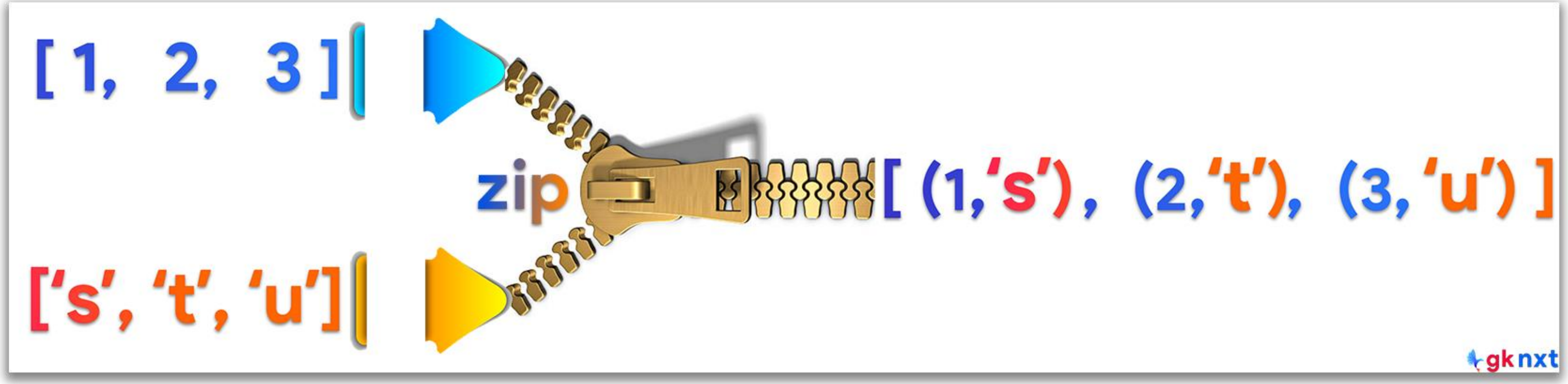
Python Bootcamp & Masterclass

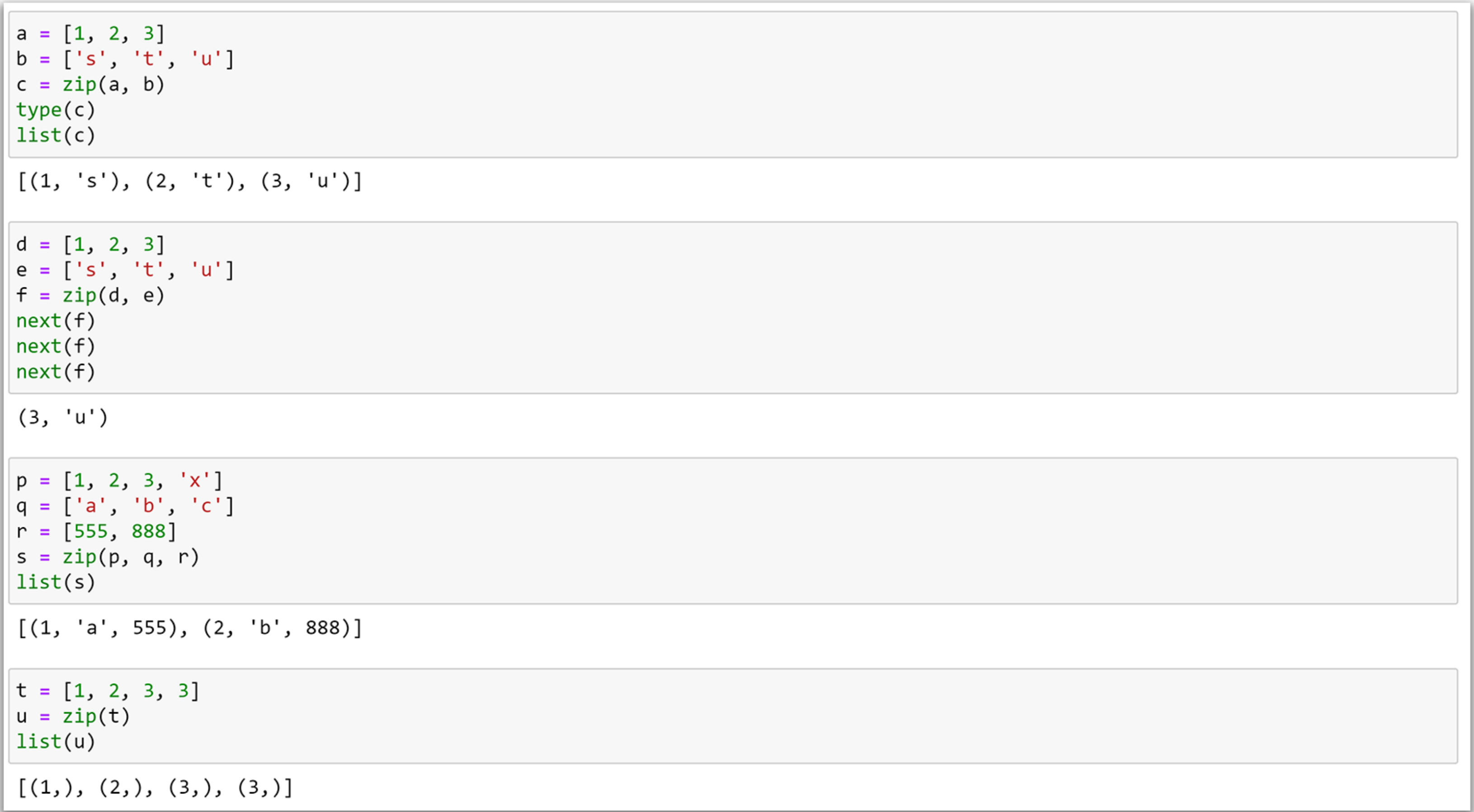
zip

gknxt

The zip(*iterables, strict=False) function returns an iterator of tuples, where the ith tuple contains the ith element from each of the iterables. By default, zipping stops when the shortest input iterable is exhausted. With a single iterable argument, it returns an iterator of 1-tuples. With no arguments, it returns an empty iterator.



gknxt



