Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Conversión entre Contenedores

Cadenas, Listas, Tuplas, Conjuntos y Diccionarios

Hechos QUE CONECTAN

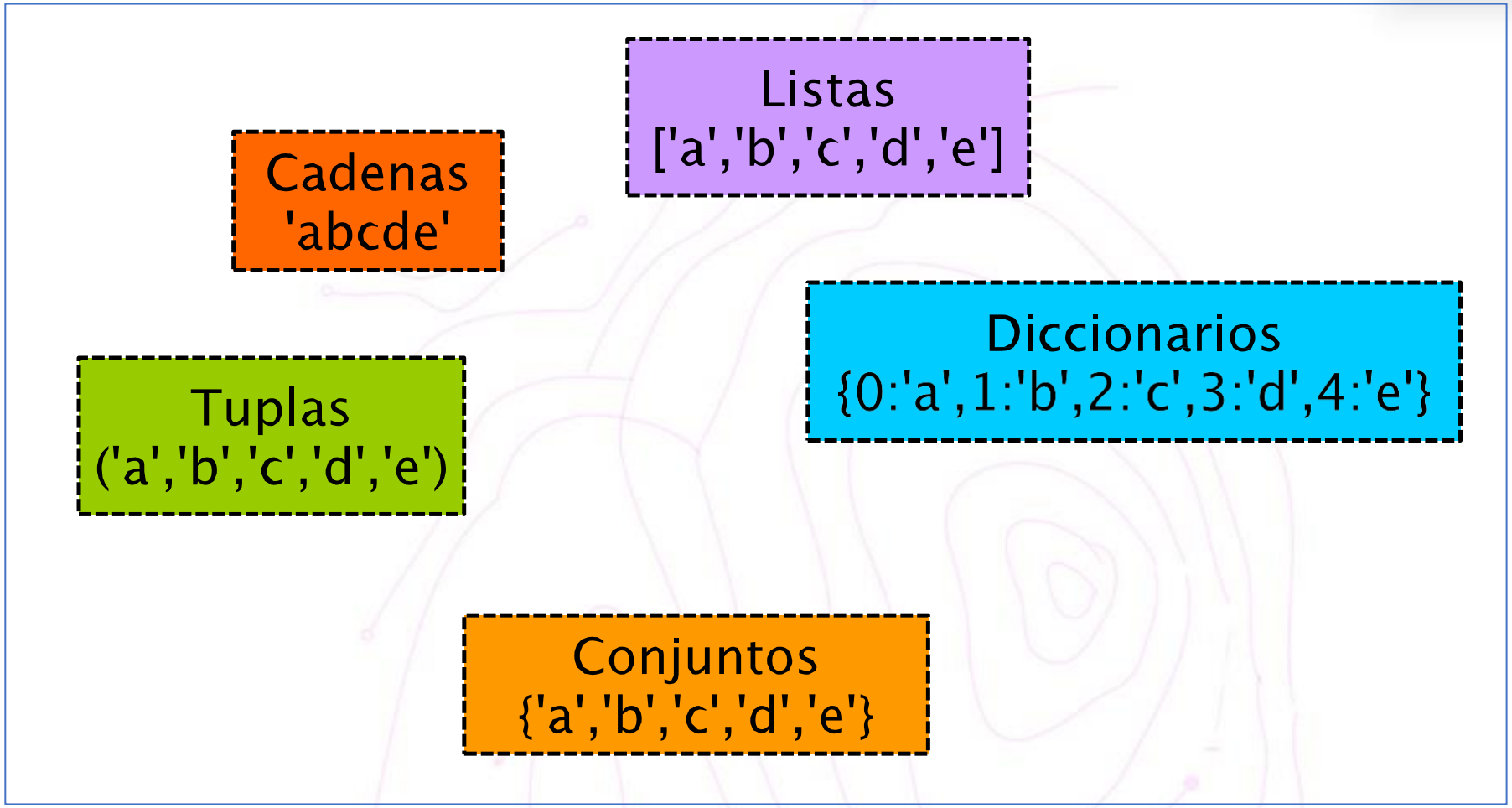
Introducción

- En la Unidad 3, hemos trabajado con varios contenedores, además hemos aprendido a diferenciar ***cuándo pueden ajustarse mejor a nuestras necesidades.***

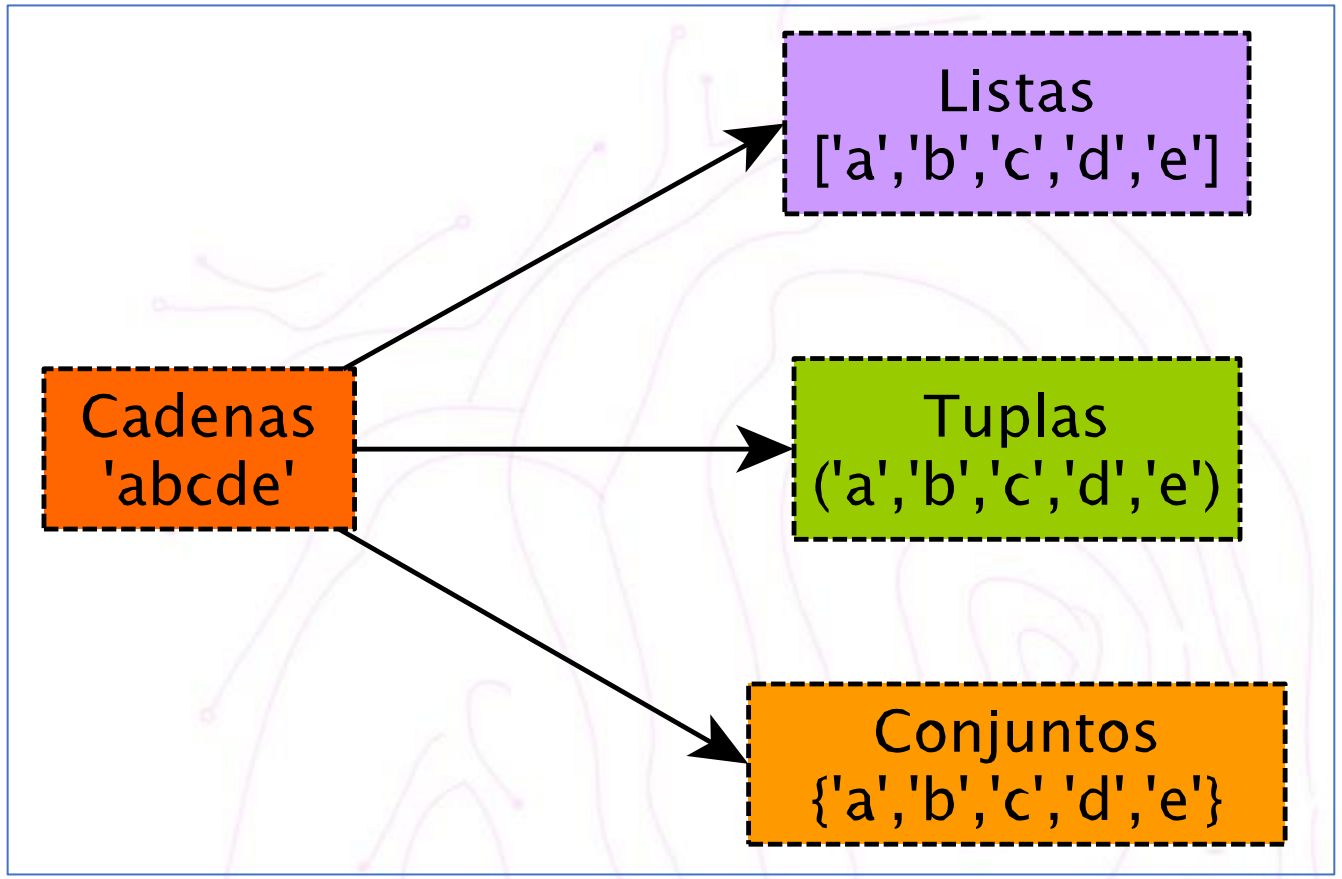
- Una de las grandes fortalezas del lenguaje Python, es la gran flexibilidad que presentan los contenedores para su creación y actualización.

