- -> notebooks from this lecture: https://github.com/ine-rmotr-curriculum/freecodecamp-intro-to-numpy
- -> linear algebra
- ***-> numpy already contains linear algebra operations***
  - dot products
  - cross produces
  - transposing matrices
- -> in Python -> 28 bytes is the regular size
- -> you need 28 bytes to store the number 1 in Python
- -> this isn't efficient enough
- -> in numpy, we are allowed to pick the number of bytes
- -> the difference between the size of an integer in Python and numpy
- ***-> focusing on performance <- Python vs numpy***
  - -> he's taken a Python array and a numpy array and is performing the same operation on them (squaring them and summing the elements)
  - -> it's a lot faster in numpy than it is with Python
  - -> this is only in terms of a smaller array size
- -> next -> pandas
- ***-> question***

What is the relationship between size of objects (such as lists and datatypes) in memory in Python's standard library and the NumPy library? Knowing this, what are the implications for performance?

Standard Python objects take up much more memory to store than NumPy objects; operations on comparable standard Python and NumPy objects complete in roughly the same time.

NumPy objects take up much more memory than standard Python objects; operations on NumPy objects complete very quickly compared to comparable objects in standard Python.

NumPy objects take up much less memory than Standard Python objects; operations on Standard Python objects complete very quickly compared to comparable objects on NumPy Object.

Standard Python objects take up more memory than NumPy objects; operations on NumPy objects complete very quickly compared to comparable objects in standard Python. <u>*<- This one, objects in numpy take up less space than Python objects - and so operations on them are faster. Python is a higher level programming language*</u>