

# Just-in-Time with Numba

Presented by:

Ong Chin Hwee (@ongchinhwee)

28 March 2020

PyLadies International Women's Month Lightning Talks

# About me

**Ong Chin Hwee** 王敬惠

- Data Engineer @ ST Engineering
- Background in aerospace engineering + computational modelling
- Contributor to pandas 1.0 release
- Mentor team at BigDataX

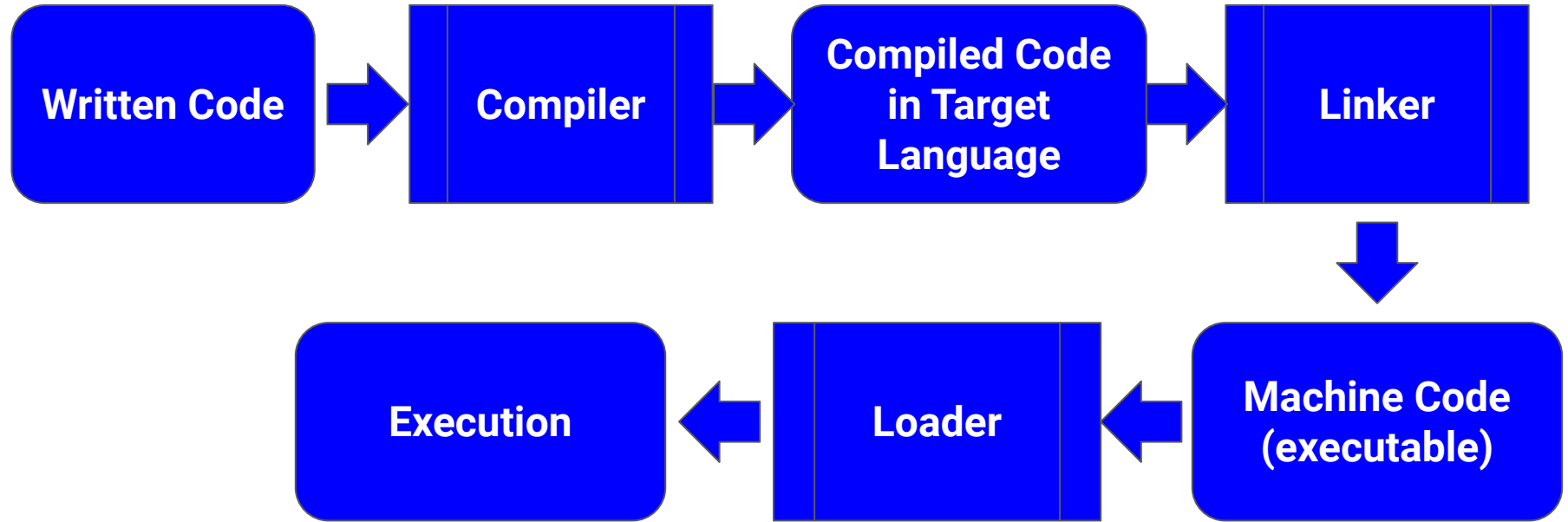


@ongchinhwee

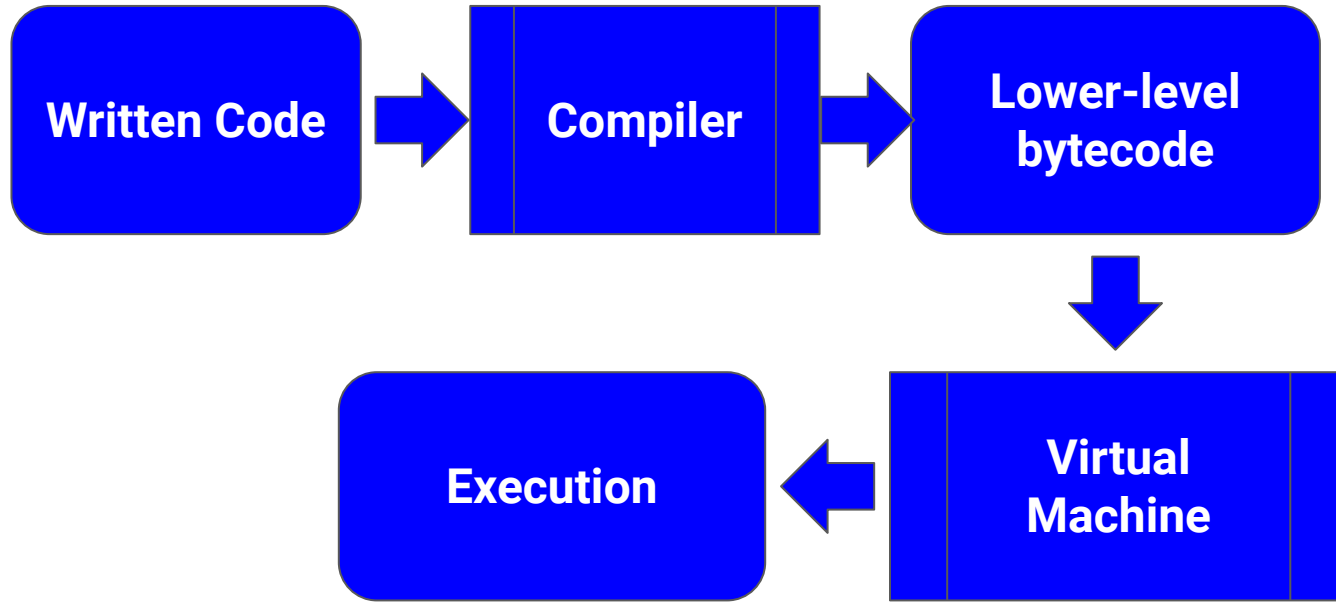
# Bottlenecks in a data science project

- Lack of data / Poor quality data
- Data Preprocessing
  - The 80/20 data science dilemma
    - In reality, it's closer to 90/10
  - Slow processing speeds in Python!
    - Python runs on the **interpreter**, not compiled

# Compiled vs Interpreted Languages



# Compiled vs **Interpreted** Languages



# What is Just-in-Time?

## Just-In-Time (JIT) compilation

- Converts source code into **native machine code** at runtime
- Is the reason why Java runs on a Virtual Machine (JVM) yet has **comparable performance** to compiled languages (C/C++ etc., Go)

# Just-in-Time with Numba

## numba module

- **Just-in-Time (JIT) compiler** for Python that converts Python functions into machine code
- Can be used by simply applying **a decorator (a wrapper)** around functions to instruct numba to compile them
- Two modes of execution:
  - *njit* for no-Python mode (JIT only)
  - *jit* for object mode (JIT + Python interpreter)

# Practical Implementation



# Initialize File List in Directory

```
import numpy as np
```

```
import os  
import sys  
import time
```

No. of images in  
'train/NORMAL': **1431**

```
DIR = './chest_xray/train/NORMAL/'
```

```
train_normal = [DIR + name for name in os.listdir(DIR)  
                 if os.path.isfile(os.path.join(DIR, name))]
```

# With numba

```
from PIL import Image
from numba import jit
```

```
@jit
```

```
def image_proc(index):
    '''Convert + resize image'''
    im = Image.open(define_imagepath(index))
    im = im.convert("RGB")
    im_resized = np.array(im.resize((64, 64)))
    return im_resized
