Optimization: Python Optimization

CPSC 501: Advanced Programming Techniques
Winter 2025

Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Wednesday, March 5, 2025

UNIVERSITY OF CALGARY

Python Specific Optimizations

Code Tuning – Python

- Python is an object oriented language
- That runs in a virtual machine
- There are more inefficiencies that can be improved than we've covered for a language like c++
- Like Java Strings are objects and are slow
- Similar to Java int/double are like BigInteger/Double and are slow (but we have no standard primitive to swap them with)
- Tuning __hash__/__eq__ useful still

Code Tuning – Logging

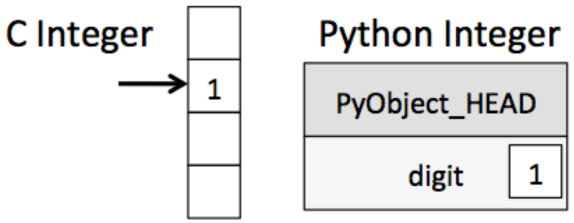
- Strings take a lot of time to create (program-wise)
- Ex. Again check if you need to make a string before you make it to print it
- Each string is immutable, a new concatenated one takes up space of old one plus space of added parts, and everything must be copied over

Code Tuning – Libraries

- Use numpy!
- Numpy has replacements for python list which is like an ArrayList
- It turns it into something closer to a primitive Java array
- In affect it uses C++ arrays with Python

integers?

- **Python** integer is more than an integer
- **Python** 3.4 integer

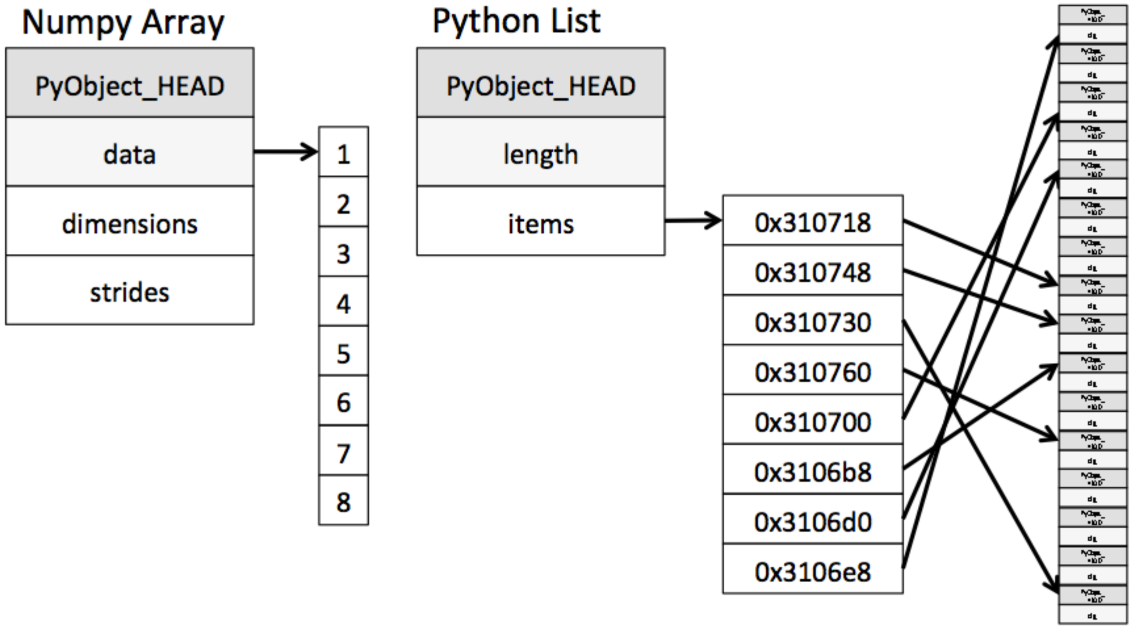



```c
struct _longobject {
  long ob_refcnt;
  PyTypeObject *ob_type;
  size_t ob_size;
  long ob_digit[1];
}
```

- A **C** integer is essentially a label for a position in memory whose bytes encode an integer value.
- A **Python** integer is a pointer to a position in memory containing all the Python object information, including the bytes that contain the integer value.

lists?

- A **Python** list is more than just a list of values



- A **Python** list contains a pointer to a block of pointers, each of which in turn points to a full **Python** object like the **Python** integer we saw earlier.
- **numpy** arrays are a single pointer to a block of contiguous data.

UNIVERSITY OF CALGARY

numpy

- **Num**erical **Py**thon library
- More efficient data and storage operations as arrays grow larger
- **Python** integer is more than an integer, and a list more than just values
- **numpy** arrays allow us to put data all in one place and drastically improve the our ability to manipulate it quickly.
- In essences what **numpy** does is provide a portal from **Python** through to **C** implementations of storage arrays, allowing us to access the strengths of that language in ways not normally available in **Python**.

```
import numpy as np
```

Speed

- numpy operations speed comes from a large variety of Universal Functions (Ufuncs) which are ‘vectorized operations’ designed to run at higher speed than if we let Python’s loops manage things
- Both compute same answer!

```
[48] import numpy as np
     np.random.seed(0)

     def compute_reciprocals(values):
         output = np.empty(len(values))
         for i in range(len(values)):
             output[i] = 1.0 / values[i]
         return output

     values = np.random.randint(1, 10, size=5)

     print(compute_reciprocals(values))
     print(1.0 / values)
```

```
[0.16666667 1.         0.25       0.25       0.125     ]
[0.16666667 1.         0.25       0.25       0.125     ]
```

Speed

- Both compute same answer!
- Milliseconds versus microseconds (a difference in order of magnitude of 1000 here)

```
big_array = np.random.randint(1, 100, size=100000)
print("Python")
%timeit compute_reciprocals(big_array)
print("numpy")
%timeit (1.0 / big_array)
```

```
Python
1 loop, best of 5: 225 ms per loop
numpy
The slowest run took 4.95 times longer t
1000 loops, best of 5: 227 µs per loop
```

Memory

Code Tuning – Memory Leaks

- Python is stuck with garbage collection
- We can stop pointing at things but not delete them
- If your program naively leaves created objects connected to current code (heap will continue to grow)
- You can generally see this via Profiling and heap dumps
- Useful to consider making sure you =null out pointers of large amounts of data you want to be cleaned up (otherwise they may hang around), returning from a function often solves this for local variables
- Other tunings for locking, timeout, caching apply similarly like Java

Onward to … digital signal processing.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY