Department of Computer Science and Engineering
PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving
UE23CS151A

Lecture #93

The Function – zip

By,
Prof. Sindhu R Pai,
Anchor, PCPS - 2023
Assistant Professor
Dept. of CSE, PESU
&
Dr. Manju More E
Associate Professor
Dept. of CSE, PESU

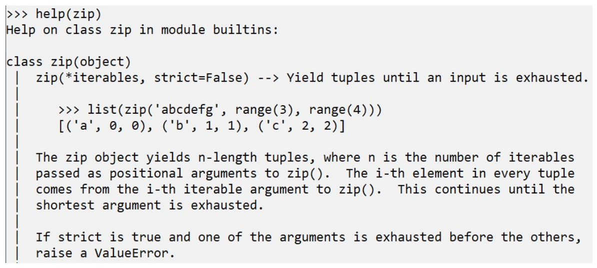
Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Centers, Former Chairperson, CSE, PES University)
Prof. Chitra G M, (Asst. Prof, Dept. of CSE, PCPS Anchor – 2022)

The function – zip()

The zip() function returns a **zip object, which is an iterator of tuples** where the ith tuple will be created by pairing the ith element from each of the iterables. If the passed **iterables have different lengths, the iterable with the least items decides the length of the new iterator.**

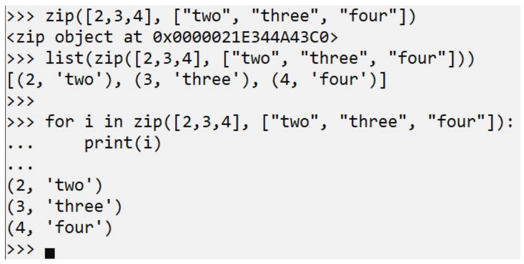
It does not stand for data compression. It is used to associate the corresponding elements of two or more iterables into a single lazy iterable of tuples. **It does not have any callback function.** It is used to map the similar index of multiple containers so that they can be used just using a single entity.

Syntax: zip(iterator1, iterator2, iterator3 ...)



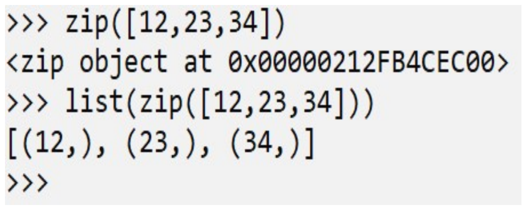
```
>>> help(zip)
Help on class zip in module builtins:

class zip(object)
 |  zip(*iterables, strict=False) --> Yield tuples until an input is exhausted.
 |
 |     >>> list(zip('abcdefg', range(3), range(4)))
 |     [('a', 0, 0), ('b', 1, 1), ('c', 2, 2)]
 |
 |  The zip object yields n-length tuples, where n is the number of iterables
 |  passed as positional arguments to zip().  The i-th element in every tuple
 |  comes from the i-th iterable argument to zip().  This continues until the
 |  shortest argument is exhausted.
 |
 |  If strict is true and one of the arguments is exhausted before the others,
 |  raise a ValueError.
```

You iterate through the series of tuples returned by zip() and unpack the elements.

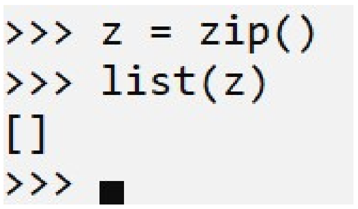


```
>>> zip([2,3,4], ["two", "three", "four"])
<zip object at 0x0000021E344A43C0>
>>> list(zip([2,3,4], ["two", "three", "four"]))
[(2, 'two'), (3, 'three'), (4, 'four')]
>>>
>>> for i in zip([2,3,4], ["two", "three", "four"]):
...     print(i)
...
(2, 'two')
(3, 'three')
(4, 'four')
>>> ■
```

With a single iterable argument, it returns an iterator of 1-tuples. With no arguments, it returns an empty iterator.



```
>>> zip([12,23,34])
<zip object at 0x00000212FB4CEC00>
>>> list(zip([12,23,34]))
[(12,), (23,), (34,)]
>>>
```



```
>>> z = zip()
>>> list(z)
[]
>>> ■
```

Let us discuss few example codes related to zip function.

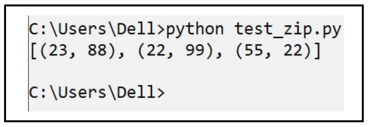
Example_code_1: Create a list of tuples with two lists given

A = [23,22,55, 99]

B = [88,99,22]

Expected output: [(22, 88), (22,99),(55,22)]

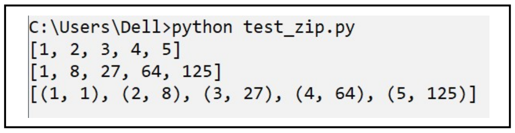
```
A = [23,22,55, 99]
B = [88,99,22]
print(list(zip(A,B)))
```



```
C:\Users\Dell>python test_zip.py
[(23, 88), (22, 99), (55, 22)]

C:\Users\Dell>
```

Example_code_2: Given a list, create a new list by creating the cubes of items from the first list. Combine these two lists to one list containing the tuple where each tuple has 2 items respectively from first and second list.

```
a = [1,2,3,4,5]
b = list(map(lambda x : x * x * x, a))
print(a)
print(b)
print(list(zip(a, b)))
```



```
C:\Users\Dell>python test_zip.py
[1, 2, 3, 4, 5]
[1, 8, 27, 64, 125]
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125)]
```

If we do not convert the map object to list, do we get the same output in the final list?

Example_code_3:

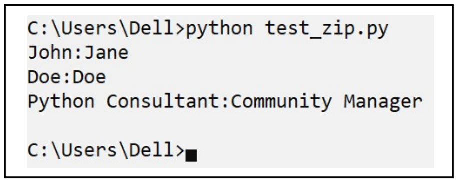
dict_one = {'name': 'John', 'last_name': 'Doe', 'job': 'Python Consultant'}

dict_two = {'name': 'Jane', 'last_name': 'Doe', 'job': 'Community Manager'}

Expected output:

John:Jane
Doe:Doe
Python Consultant:Community Manager

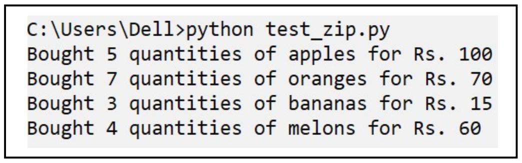
```
dict_one = {'name': 'John', 'last_name': 'Doe', 'job': 'Python Consultant'}
dict_two = {'name': 'Jane', 'last_name': 'Doe', 'job': 'Community Manager'}
z = zip(dict_one.items(), dict_two.items())
for item1, item2 in z:
        print(item1[1]+":"+item2[1])
```



```
C:\Users\Dell>python test_zip.py
John:Jane
Doe:Doe
Python Consultant:Community Manager

C:\Users\Dell>
```

Example_code_4: Given a list of fruits, their prices and the quantities that you purchased in three different lists, program to find the total amount spent on each item.

```
fruits = ["apples","oranges","bananas","melons"]
prices = [20,10,5,15]
quantities = [5,7,3,4]
for fruit,price, q in zip(fruits, prices, quantities):
   print("Bought",q,"quantities of", fruit,"for Rs.",price*q)
```



```
C:\Users\Dell>python test_zip.py
Bought 5 quantities of apples for Rs. 100
Bought 7 quantities of oranges for Rs. 70
Bought 3 quantities of bananas for Rs. 15
Bought 4 quantities of melons for Rs. 60
```

Think about this!

- **list(zip(range(5), range(100))) returns what?**
- **If we want the longest iterable to be considered, do we have any function?**

  Yes – zip_longest from itertools module.

  ```
  >>> import itertools
  >>> list(itertools.zip_longest([1,2], [11,22,33,44,55]))
  [(1, 11), (2, 22), (None, 33), (None, 44), (None, 55)]
  ```

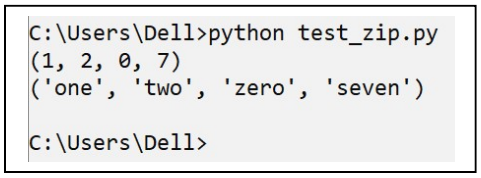
- **Can we use next on zip object or zip_longest object?**
- **Can we unzip the zipped object?**

Unzipping the zipped object using *

Unzipping means converting the zipped values back to the individual iterables as they were. **This is done with the help of "*" operator.**

Let us consider an example code to get the two tuples from the given list of tuples.

```
li = [(1,"one"), (2, "two"), (0, "zero"), (7,"seven")]
numbers, words = zip(*li)
print(numbers, words, sep = "\n")
```



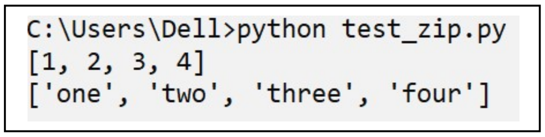
```
C:\Users\Dell>python test_zip.py
(1, 2, 0, 7)
('one', 'two', 'zero', 'seven')

C:\Users\Dell>
```

Points to think!

- **If you pass *li to print(), i.e., print(*li), what gets printed? Think!!!**
- **Can you pass the zip object to zip function with * for unzipping? - Yes**

Example_code_5: Create the original lists by unzipping on the zipped object.

```
la = [1,2,3,4]
lb = ["one", "two", "three", "four"]
z = zip(la, lb)
orig_la, orig_lb = zip(*z)
print(list(orig_la))
print(list(orig_lb))
```

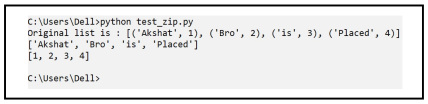


```
C:\Users\Dell>python test_zip.py
[1, 2, 3, 4]
['one', 'two', 'three', 'four']
```

Can we use list comprehension to unzip a list of tuples? Yes

Example_code_6: Create two lists – words and numbers from the given list of tuples

```
test_list = [('Akshat', 1), ('Bro', 2), ('is', 3), ('Placed', 4)]
print("Original list is : " + str(test_list))
words,numbers = [[i for i, j in test_list],[j for i, j in test_list]]
print(words, numbers, sep = "\n")
```



```
C:\Users\Dell>python test_zip.py
Original list is : [('Akshat', 1), ('Bro', 2), ('is', 3), ('Placed', 4)]
['Akshat', 'Bro', 'is', 'Placed']
[1, 2, 3, 4]

C:\Users\Dell>
```

-END-