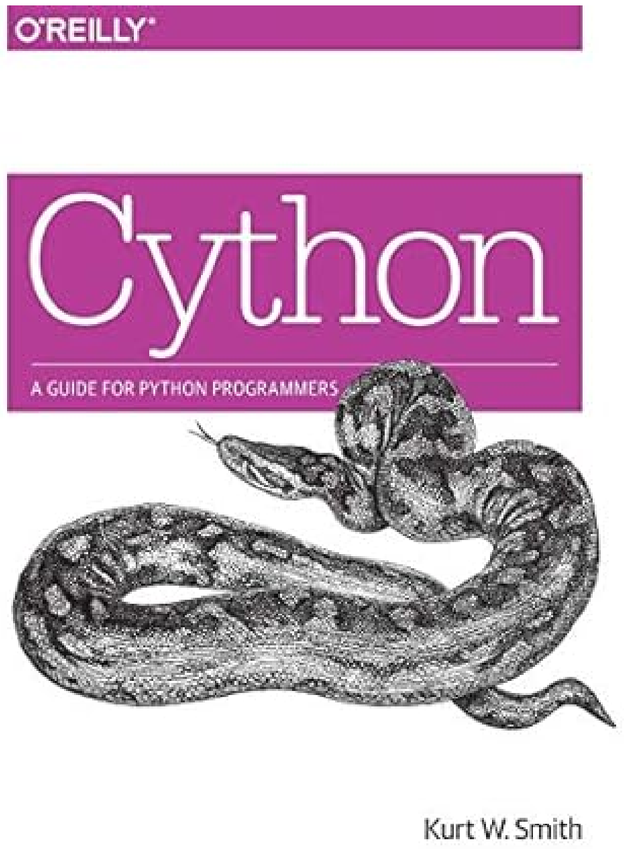
Xebia

RADICALLY IMPROVE PYTHON PERFORMANCE WITH CYTHON

Eric Greene
eric@cloudcontraptions.com

Learn More About Cython



Cython

by Kurt W. Smith

GitHub Examples: https://github.com/cythonbook/examples

Xebia

Goals for this Session

- Learn why Python is slow
- Explore a Python C Extension
- Cython to the Rescue
- Explore how Cython can be used to improve Python App performance

Xebia

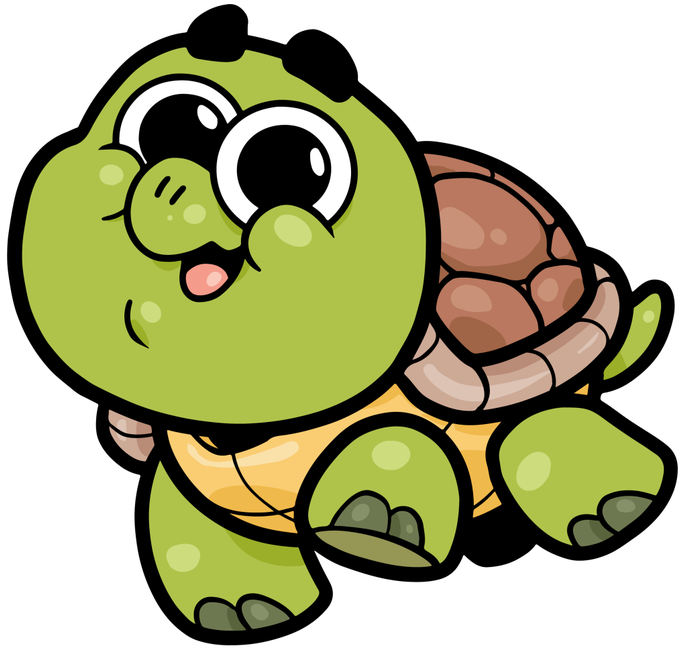
Cython Topics

- Why is Python slow?
- Improve Performance with Custom C Extensions
- Challenges of Custom C Extensions
- Using Cython
- Cython Development Tools
- Explore Cython Examples

Xebia

Why is Python slow?