- Adicionalmente, en Python es posible ***transformar un contenedor cargado de información, a otro tipo de contenedor*** en caso de que necesitemos cambiar la forma como manipulamos la información que alojan.

Introducción

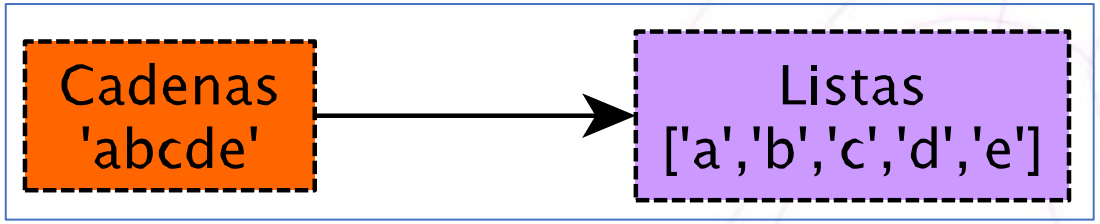


Conversión de Cadenas



Conversión de Cadenas

- Para aprovechar la mutabilidad de las listas, una cadena puede ser fácilmente convertida a una lista, conservando la notación de acceso a los elementos.

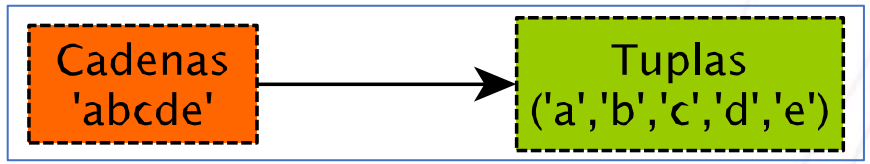



```
1   cadena = "hola"
2   lista = list(cadena)
3   print(lista)
```

```
['h', 'o', 'l', 'a']
```

Conversión de Cadenas

- Aunque las cadenas son inmutables, podemos convertirlas en tuplas, las cuales presentan un rendimiento mayor que otras formas de coleccionar elementos, en caso de necesitar transportar los caracteres de la cadena, o agruparlos con otros elementos para paso de parámetros.

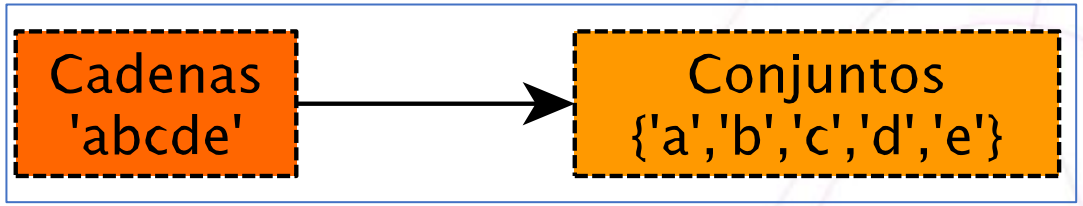



```
1  cad = "hola"
2  cad2 = "adiós"
3  num = 15
4  tupla1 = tuple(cad)
5  tuplaGeneral = (tupla1,num,tuple(cad2))
6  print(tuplaGeneral)
```

```
(('h', 'o', 'l', 'a'), 15, ('a', 'd', 'i', 'ó', 's'))
```