```python
a = [1, 2, 3]
b = ['s', 't', 'u']
c = zip(a, b)
type(c)
list(c)
```

```
[(1, 's'), (2, 't'), (3, 'u')]
```

```python
d = [1, 2, 3]
e = ['s', 't', 'u']
f = zip(d, e)
next(f)
next(f)
next(f)
```

```
(3, 'u')
```

```python
p = [1, 2, 3, 'x']
q = ['a', 'b', 'c']
r = [555, 888]
s = zip(p, q, r)
list(s)
```

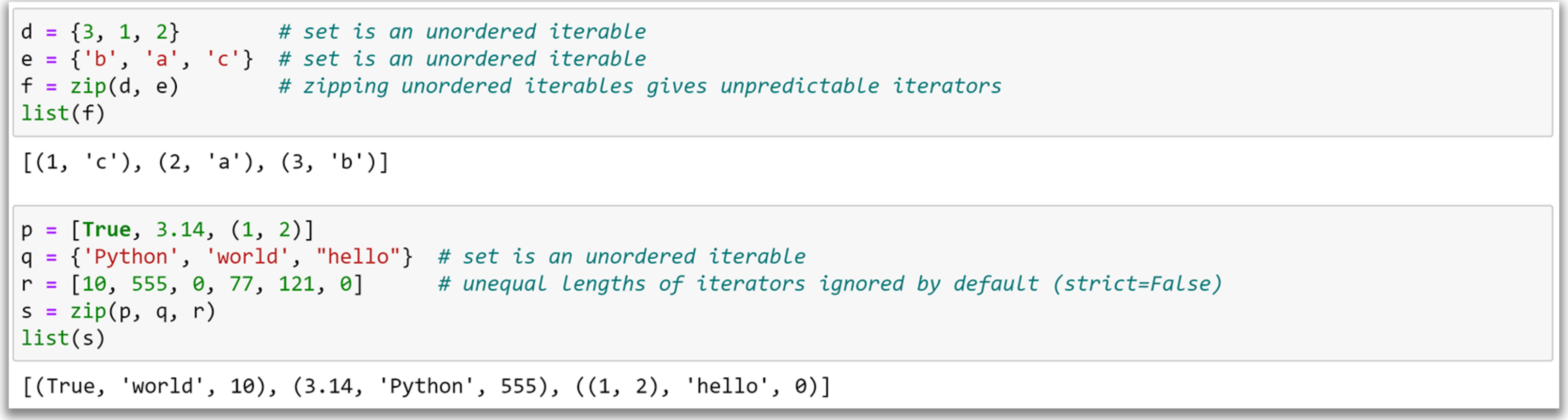
```
[(1, 'a', 555), (2, 'b', 888)]
```

```python
t = [1, 2, 3, 3]
u = zip(t)
list(u)
```

```
[(1,), (2,), (3,), (3,)]
```

