

0.1 Introduction to Python

- Install Python 2.7 on your computer (c.f. <https://wiki.python.org/moin/BeginnersGuide/Download>).
- If you have not yet worked with Python, reproduce the first 7 Chapters of the Python tutorial <https://docs.python.org/2.7/tutorial/>. (You can omit sections 2.2, 4.7, 4.8, 5.8, 6.4 since they discuss more advanced topics)
- Implement the Seive of Eratosthenes https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes, an ancient algorithm for finding prime numbers.
- Using your implementation, find the sum of all the primes below two million (<https://projecteuler.net/problem=10>).

0.2 Values and References

The difference between assignment by reference or assignment by value in programming languages can be explained using the following analogy: When browsing the Internet, you might hit a page that you want to store for later use. In this case, you can either bookmark it (i.e., save its URL to your disk) or save the complete page to your disk.

This distinction exists in high-level programming languages as well. There, if you assign a variable to another variable you can either copy the value or copy the memory location address.

- Describe the advantages and disadvantages of both behaviors in terms of memory usage, data consistency and the possibility that the original data source might disappear.

In Python, variable assignments are by reference ¹ by default (except for number types, such as bool, int and float).

As examples, take the following excerpts from a Stackoverflow question <https://stackoverflow.com/questions/13530998/python-variables-are-pointers>:

```
i = 5      # create int(5) instance, bind it to i
j = i      # bind j to the same int as i
j = 3      # create int(3) instance, bind it to j
print i    # i still bound to the int(5), j bound to the int(3)

i = [1,2,3] # create the list instance, and bind it to i
j = i       # bind j to the same list as i
i[0] = 5    # change the first item of i
print j     # j is still bound to the same list as i
```

However, by using the `copy` module, you can enforce assignment by value. The distinction between value and reference assignment is reflected in the existence of the two comparison operators `==` and `is`. The `==` operator compares the values of two variables while the `is` operator checks whether two variables reference the same object. Given the following definitions:

¹In other programming languages (like C++), there is a distinction between references and pointers. In Python variable hold what is called a reference to an object. They should however not be mistaken for C++ references or pointers. See <http://scottlobdell.me/2013/08/understanding-python-variables-as-pointers/> and https://www.tutorialspoint.com/cplusplus/cpp_references.htm.

```
import copy

a = [1, 2, 3, 4, 5]
b = a
c = copy.deepcopy(a)
```

Write down the results of the following statements (without using your Python interpreter):

- b) `print a == b, a == c, b == c`
- c) `print a is b, a is c, b is c`

Now, we'll change a: `a[0] = 42`. What do the statements now result in:

- d) `print a == b, a == c, b == c`
- e) `print a is b, a is c, b is c`
- f) `print a[0], b[0], c[0]`

- g) What's the difference between `deepcopy` and `copy`? Could we have used `copy` for our example as well?

0.3 Matrix multiplication and numpy

This exercise introduces you to a very important Python package called `numpy`. With it, you can create and efficiently handle large, multi-dimensional arrays and do mathematical operations on them.

Your task will be to compare the run-time of the matrix multiplication in `numpy` with an implementation that you will do yourself.

- a) Write a function that creates a random $N \times M$ matrix as a `numpy` array (see <https://docs.scipy.org/doc/numpy/user/basics.creation.html#arrays-creation> and <https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randint.html#numpy.random.randint>)
- b) Implement a matrix multiplication using `for` loops.
- c) Compare your implementation's results with that of `numpy` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html>). Hint: `numpy` arrays can be compared with the `==` operator, using their `all()` method.
- d) Measure the runtime of both implementations. For this, you can either use the `timeit` package or the following approach (you might have to introduce some repetitions to get a meaningful measurement):

```
import time

start = time.time()
print("hello")
end = time.time()
print(end - start)
```