Conversión de Cadenas

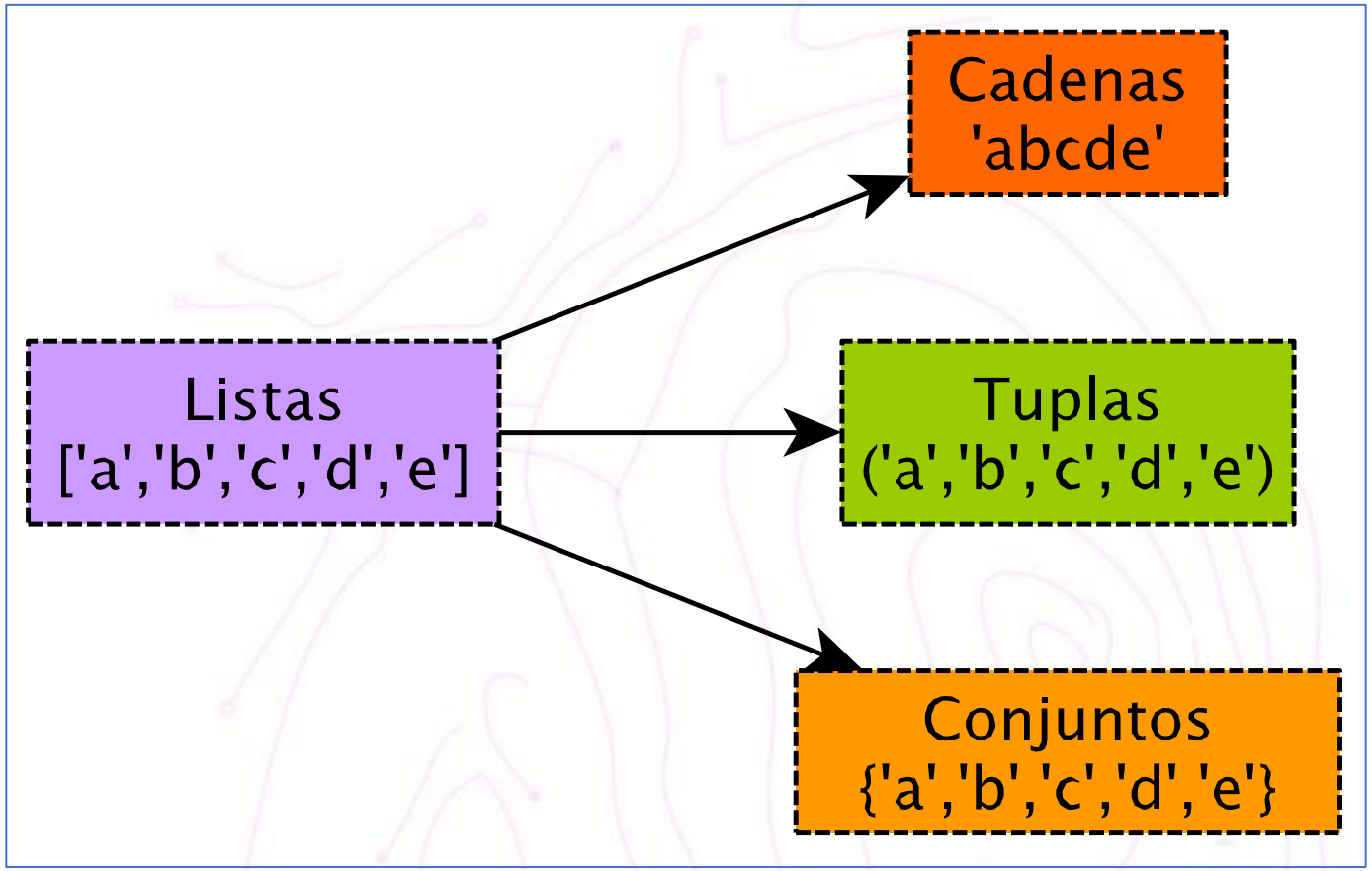
- Podemos convertir una cadena en un conjunto, y aprovechar que naturalmente este contenedor elimina las repeticiones, sin embargo, no conserva el orden de los elementos.




```
1    cadena = "hhola"
2    print(set(cadena))
```

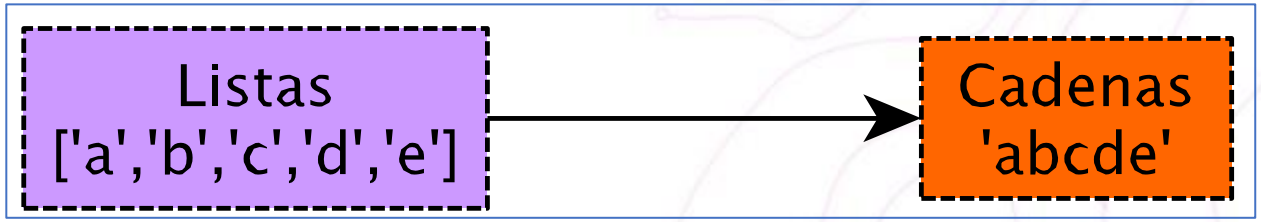
```
{'o', 'a', 'h', 'l'}
```

Conversión de Listas



Conversión de Listas

- A través del método **join** de las cadenas, todos los elementos de una lista pueden ser concatenados en una cadena, en caso de que sea necesario coleccionarlos de esta forma.

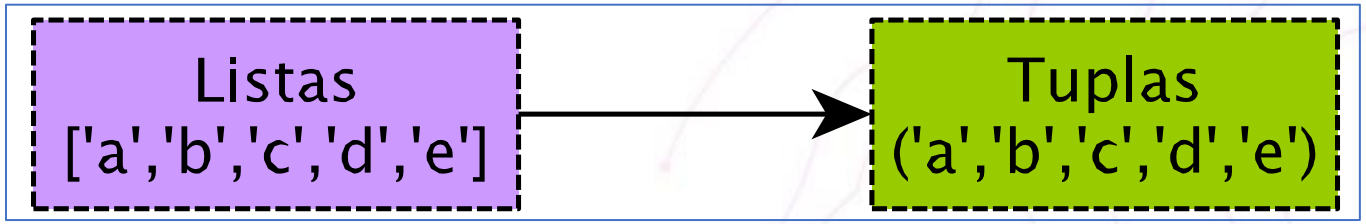



```
1   lista = ['h', 'o', 'l', 'a']
2   cadena = ''.join(lista)
3   print(cadena)
```

```
hola
```

Conversión de Listas

- Cuando queremos evitar modificaciones en los elementos de una lista, podemos convertirla en una tupla, aprovechando además tiempos de respuesta menores que los de la lista y asegurando el paso de parámetros de la información entre funciones.

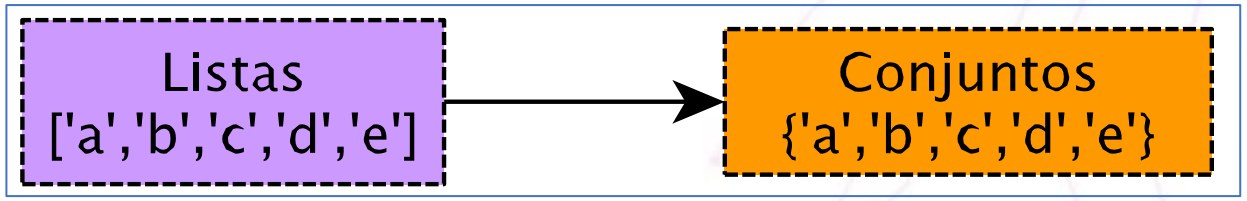



```
1  lista = ['h', 'o', 'l', 'a', 123]
2  tupla = tuple(lista)
3  print(tupla)
```

```
('h', 'o', 'l', 'a', 123)
```

