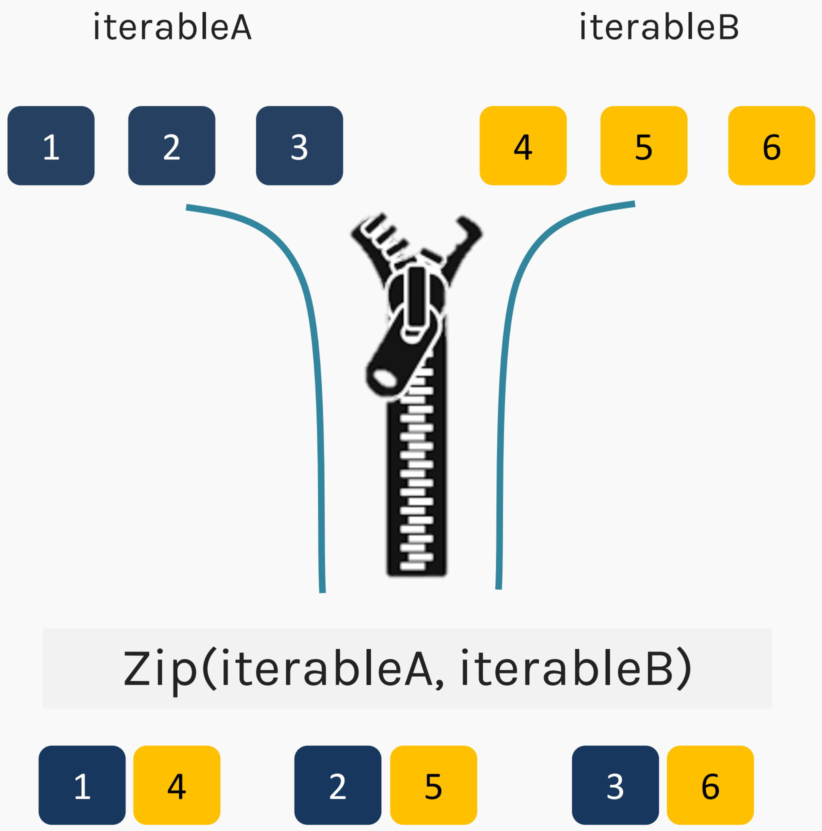
Zip & Enumerate

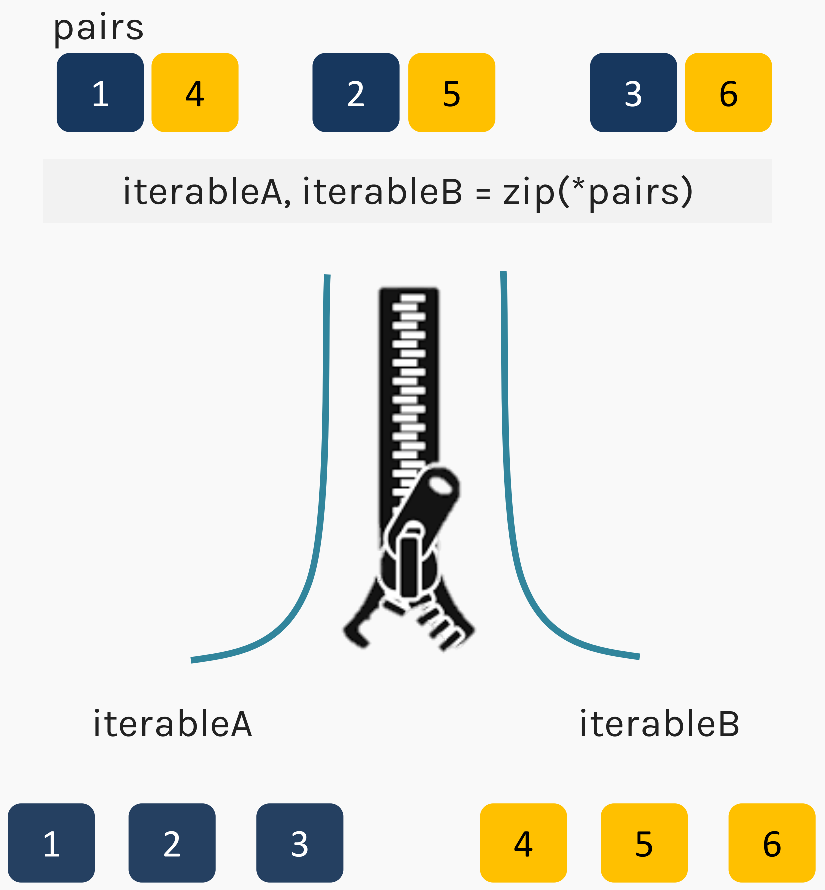
Zip



Python's zip() function creates an iterator that will aggregate elements from two or more iterables.

```
>>> letters = ['a','b','c','d']
>>> numbers = [1,2,3,4]

>>> for letter, number in
zip(letters, numbers):
...       print(letter,number)
'a' 1
'b' 2
'c' 3
'd' 4
```

Unpacking operator (splat)*



We've seen how to zip something. But how do we unzip something? This is where the * operator comes in.

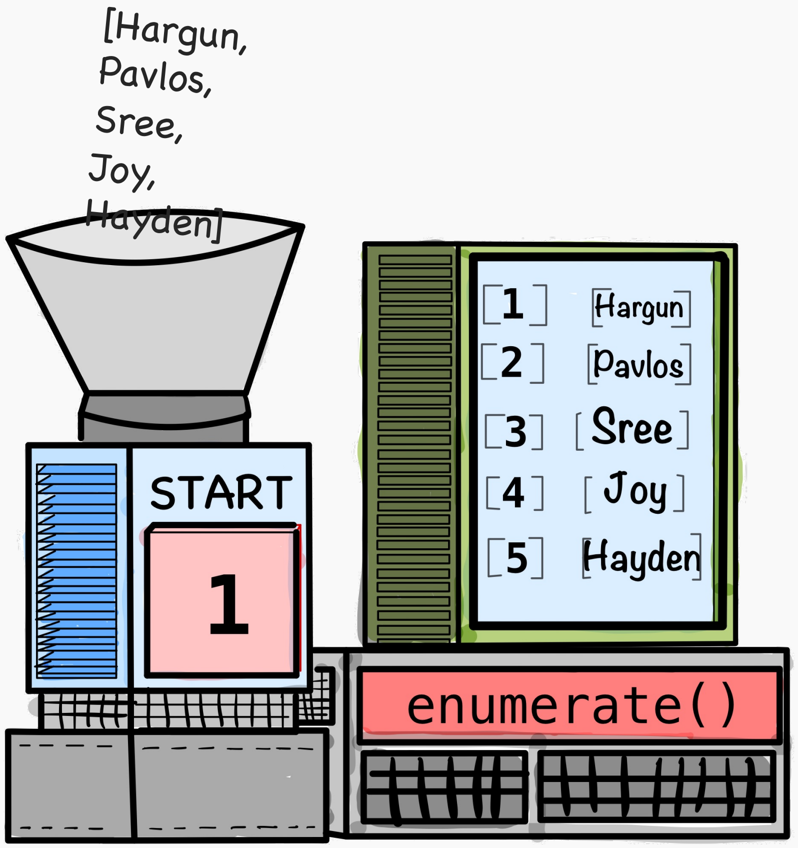
```
>>> pairs =
[(1,’a’),(2,’b’),(3,’c’)]
>>> numbers, letters =
zip(*pairs)

>>> print(numbers)
(1,2,3)

>>> print(letters)
(‘a’,’b’,’c’)
```

Enumerate

- When you use enumerate(), the function gives you back *two* loop variables:
  - The **count** of the current iteration
  - The **value** of the item at the current iteration
- The use of two loop variables i.e count and value, is an example of argument unpacking.

```
SYNTAX: for count,value in enumerate(values):
   ...:      Do something
```

- There are many times when you might not want to count from index 0. In that case, you can use the start argument to change the starting count index.


[Hargun,
Pavlos,
Sree,
Joy,
Hayden]
START
1
[1] [Hargun]
[2] [Pavlos]
[3] [Sree]
[4] [Joy]
[5] [Hayden]
enumerate()