- Interpreted language
- Dynamic typing
- Global Interpreter Lock (GIL)
- Memory management
- Lack of low-level optimization



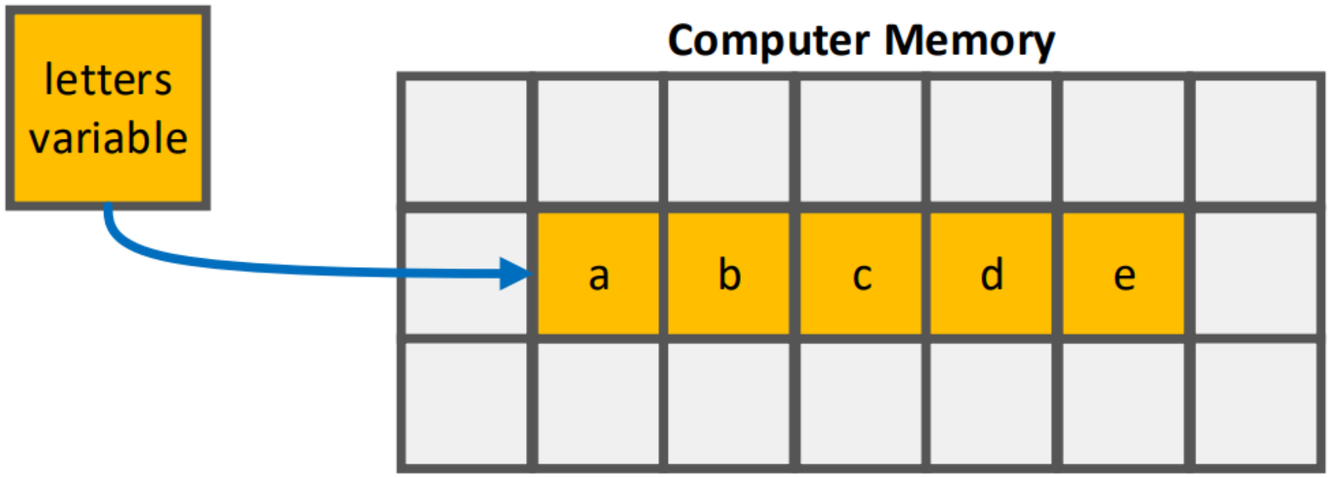
Xebia

Slow Python: An Example with the List Object

Consider the following Python List of letters:

```
letters = ['a', 'b', 'c', 'd', 'e']
```

How is this stored in memory? Are you thinking of something similar to this?



Guess what? Not even close...

Xebia

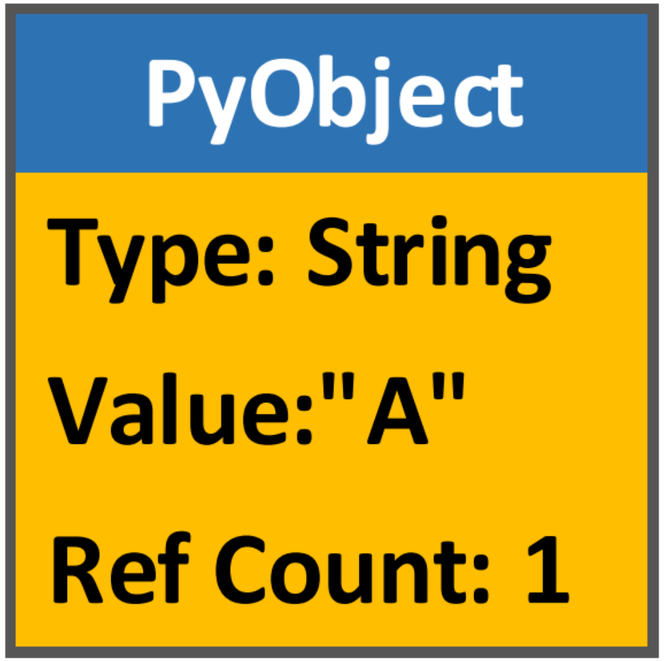
Slow Python: Python Object

In Python, all data is an object including integers, strings, floats, etc.

Each object tracks several things including the type (int, str, float), value (1, 'a', 3.2), and the reference count (used for garbage collection)

For each object accessed at runtime, the object must be loaded from memory, its type must be examined, and then its value interpreted according to the type - this happens each time the data is used at runtime

All of this complexity is needed to make Python a dynamically typed language which makes the programmer's job a little easier at development time



Xebia

Slow Python: Python List Object

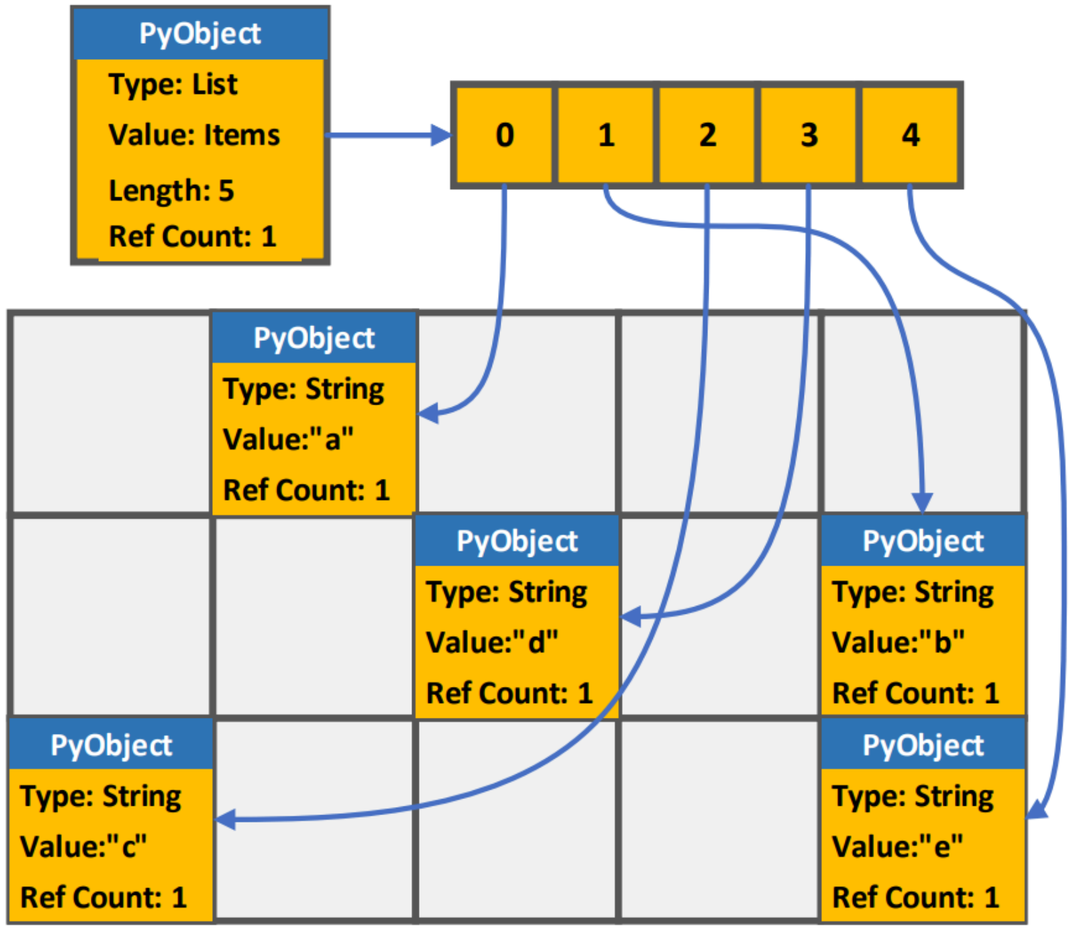
Consider the previous Python List of letters:

```
letters = ['a', 'b', 'c', 'd', 'e']
```

So how is this truly stored in memory?

Crazy, right?

As each element of the list is accessed,
the list's PyObject must be dynamically
examined, then the element must be retrieved,
then the element's PyObject must be dynamically
examined, and the value returned



Xebia

Is Python Slowness a Real Problem?