Conversión de Listas

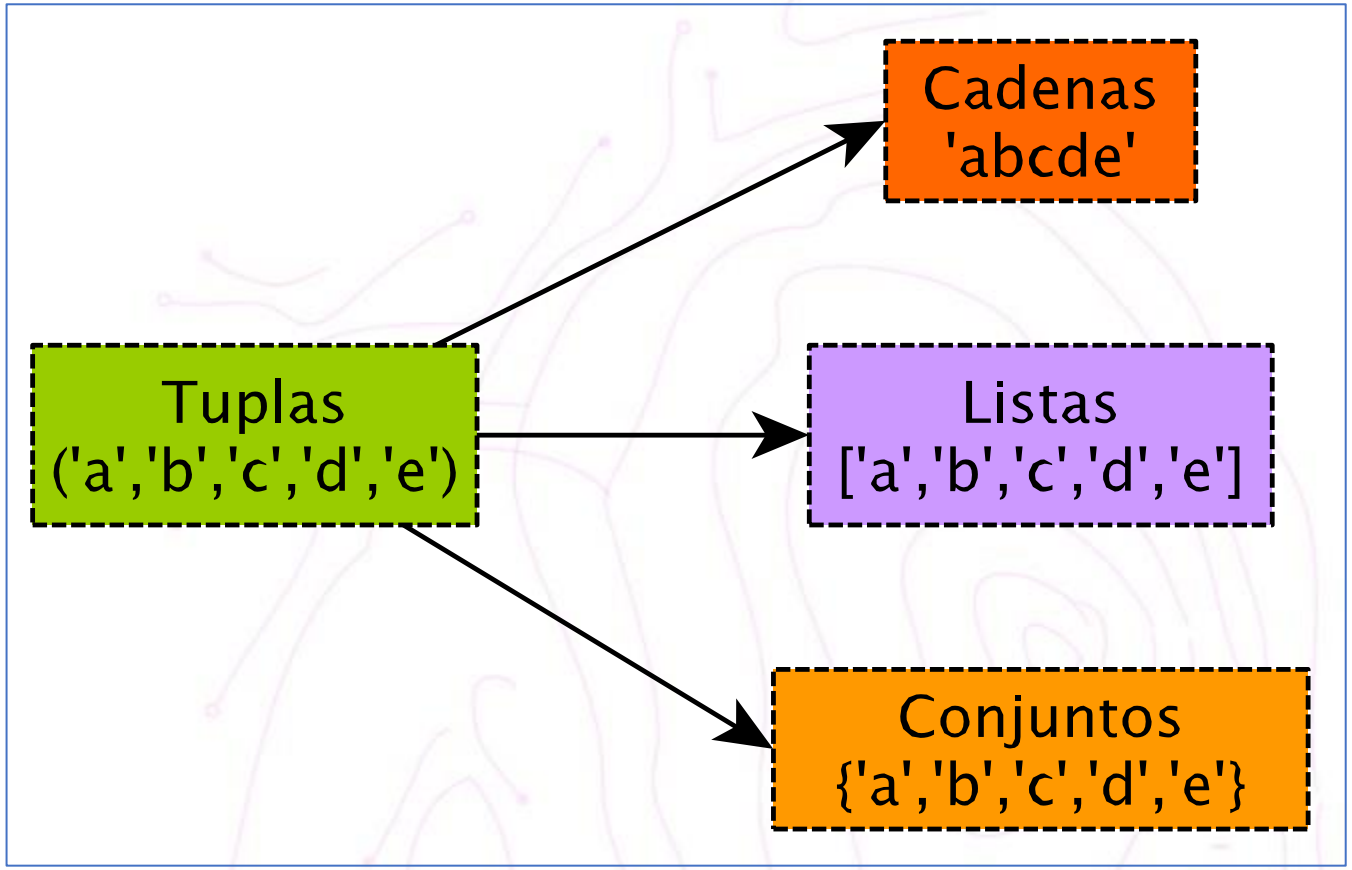
- Esta conversión entre contenedores, tendrá una ganancia inmediata eliminando repeticiones. Adicionalmente, los tipos de datos de los elementos pueden diferir, y esto no es obstáculo para una conversión exitosa. Como se observa, el orden no se conserva en los conjuntos.




```
1  lista = ['h','o','o','l','a',1,1,2]
2  conjunto = set(lista)
3  print(conjunto)
```

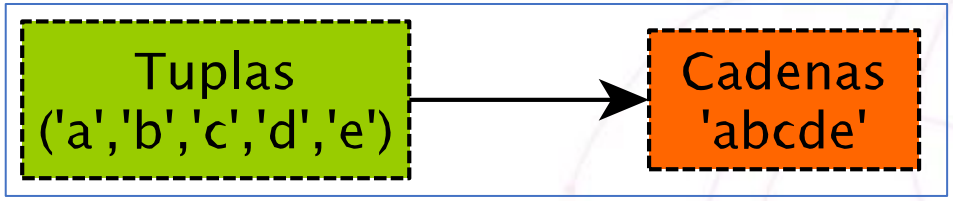
```
{1, 2, 'h', 'a', 'l', 'o'}
```

Conversión de Tuplas



Conversión de Tuplas

- Ambos contenedores son inmutables, sin embargo, esta opción nos permite pasar de una tupla de caracteres a una cadena, teniendo esta última una gama más amplia de métodos y funcionalidades.

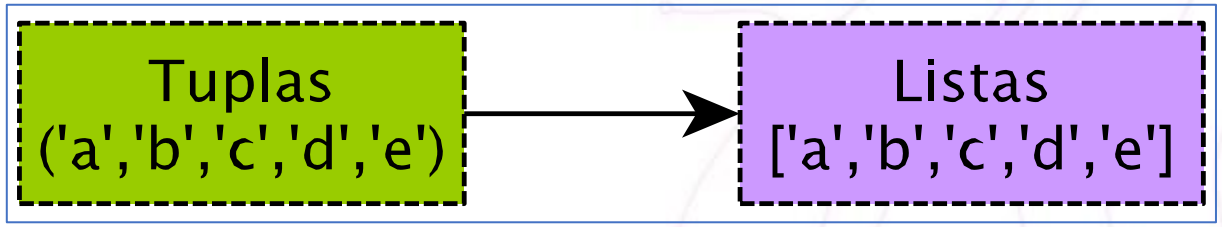



```
1  tupla = ('h', 'o', 'l', 'a')
2  cadena = ''.join(tupla)
3  print(cadena)
```

```
hola
```

Conversión de Tuplas

- Esta conversión es muy útil cuando recibimos parámetros tipo tupla, y necesitamos una copia mutable (modificable) de la tupla para realizar algún cómputo.

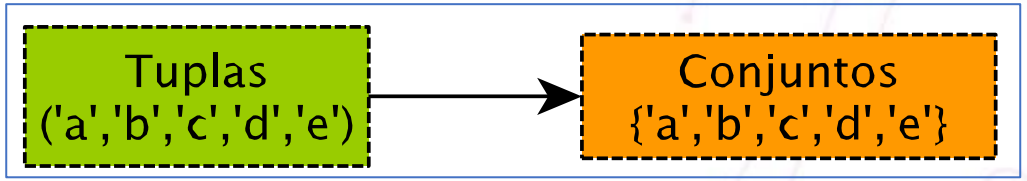



```
1   tupla = ('hola',111,'mundo')
2   lista = list(tupla)
3   print(lista)
```

```
['hola', 111, 'mundo']
```

Conversión de Tuplas

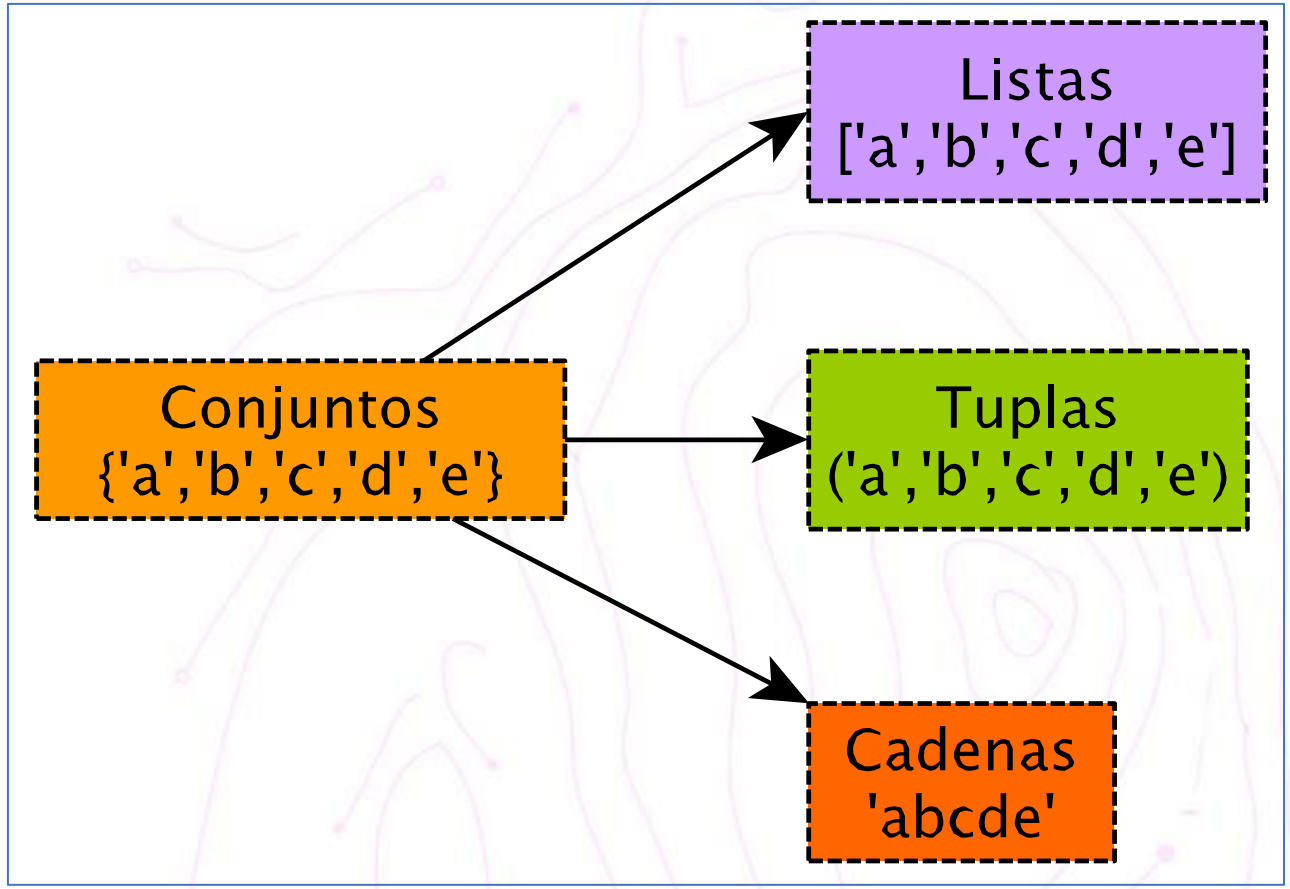
- Los elementos de los conjuntos son inmutables, pero no el incremento o decremento de los elementos que agrupan. Si queremos eliminar repeticiones, y/o quitar la restricción de crecimiento de elementos de una tupla, sin que el orden de los elementos sea relevante, protegiendo el contenido de cada uno, esta conversión puede ser muy útil.




```
1  tupla = ('hola',111,'mundo','hola')
2  conjunto = set(tupla)
3  conjunto.add('15')
4  print(conjunto)
```

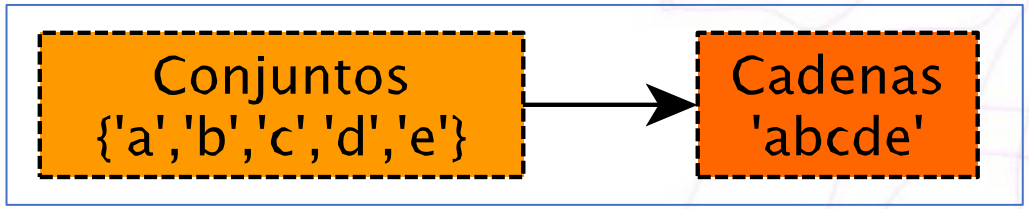
```
{'15', 'hola', 'mundo', 111}
```

Conversión de Conjuntos



Conversión de Conjuntos

- Python permite la posibilidad de convertir un conjunto (colección no ordenada) de elementos en un flujo de caracteres, de manera análoga como se ha presentado previamente.

