9

Python Quick Tips

Zip Iteration

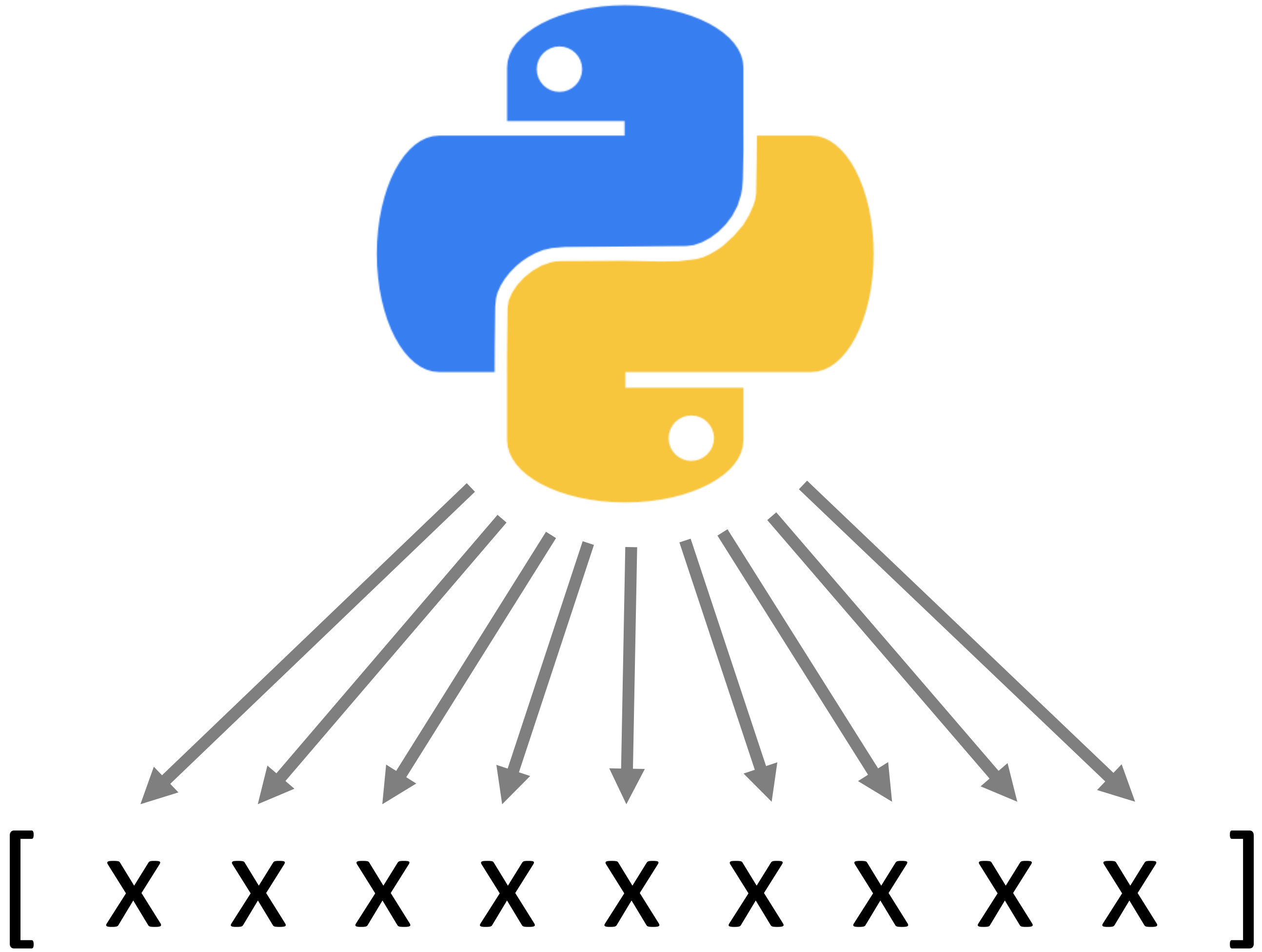
Python Quick Tips #9

Zip Iteration

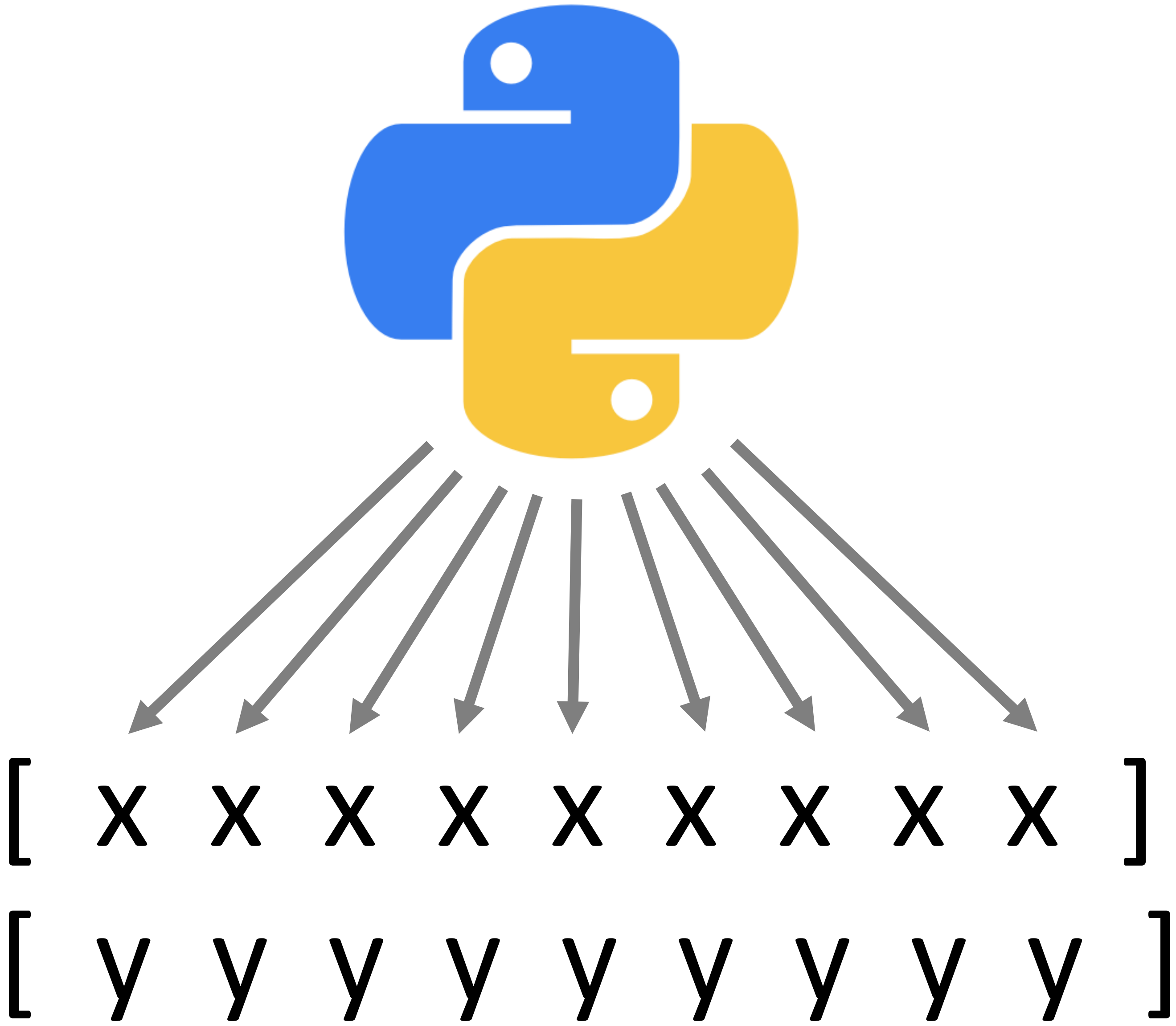
Iteration