gknxt

When zipping sequences (lists, tuples, or strings) the iterables are guaranteed to be evaluated from left to right. This is not the case with unordered container objects (sets)



```
d = {3, 1, 2}          # set is an unordered iterable
e = {'b', 'a', 'c'}    # set is an unordered iterable
f = zip(d, e)          # zipping unordered iterables gives unpredictable iterators
list(f)
```

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

```
p = [True, 3.14, (1, 2)]
q = {'Python', 'world', "hello"}  # set is an unordered iterable
r = [10, 555, 0, 77, 121, 0]      # unequal lengths of iterators ignored by default (strict=False)
s = zip(p, q, r)
list(s)
```

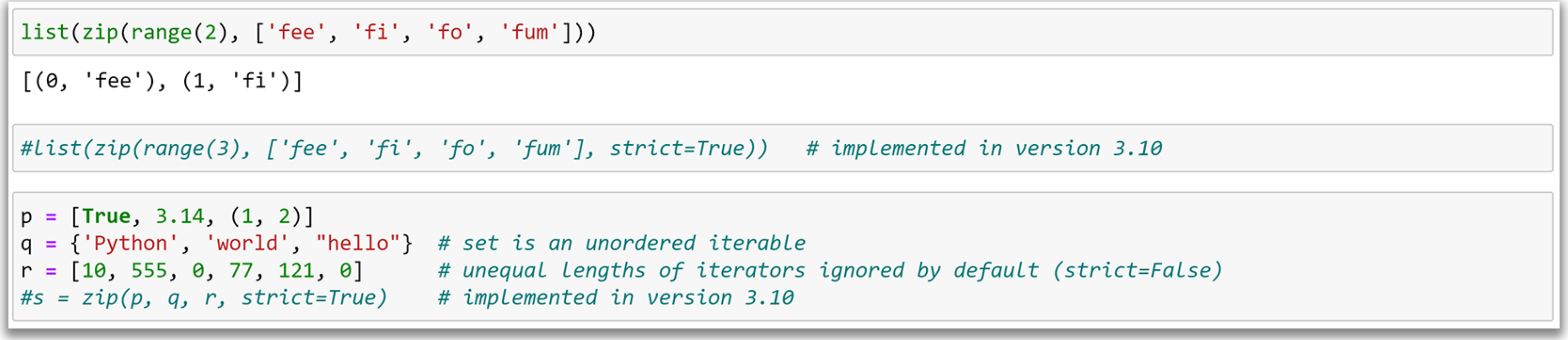
```
[(True, 'world', 10), (3.14, 'Python', 555), ((1, 2), 'hello', 0)]
```

strict

By default, `zip()` stops when the shortest iterable is exhausted. It will ignore the remaining items in the longer iterables, cutting off the result to the length of the shortest iterable. Since Python 3.10, zip() has a new optional keyword argument called `strict` to provide a safe way to handle iterables of unequal length.

zip() is often used in cases where the iterables are assumed to be of equal length. In such cases, it’s recommended to use the `strict=True` option. Without the strict=True argument, any bug that results in iterables of different lengths will be silenced, possibly manifesting as a hard-to-find bug in another part of the program.

