In [1]:

```
# Some ways to iterate over an array
# By Tor Olav Kristensen, 2021-04-06
```

In [2]:

```
import itertools
import numpy
```

In [3]:

```
array = \
    numpy.array(
        [
            [
                [ -6, +1, -5 ],
                [ +2, -4, +3 ],
            ],
            [
                [ -3, +4, -2 ],
                [ +5, -1, +6 ],
            ]
        ]
    )
array
```

Out[3]:

```
array([[[-6,  1, -5],
        [ 2, -4,  3]],

       [[-3,  4, -2],
        [ 5, -1,  6]]])
```

In [4]:

```
sizes = array.shape
sizes
```

Out[4]:

```
(2, 2, 3)
```

In [5]:

```
dimensions = len(sizes)
dimensions
```

Out[5]:

```
3
```

In [6]:

```
for element in numpy.nditer(array):
    print(element)
```

```
-6
1
-5
2
-4
3
-3
4
-2
5
-1
6
```

In [7]:

```
ranges = tuple(range(s) for s in sizes)
ranges
```

Out[7]:

```
(range(0, 2), range(0, 2), range(0, 3))
```

In [8]:

```
for index in itertools.product(*ranges):
    print(index, array[index])
```

```
(0, 0, 0) -6
(0, 0, 1) 1
(0, 0, 2) -5
(0, 1, 0) 2
(0, 1, 1) -4
(0, 1, 2) 3
(1, 0, 0) -3
(1, 0, 1) 4
(1, 0, 2) -2
(1, 1, 0) 5
(1, 1, 1) -1
(1, 1, 2) 6
```

In [9]:

```
size_products = [ 1 ]
for s in reversed(sizes):
    size_products.insert(0, s * size_products[0])
size_products
```

Out[9]:

```
[12, 6, 3, 1]
```

In [10]:

```
for count in range(size_products[0]):
    remainder = count
    quotients = [ ]
    for p in size_products:
        quotients.append(remainder // p)
        remainder %= p
    index = tuple(quotients[1:])
    print(index, array[index])
```

```
(0, 0, 0) -6
(0, 0, 1) 1
(0, 0, 2) -5
(0, 1, 0) 2
(0, 1, 1) -4
(0, 1, 2) 3
(1, 0, 0) -3
(1, 0, 1) 4
(1, 0, 2) -2
(1, 1, 0) 5
(1, 1, 1) -1
(1, 1, 2) 6
```