The **repetition** of a process to generate an **outcome**

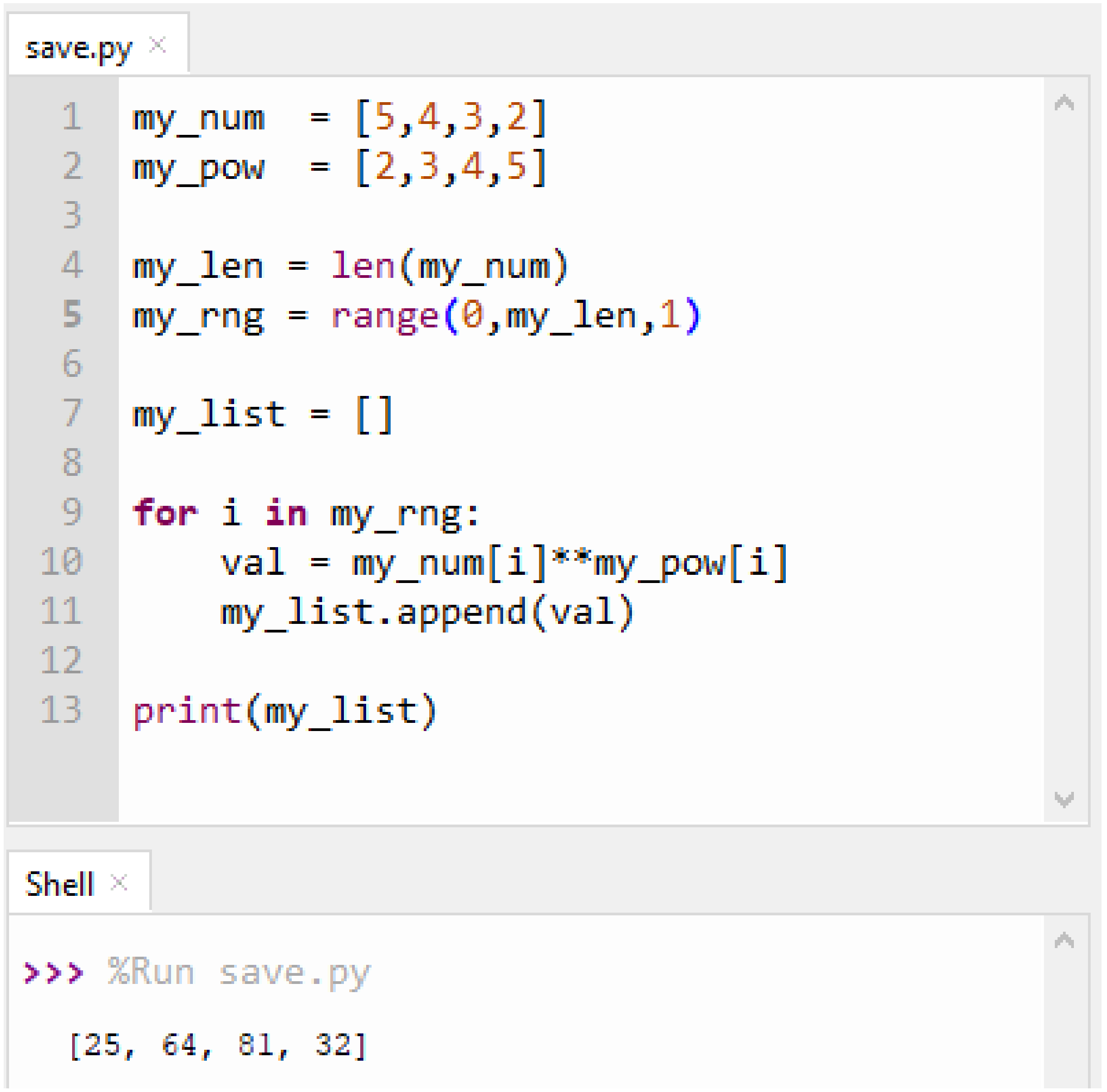
Loops



What if...



You could build an Index loop… or…



```python
my_num  = [5,4,3,2]
my_pow  = [2,3,4,5]

my_len = len(my_num)
my_rng = range(0,my_len,1)

my_list = []

for i in my_rng:
    val = my_num[i]**my_pow[i]
    my_list.append(val)

print(my_list)
```

```
>>> %Run save.py
  [25, 64, 81, 32]
```

