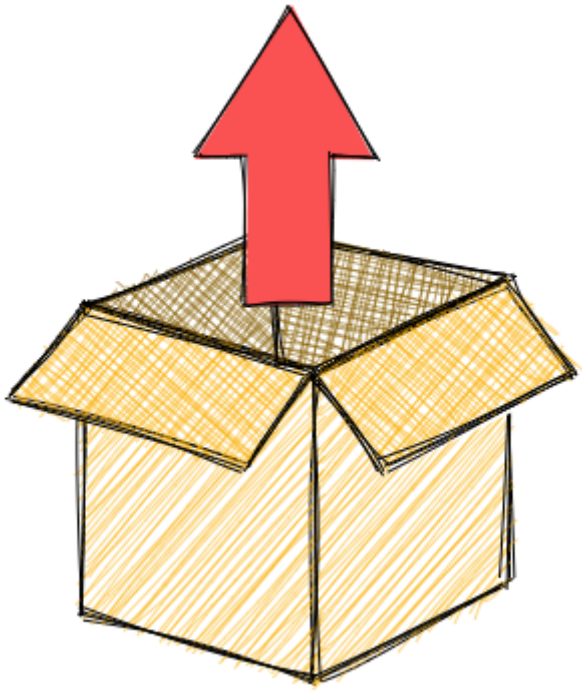
Efficient Python Tricks and Tools for Data Scientists - By Khuyen Tran

Unpack Iterables



Khuyen Tran

@khuyentran1401

How to Unpack Iterables in Python

To assign items of a Python iterables (such as list, tuple, string) to different variables, you can unpack the iterable like below.

```
nested_arr = [[1, 2, 3], ["a", "b"], 4]
num_arr, char_arr, num = nested_arr
```

```
num_arr
```

```
[1, 2, 3]
```

```
char_arr
```

```
['a', 'b']
```

Khuyen Tran

@khuyentran1401

Extended Iterable Unpacking: Ignore Multiple Values when Unpacking a Python Iterable

If you want to ignore multiple values when unpacking a Python iterable, add * to _ as shown below.

This is called “Extended Iterable Unpacking” and is available in Python 3.x.

```
a, *_, b = [1, 2, 3, 4]
print(a)
```

```
1
```

```
b
```

```
4
```

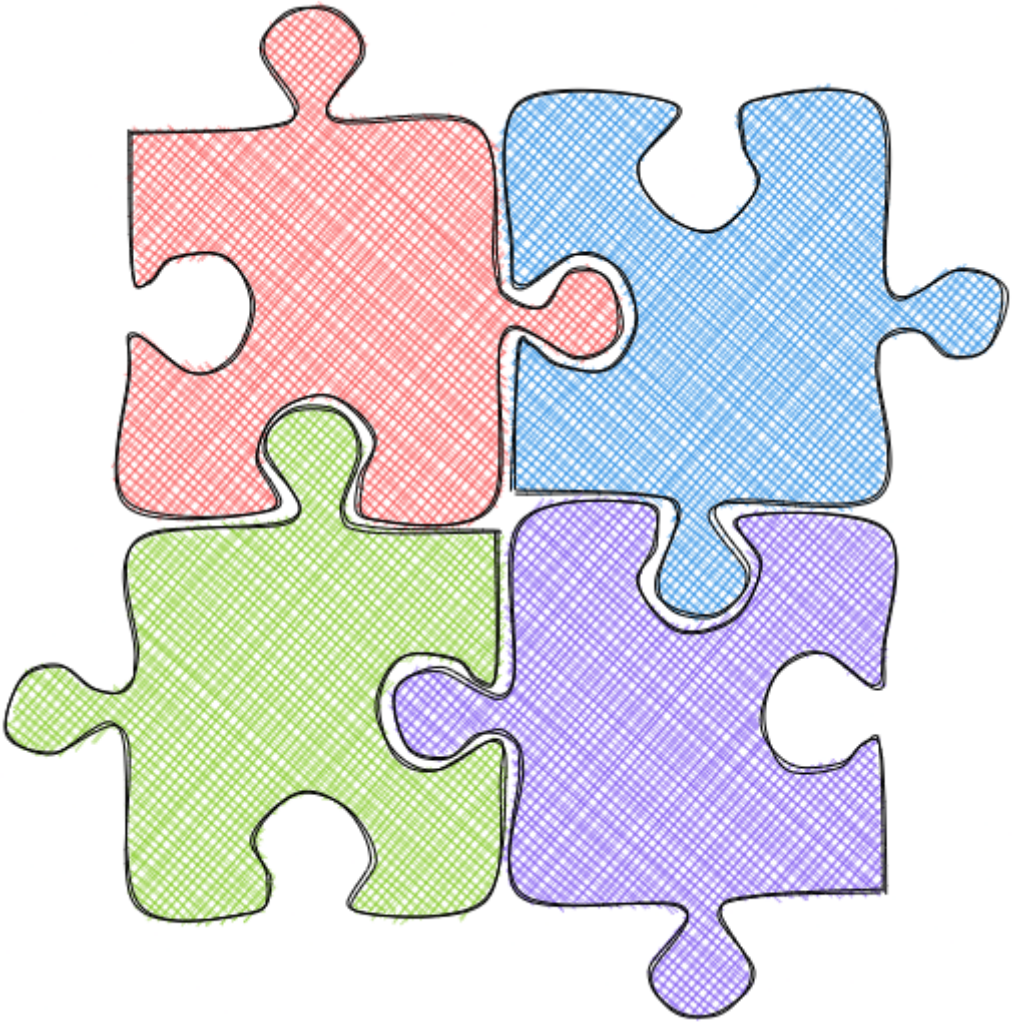
```
_
```

```
[2, 3]
```

Khuyen Tran

@khuyentran1401

Join Iterables



Khuyen Tran

@khuyentran1401

join method: Turn an Iterable into a Python String

If you want to turn an iterable into a string, use `join()`.

In the code below, I join elements in the list `fruits` using “, “.

```
fruits = ['apples', 'oranges', 'grapes']

fruits_str = ', '.join(fruits)

print(f"Today, I need to get some {fruits_str} in the grocery store")
```

```
Today, I need to get some apples, oranges, grapes in the grocery store
```

Khuyen Tran

@khuyentran1401

Zip: Associate Elements from Two Iterators based on the Order

If you want to associate elements from two iterators based on the order, combine `list` and `zip`.

```
nums = [1, 2, 3, 4]
string = "abcd"
combinations = list(zip(nums, string))
combinations
```

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```

Khuyen Tran

@khuyentran1401

Unzip a List of Iterables

You can turn a list of iterables into 2 separate iterables using `zip(*list)`.

```
comb = [(1, 'a'), (2, 'b'), (3, 'c'), (4,
'd')]
```

```
nums_2, chars_2 = zip(*comb)
nums_2
```

```
(1, 2, 3, 4)
```

```
chars_2
```

```
('a', 'b', 'c', 'd')
```

Khuyen Tran

@khuyentran1401