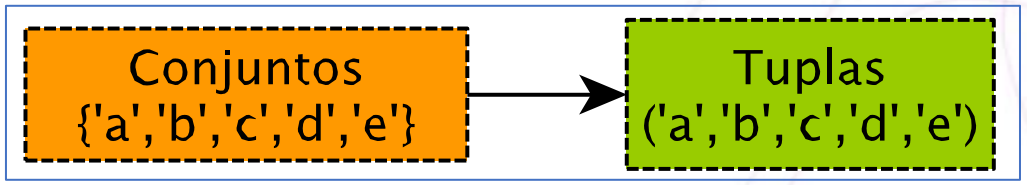



```
1   conjunto = {'h','o','l','a'}
2   cadena = ''.join(conjunto)
3   print(cadena)
```

```
ahlo
```

Conversión de Conjuntos

- Ambos contenedores, siendo restrictivos por ser inmutables, al tener formas diferentes de acceso, podría requerirse acceder de manera secuencial a los elementos, por tal razón, la conversión de conjuntos a tuplas podría ser útil en algunos contextos. Al provenir de conjuntos, el orden sólo empezará a conservarse en la tupla generada de la conversión.

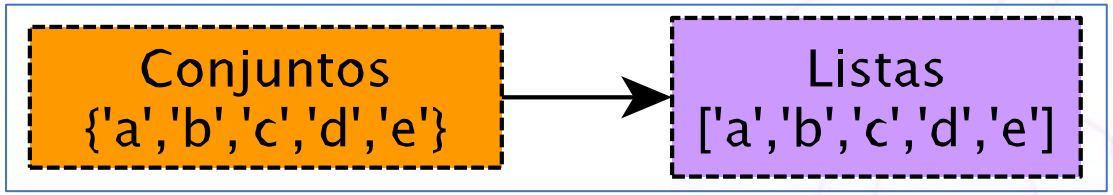



```
1   conjunto = {'h','o','l','a'}
2   tupla = tuple(conjunto)
3   print(tupla)
```

```
('l', 'h', 'o', 'a')
```

Conversión de Conjuntos

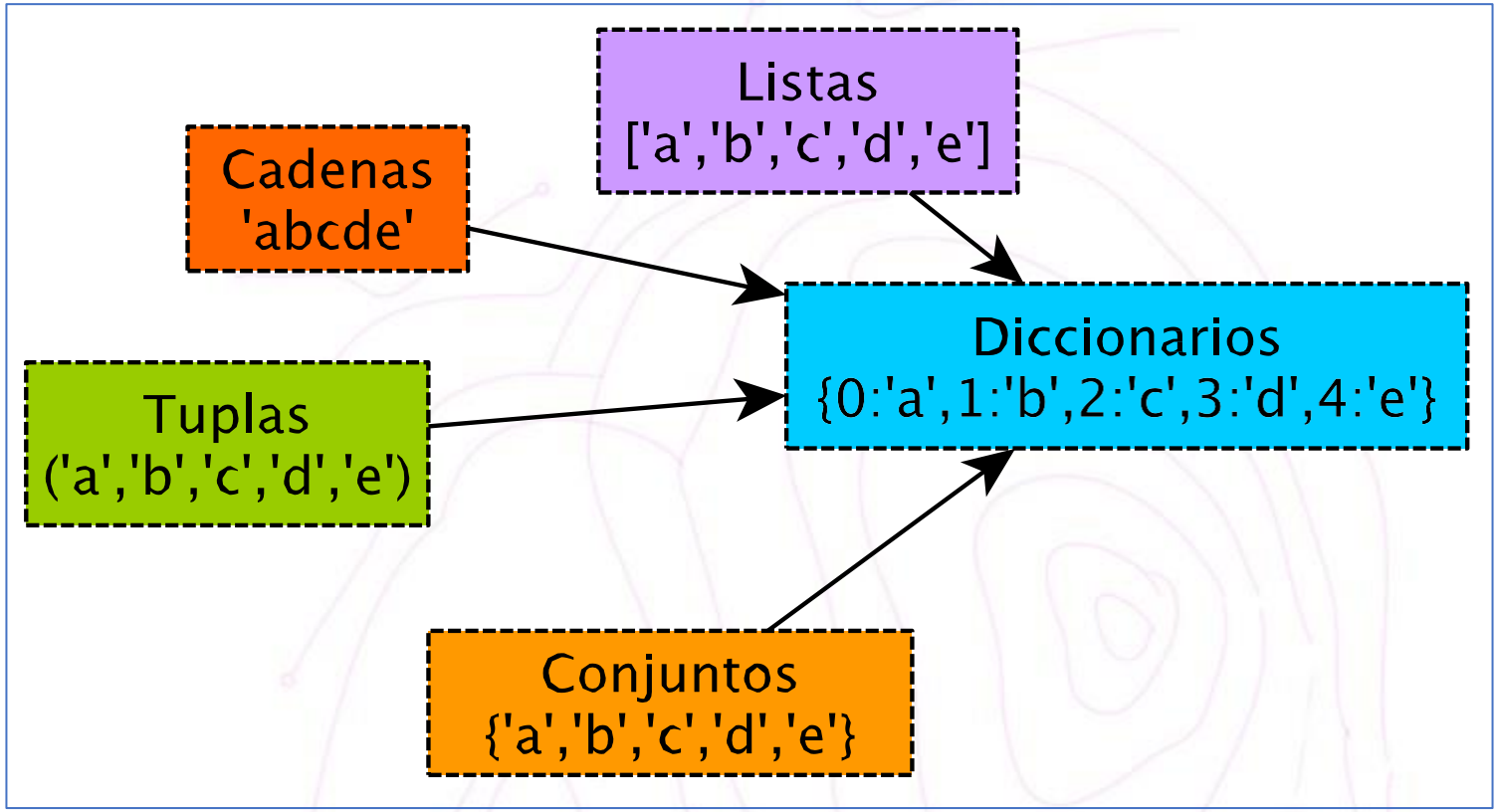
- Cuando necesitamos que los elementos de un conjunto puedan ser accedidos a través de índices secuenciales, conservar el orden de los elementos, y además necesitamos mutabilidad del contenedor que aloja los elementos, la conversión de conjuntos a listas responde a estos requerimientos.




```
1   conjunto = {'h','o','l','a'}
2   lista = list(conjunto)
3   print(lista)
```

```
['h', 'a', 'o', 'l']
```

Conversión a Diccionarios



Conversión a Diccionarios

- Los diccionarios mezclan características de los conjuntos al no permitir repeticiones de sus etiquetas, conservan el orden como los contenedores diferentes a los conjuntos, pero su conversión requiere resolver la ***llave*** o etiqueta que tiene cada uno de sus elementos.

- A diferencia de las tuplas, cadenas y listas, cuyos índices o etiquetas son autonuméricos, el diccionario necesita su especificación.