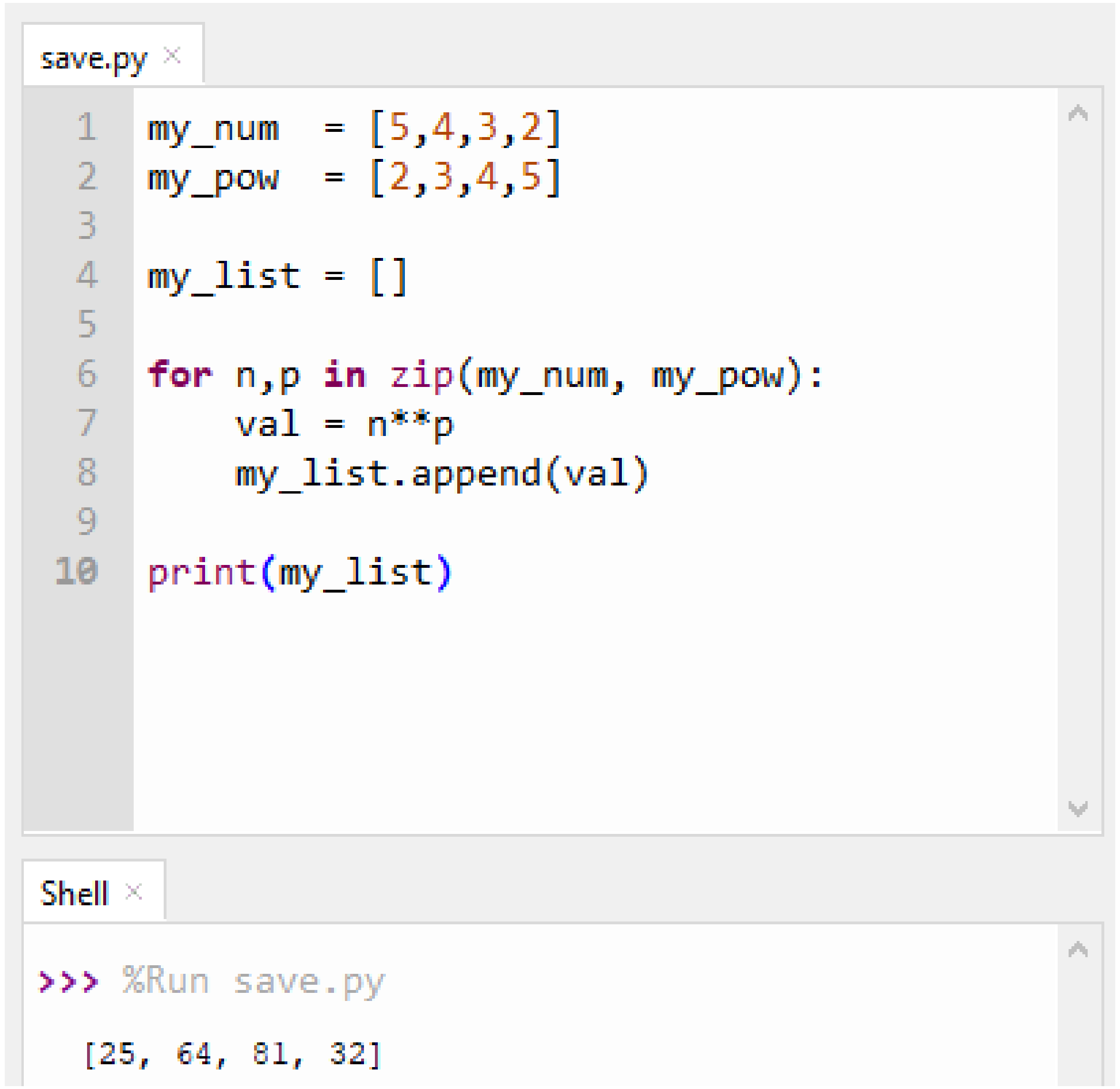
use
Zip Iteration

Syntax

for x,y **in zip(**obj,obj**)**:
(tab)code

Zip can support any number of variables and keeps lists in parallel.