- Not Always
  - If the Python program primarily does IO operations, then the real speed limitation is the IO, not Python
  - The most expensive aspect of programming is paying programmers, Python can be easier and quicker to work with in many situations
- But Sometimes
  - Python can struggle a lot with computational and memory-intensive tasks
  - When Python is used for such tasks, libraries written in C/C++ wrapped in Python are used

Python Performance Improving Techniques

- Specialized data structures built into Python and popular packages such a Numpy
  - Ex. array module, memory views, NumPy arrays
- CTypes
  - Call functions in a compiled C library directly
- C Extensions
  - A formal way to implement high-performance code in C that ties into the Python runtime system
- Cython
  - An optimizing static compiler that enables the programmer to write high-performance with Python itself

Xebia

Cython Overview

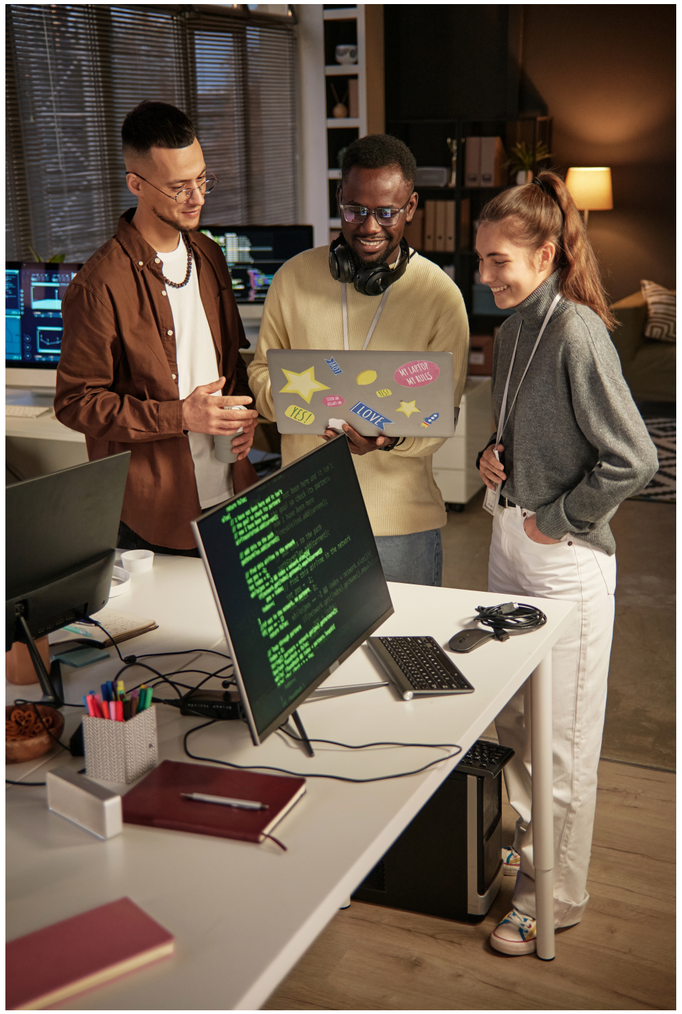
- Compiles Python-like code to a C extension
- Simplifies the creation of C extensions
- Provides code decorators to optimize Python code
- Leverage Python type hints to optimize code as well
- Excellent Cython, C, Python, and Python C API support in AI tools such as GitHub Copilot and ChatGPT
- Knowledge of C is helpful for reviewing generated code, but not essential for getting started

```
def do_double_nums(nums: list[int]) -> list[int]:
    cdef long i = 0
    cdef long nums_length = len(nums)
    cdef list result = []

    for i in range(nums_length):
        result.append(nums[i] * 2)

    return result
```

The code above is a mix of Python, Python Type Hints and Cython code decorations.

Xebia

Cython Demos



- Explore Python Performance
- Review C Extension
- Hello, World Cython
- Simple Build System
- Packaging Cython Modules
- Lists and Iteration
- Functions
- Performance Comparison
- Mandelbrot Set Example

Xebia

Programming Demo

Let's Explore Cython

Xebia

Cython Programming Next Steps

- Read the Cython documentation
  - https://cython.org
- Read a book on Cython programming, such as "Cython"
- Run the code from the webinar as GitHub Codespace and explore it on your own
- Incorporate it into your next project!

Xebia

Download the Code

github.com/cc-xebia-webinars/cython-webinar_04022024

slides and source code available

Xebia

Q&A

Questions?

Xebia

Thank you!

Eric Greene

eric@cloudcontraptions.com

Xebia