q1

February 11, 2021

[1]:
```
from timeit import default_timer
from numpy import array
from numpy.random import random
```

Part A: Returns a numpy array whose elements are the squares of a given array (construct in two ways)

[2]:
```
# a function squares each element of the original array
def square_array(original_array):

    squared_array = list(map(lambda x: x**2, original_array))
    square_array = array(squared_array)

    return squared_array
```

[3]:
```
my_array = array([1, 2, 3, 4, 5, 7, 8, 9, 10])
squared = square_array(my_array)
```

[4]:
```
squared
```

[4]:
```
[1, 4, 9, 16, 25, 49, 64, 81, 100]
```

[5]:
```
# a function that uses direct numpy syntax to construct a new array
def square_array_numpy(original_array):

    squared_array = original_array**2

    return squared_array
```

[6]:
```
my_array = array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
squared = square_array_numpy(my_array)
```

[7]:
```
squared
```

[7]:
```
array([  1,   4,   9,  16,  25,  36,  49,  64,  81, 100], dtype=int32)
```

Part B: Construct a one-dimensional numpy array of length 10^4 and reports (for each of the array squaring functions) the time taken by 1000 consecutive calls to that function

[8]:
```
# a function that reports the runtime for the two above functions after 1000␣
↪consecutive calls
def report_runtimes(random_array_size, num_iterations):

    random_array = random([random_array_size])
    print(random_array, random_array.size)

    timer_start_function1 = default_timer()

    # run the function that iterates over the original array
    for _ in range(num_iterations):
        square_array(random_array)

    timer_end_function1 = default_timer()
    timer_start_function2 = default_timer()

    # run the function that uses numpy element-wise multiplication
    for _ in range(num_iterations):
        square_array_numpy(random_array)

    timer_end_function2 = default_timer()

    # use values of the default timer to report the runtimes for each function
    print("Avg. time (1 call) w/ iteration: {:.8f} seconds".
↪format((timer_end_function1 - timer_start_function1)))
    print("Avg. time (1 call) w/ numpy squaring: {:.8f} seconds".
↪format((timer_end_function2 - timer_start_function2)))
```

[9]:
```
report_runtimes(10**4, 1000)
```

```
[0.86276176 0.74465247 0.90752341 … 0.56539004 0.20987613 0.0474892 ] 10000
Avg. time (1 call) w/ iteration: 7.67086260 seconds
Avg. time (1 call) w/ numpy squaring: 0.00561270 seconds
```