Example


save.py

```python
my_num  = [5,4,3,2]
my_pow  = [2,3,4,5]

my_list = []

for n,p in zip(my_num, my_pow):
    val = n**p
    my_list.append(val)

print(my_list)
```

Shell

```
>>> %Run save.py

  [25, 64, 81, 32]
```

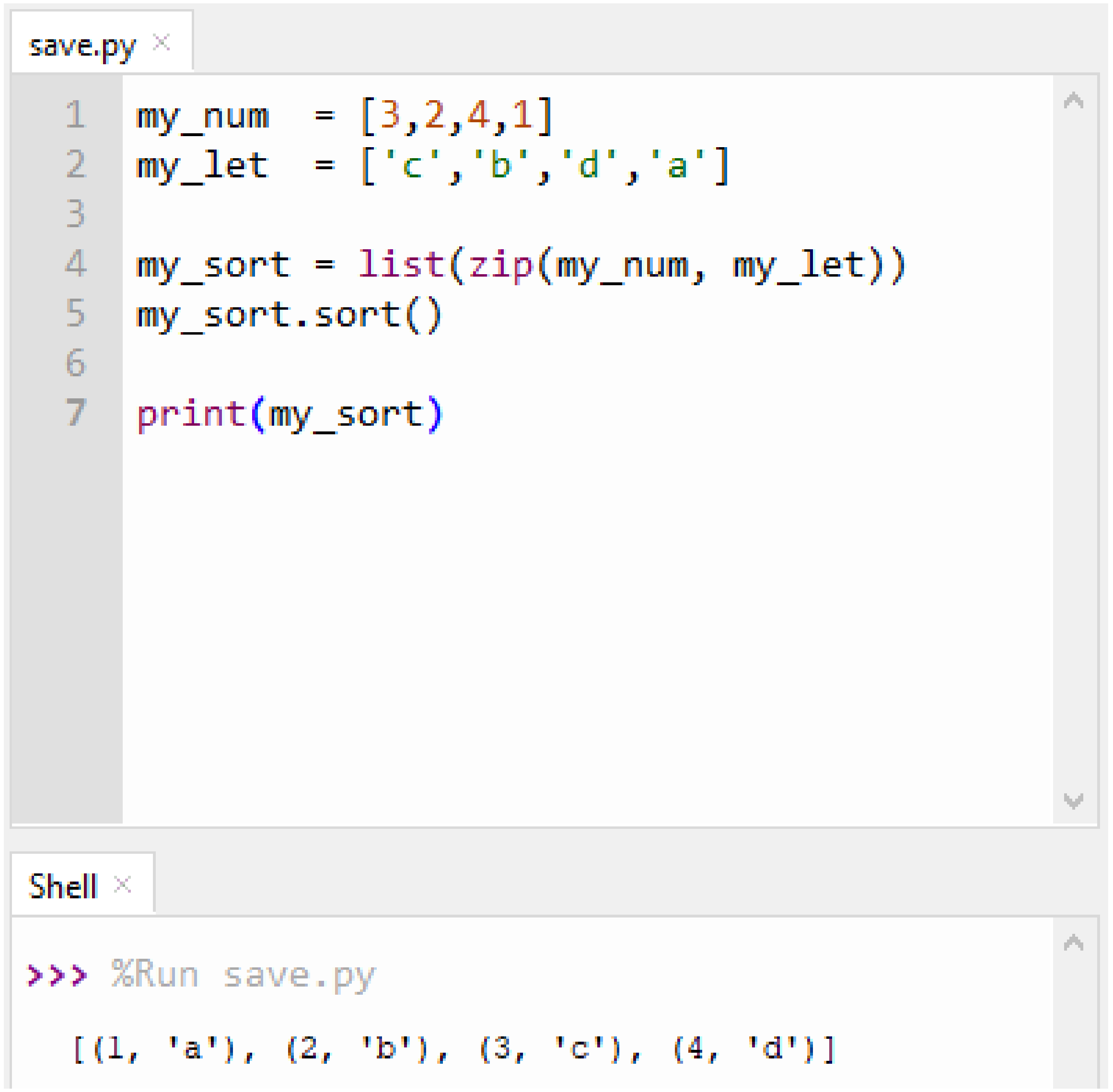
Zip sorting

Function

list(zip(keys, sort**))**

The lists will be paired into **tuples**.

