Course
CS-E4002
Course materials
Your points
Code Vault

This course has already ended.

« Functions and Sparse Data Course materials Sparse data with Scipy »

CS-E4002 / Functions and Sparse Data / Iterable manipulation

Iterable manipulation

Definition: An iterable is an object capable of returning its members one at a time.

Examples: lists, sets and dictionaries are commonly used iterables.

Definition: An iterator is an object that "points" on an iterable and allows to go back and forth in it, going to the next or previous element.

1. Comprehension iterable

To initialize an iterable in Python, it can be convenient to use what is called a comprehension. It is a way to build an iterable from another iterable, using a for loop.

Here is a first comprehension list:

```
>>> [x for x in range(0, 10)]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

If statements can also be used in these comprehension lists:

```
>>> our_list = [x for x in range(0, 10) if x%2 == 1]
>>> print(our_list)
[1, 3, 5, 7, 9]
```

We can write similar statements for sets and for dictionaries:

```
>>> {x - 1 for x in our_list}
{0, 2, 4, 6, 8}

>>> {x:y for (x,y) in enumerate(our_list)}
{0: 1, 1: 3, 2: 5, 3: 7, 4: 9}
```

2. Map function

The map() function is one of Python's built-in functions. It can be used to apply a function to each element of an iterable. It returns an iterator, that can be converted into the desired iterable.

```
>>> map(lambda a : a**2, our_list)
<map object at 0x7f2134bd7d90>

>>> list(map(lambda a : a**2, our_list))
[1, 9, 25, 49, 81]
```

3. Zip function

The zip() function is also one of Python's built-in functions. It is used to create an iterator of tuples from iterables.

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> list(zip(x, y))
[(1, 4), (2, 5), (3, 6)]

>>> z = ["a", "b", "c"]
>>> list(zip(x, y, z))
[(1, 4, 'a'), (2, 5, 'b'), (3, 6, 'c')]
```

Notice that the iterables have to have the same size (other functions exist for special cases like zip_longest() but they are not built-in). However, the types of the elements can be different between iterables.

« Functions and Sparse Data Course materials Sparse data with Scipy »