Introduction

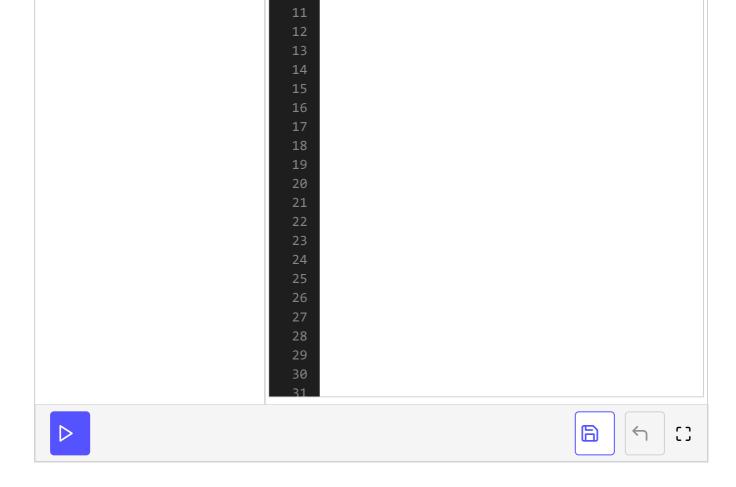In this lesson, we'll learn how to maximize the speed using NumPy!

This chapter explains the basic anatomy of NumPy arrays, especially regarding the memory layout, view, copy and the data type. They are critical notions to understand if you want your computation to benefit from NumPy philosophy.

Let's consider a simple example where we want to clear all the values from an array which has the data type `np.float32`. How does one write it to **maximize speed**? The below syntax is rather obvious (at least for those familiar with NumPy) but the above question asks to find the fastest operation.

```
1  import numpy as np
2  Z = np.ones(4*1000000, np.float
3  print(Z)
4  Z[...] = 0 #clear the array,se
5  print(Z)
6  print(Z.dtype)#prints the data
```

If you look more closely at both the `dtype` and the size of the array, you can observe that this array can be casted (i.e. viewed) into many other "compatible" data types. By compatible, I mean that `Z.size * Z.itemsize` can be divided by the new `dtype` `itemsize`.

main.py

tools.py

```
1
2
3
4
5
6
7
8
9
10
```

```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

Here timeit is a custom function used. Interestingly enough, the obvious way of clearing all the values is not the fastest. The total number of CPU cycle to execute each above instruction are 100 but the two instruction take less time per loop. By casting the array into a larger data type such as `np.float64`, we gained a **25% speed factor**. But, by viewing the array as a byte array (`np.int8`), we gained a **50% factor**. The reason for such speedup is to be found in the internal NumPy machinery and the compiler optimization.

Solve this Quiz !

Q

How can you increase the speed factor for clearing data from an array(setting all values in an array to 0)?

```
Z = np.ones(4*1000000, np.float32)
```

A)
```
timeit("Z.view(np.float64)[...] = 0", globals())
```

B)

```
timeit("Z.view(np.float16)[...] = 0", globals())
```

CHECK ANSWERS

This simple example illustrates the philosophy of NumPy. Let's move on to the next lesson to learn memory layouts.