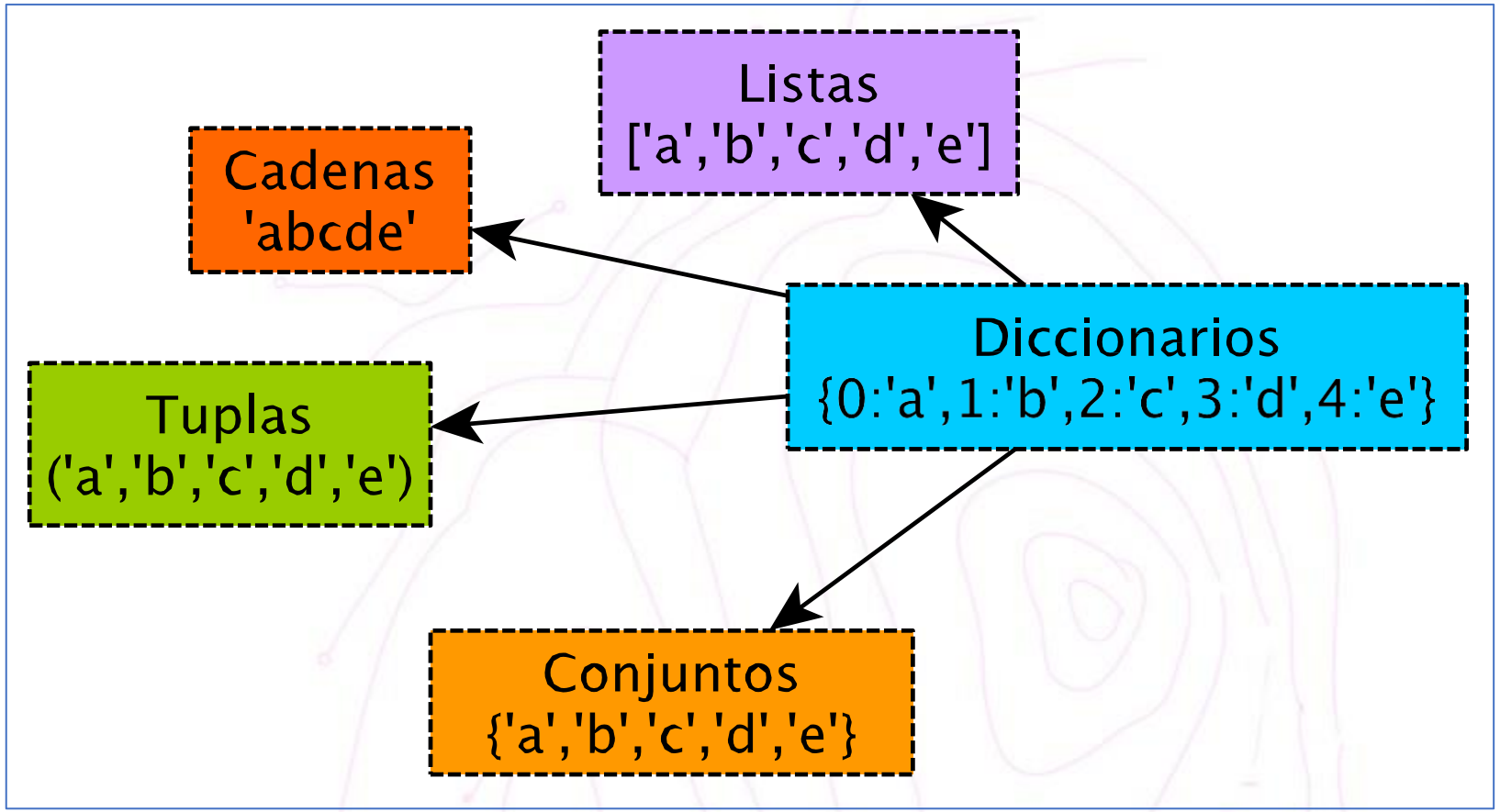
- En los siguientes ejemplos se presenta el uso de la función ***zip*** para relacionar una colección generada automáticamente con los elementos y satisfacer la llave. Esta función se presentará con más detalle en el transcurso de la unidad.

Conversión a Diccionarios

```python
cadena = "hola"
diccionario = dict(zip(range(len(cadena)),cadena))
print(diccionario)

lista = ['h','o','l','a']
diccionario = dict(zip(range(len(lista)),lista))
print(diccionario)

tupla = ('h','o','l','a')
diccionario = dict(zip(range(len(tupla)),tupla))
print(diccionario)

conjunto = {'h','o','l','a'}
diccionario = dict(zip(range(len(conjunto)),conjunto))
print(diccionario)
```

```
{0: 'h', 1: 'o', 2: 'l', 3: 'a'}
{0: 'h', 1: 'o', 2: 'l', 3: 'a'}
{0: 'h', 1: 'o', 2: 'l', 3: 'a'}
{0: 'l', 1: 'a', 2: 'h', 3: 'o'}
```

Conversión desde Diccionarios



Conversión desde Diccionarios

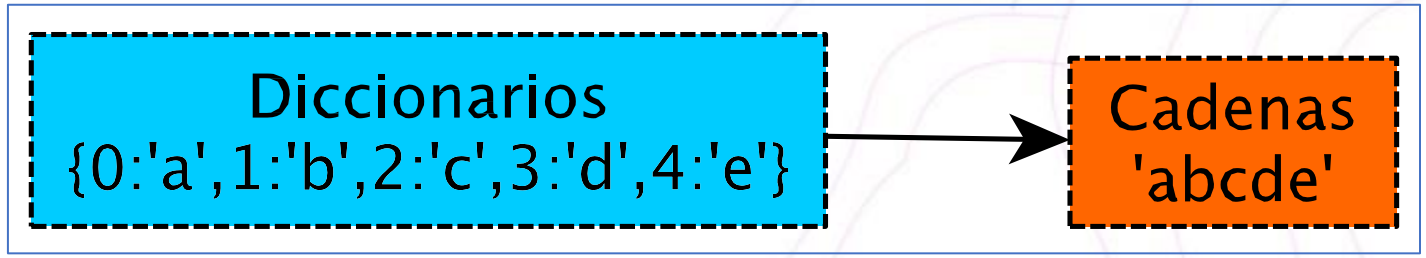
- En el proceso contrario, dado que los diccionarios constituyen dos colecciones, una correspondiente a las ***llaves*** y otra correspondiente a los ***valores***, se debe especificar cuál colección se requiere, o generalizar obteniendo todos los ítems, que son retornados como tuplas.