Example



```
my_num  = [3,2,4,1]
my_let  = ['c','b','d','a']

my_sort = list(zip(my_num, my_let))
my_sort.sort()

print(my_sort)
```

```
>>> %Run save.py
  [(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```

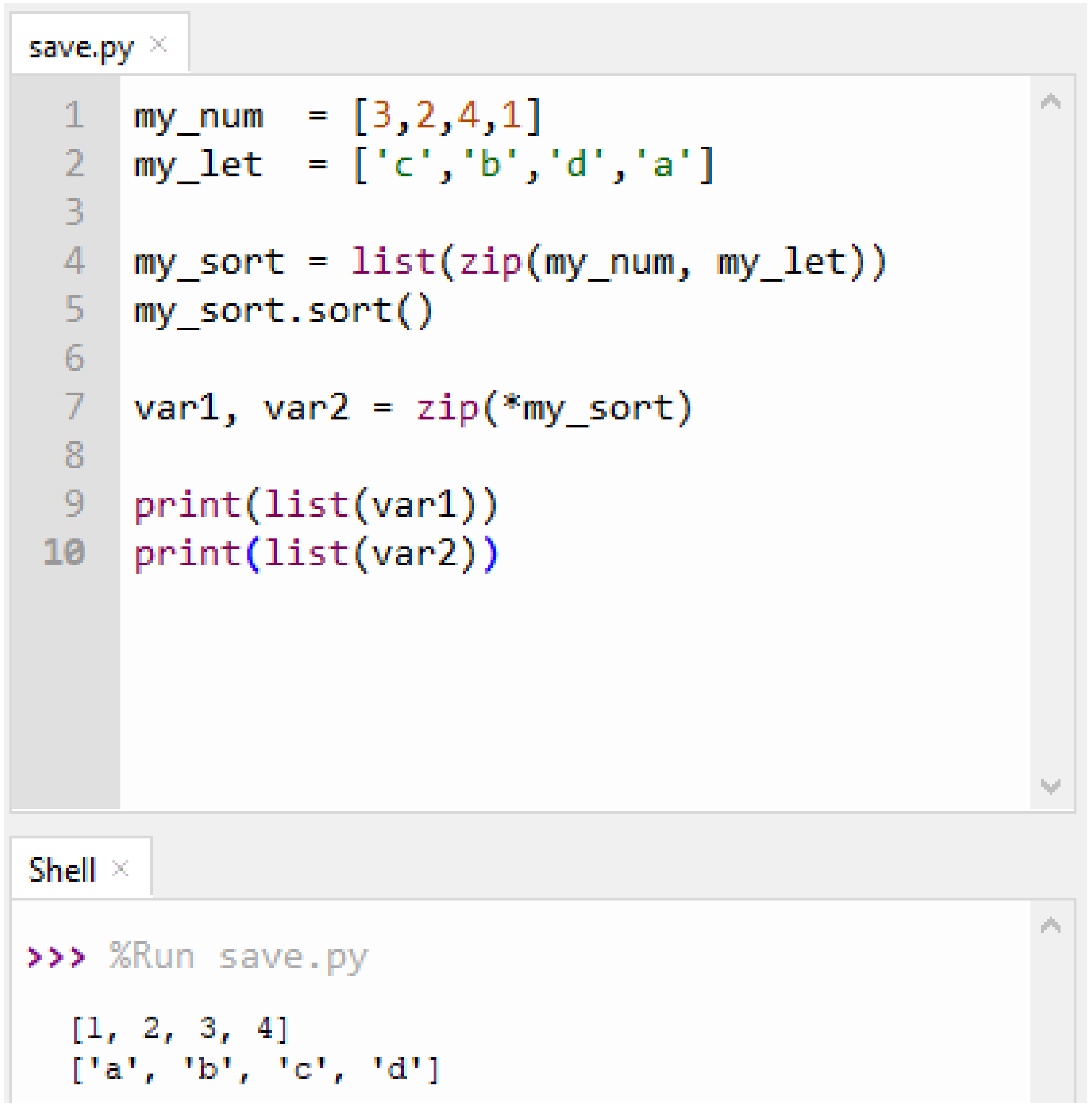
Unzipping

Syntax

```
var1, var2 = zip(*list)
```

var1 and var2 are brand new variables to assign the lists back to.

Example



```
my_num  = [3,2,4,1]
my_let  = ['c','b','d','a']

my_sort = list(zip(my_num, my_let))
my_sort.sort()

var1, var2 = zip(*my_sort)

print(list(var1))
print(list(var2))
```

```
>>> %Run save.py

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

More uses for zip

Parallel functions
e.g. Arithmetic

Combining lists into
Dictionaries

Next on #10

Putting it all together!

9

Python Quick Tips

Zip Iteration