tupla con ∅ componentes, y se indica como ()



UNITARIA

Para crear una tupla con un solo elemento, incluimos la coma final. Sin esta coma, Python pensará que es un valor simple entre paréntesis:



OPERACIONES SOBRE TUPLAS

Las tuplas soportan las mismas operaciones de consulta que las listas.

El operador de indexación selecciona un elemento de una tupla usando [ ]:

```
In [32]: coordenada[1]
Out[32]: 6
```

Podemos aplicar también todo el poder de los slices:

```
In [30]: rectangulo[0:2]
Out[30]: ((2, 3), (7, 4))

In [31]: rectangulo[-1]
Out[31]: 'rojo'
```

Para consultar el tamaño de una tupla usamos la función len:

```
In [36]: len(rectangulo)
Out[36]: 3

In [37]: len(rectangulo[0])
Out[37]: 2
```

DESEMPAQUETADO DE TUPLAS

- Si se tiene una tupla con k elementos, se puede asignar la tupla a k variables distintas y en cada variable quedará uno de los componentes de la tupla.
- A esta operación se la denomina desempaquetado de tuplas.
- Si no se usan exactamente k variables (la misma cantidad que elementos en la tupla), se producirá un error.

```
In [38]: esquina, dimension, color = rectangulo

In [39]: esquina
Out[39]: (2, 3)

In [40]: dimension
Out[40]: (7, 4)

In [41]: color
Out[41]: 'rojo'
```

TUPLAS COMO VALORES DE RETORNO

- Una función solo pueden devolver un valor, pero si ese valor en una tupla, podremos agrupar todos los valores que queramos para devolverlos juntos.
- Esto se puede combinar con el desempaquetado de tuplas.

```python
def calcular_area_y_perimetro(rectangulo: tuple) -> tuple:
    dimension = rectangulo[1] #  Extraer la dimensión del rectángulo
    ancho, alto = dimension   #  Desempaquetar la dimensión para
                              #  extraer sus componentes
    area = ancho * alto
    perimetro = (ancho + alto) * 2

    return (area, perimetro)  # Retornar una tupla con los dos valores

# Invocar la función y guardar el resultado en variables separadas
area, perimetro = calcular_area_y_perimetro(rectangulo)
```

TUPLAS COMO PARÁMETROS

Las tuplas también pueden usarse como parámetros de una función para disminuir la cantidad de parámetros y para facilitar la comprensión

```
def calcular_distancia(punto_1: tuple, punto_2: tuple) -> float:
    delta_x = abs(punto_1[0] - punto_2[0])
    delta_y = abs(punto_1[1] - punto_2[1])
    distancia = math.sqrt(delta_x ** 2 + delta_y ** 2)
    return distancia
```

```
In [54]: p1 = (1,2)

In [55]: p2 = (4,6)

In [56]: calcular_distancia(p1, p2)
Out[56]: 5.0
```