gknxt



```
list(zip(range(2), ['fee', 'fi', 'fo', 'fum']))
```

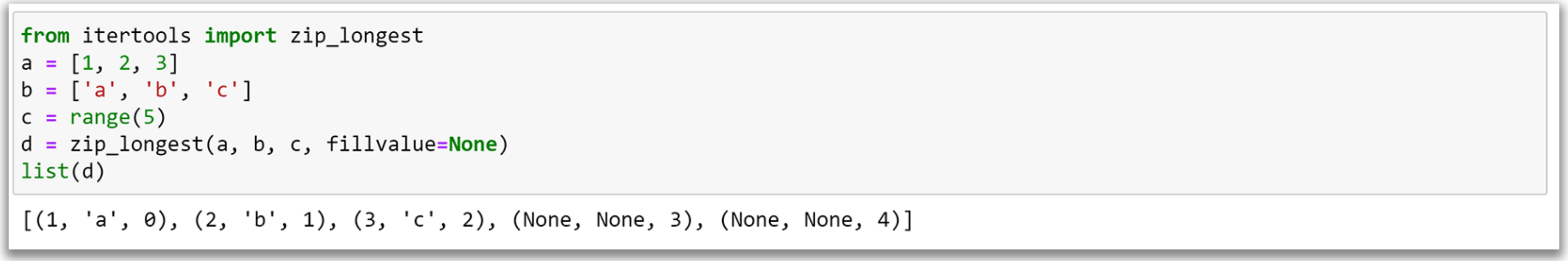
```
[(0, 'fee'), (1, 'fi')]
```

```
#list(zip(range(3), ['fee', 'fi', 'fo', 'fum'], strict=True))   # implemented in version 3.10
```

```
p = [True, 3.14, (1, 2)]
q = {'Python', 'world', "hello"}  # set is an unordered iterable
r = [10, 555, 0, 77, 121, 0]      # unequal lengths of iterators ignored by default (strict=False)
#s = zip(p, q, r, strict=True)    # implemented in version 3.10
```

zip longest

Shorter iterables can be padded with a constant value to make all the iterables have the same length. This is done by `itertools.zip_longest()`

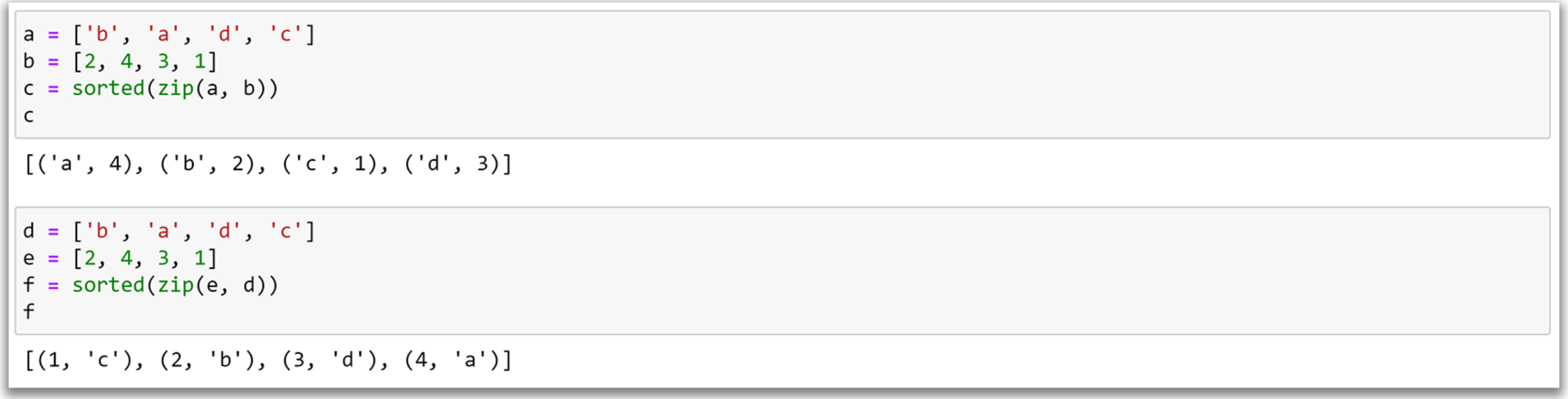


```python
from itertools import zip_longest
a = [1, 2, 3]
b = ['a', 'b', 'c']
c = range(5)
d = zip_longest(a, b, c, fillvalue=None)
list(d)
```

```
[(1, 'a', 0), (2, 'b', 1), (3, 'c', 2), (None, None, 3), (None, None, 4)]
```

sorted( )

The `sorted( )` function can be used to sort iterables.



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

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

```
d = ['b', 'a', 'd', 'c']
e = [2, 4, 3, 1]
f = sorted(zip(e, d))
f
```

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

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python
Bootcamp
& Masterclass

Thank You
for your Rating & Review

gknxt