- En los siguientes ejemplos se presenta cómo la información se puede extraer y convertir en cualquiera de los contenedores.

- En algunos casos sólo se convierten los valores por la compatibilidad de tipos que requerirían funciones adicionales.

Conversión desde Diccionarios

- Podemos convertir los elementos de un diccionario a una cadena, siempre y cuando el tipo de datos coincida. Es posible realizar la conversión tanto de las llaves, como de los valores del diccionario. En el ejemplo, el tipo apto para esta conversión son los valores (***values***) del diccionario.

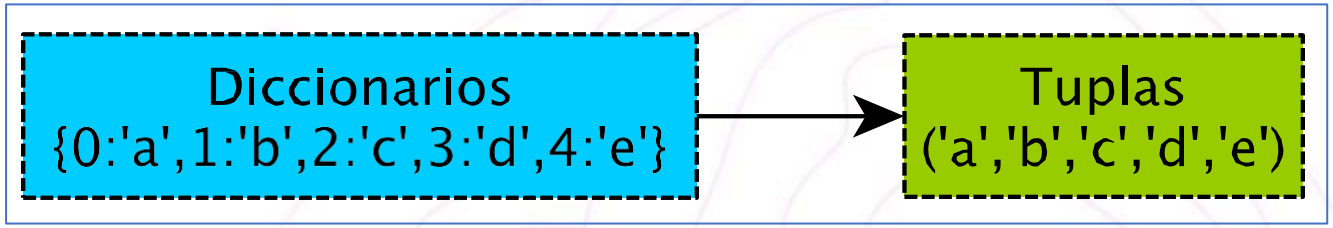



```
1  diccionario = {0: 'h', 1: 'o', 2: 'l', 3: 'a'}
2  cadenaValores = "".join(diccionario.values())
3  print(cadenaValores)
```

```
hola
```

Conversión desde Diccionarios

- Llaves, valores e ítems (la combinación de los anteriores del diccionario) pueden ser almacenados en tuplas, estructura que protege los contenidos ante modificación, y ofrecen tiempos de respuesta menores.

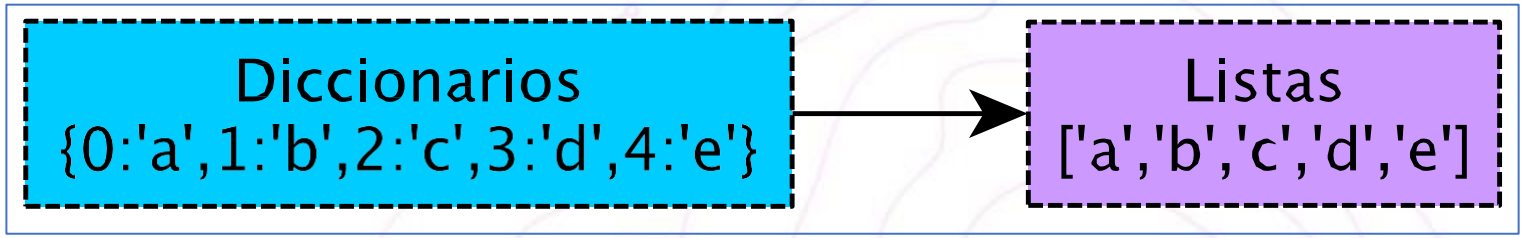



```
1  diccionario = {0: 'h', 1: 'o', 2: 'l', 3: 'a'}
2  tuplaLlaves = tuple(diccionario.keys())
3  tuplaValores = tuple(diccionario.values())
4  tuplaItems = tuple(diccionario.items())
5  print(tuplaLlaves)
6  print(tuplaValores)
7  print(tuplaItems)
```

```
(0, 1, 2, 3)
('h', 'o', 'l', 'a')
((0, 'h'), (1, 'o'), (2, 'l'), (3, 'a'))
```

Conversión desde Diccionarios

- Llaves, valores e ítems (la combinación de los anteriores del diccionario) pueden ser almacenados en listas, estructura mutable para una manipulación flexible de dicha información.

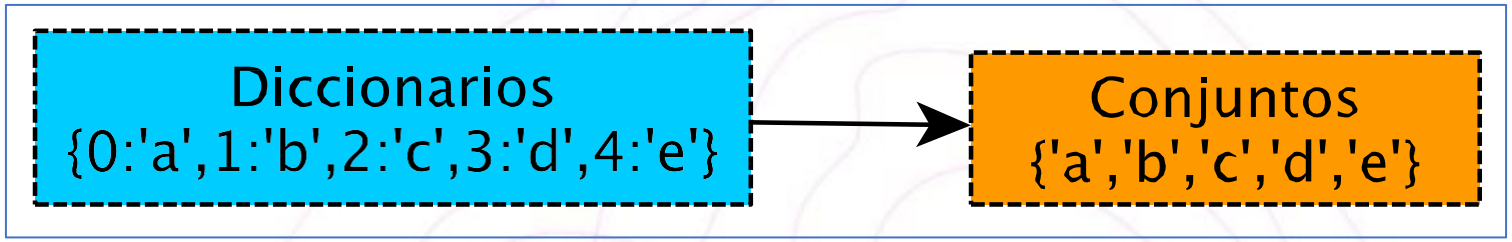



```
1  diccionario = {0: 'h', 1: 'o', 2: 'l', 3: 'a'}
2  listaLlaves = list(diccionario.keys())
3  listaValores = list(diccionario.values())
4  listaItems = list(diccionario.items())
5  print(listaLlaves)
6  print(listaValores)
7  print(listaItems)
```

```
[0, 1, 2, 3]
['h', 'o', 'l', 'a']
[(0, 'h'), (1, 'o'), (2, 'l'), (3, 'a')]
```

Conversión desde Diccionarios

- Llaves, valores e ítems (la combinación de los anteriores del diccionario) pueden ser almacenados en conjuntos, permitiendo la eliminación de repeticiones de los valores del diccionario.




```
1  diccionario = {0: 'h', 1: 'o', 2: 'l', 3: 'a', 4:'a'}
2  conjuntoLlaves = set(diccionario.keys())
3  conjuntoValores = set(diccionario.values())
4  conjuntoItems = set(diccionario.items())
5  print(conjuntoLlaves)
6  print(conjuntoValores)
7  print(conjuntoItems)
```

```
{0, 1, 2, 3, 4}
{'a', 'o', 'h', 'l'}
{(4, 'a'), (3, 'a'), (0, 'h'), (2, 'l'), (1, 'o')}
```