```

@ongchinhwee

# With numba

```
from PIL import Image
```

```
from numba import jit
```

```
@jit
```

```
def image_proc(index):
```

```
    '''Convert + resize image'''
```

```
    im = Image.open(define_imagepath(index))
```

```
    im = im.convert("RGB")
```

```
    im_resized = np.array(im.resize((64, 64)))
```

```
    return im_resized
```

Note: I can't use no-Python mode (@njit) as PIL codes can't seem to be compiled into machine code

@ongchinhwee

# With numba

```
start_cpu_time = time.clock()
```

Python-only:  
**218.1 seconds**

```
listcomp_output = np.array([image_resize(x) for x in  
train_normal])
```

After compilation:  
**169.6 seconds**

```
end_cpu_time = time.clock()  
total_tpe_time = end_cpu_time - start_cpu_time  
sys.stdout.write('List comprehension completed in {}  
seconds.\n'.format(  
    total_tpe_time))
```

# With numba

```
import numpy as np
from numba import njit

@njit
def square(a_list):
    squared_list = []
    '''Resize and reshape image'''
    for x in a_list:
        squared_list.append(np.square(x))
    return squared_list
```

@ongchinhwee

# With numba

```
import numpy as np
```

```
from numba import njit
```

Code runs in no-Python mode  
(@njit or @jit(nopython=true))

```
@njit
```

```
def square(a_list):
```

```
    squared_list = []
```

```
    '''Resize and reshape image'''
```

```
    for x in a_list:
```

```
        squared_list.append(np.square(x))
```

```
    return squared_list
```

@ongchinhwee

# With numba

```
a_list = np.array([i for i in range(1,100000)])
start_cpu_time = time.time()
listcomp_array_output = square(a_list)
end_cpu_time = time.time()
total_tpe_time = end_cpu_time - start_cpu_time
sys.stdout.write(
    'Elapsed (after compilation) {}
seconds.\n'.format(total_tpe_time))
```

Python-only:  
**0.51544 seconds**

After compilation:  
**0.00585 seconds**

# Key Takeaways



# Just-in-Time with numba

- **Just-in-Time (JIT) compilation with numba**
  - converts source code from non-compiled languages into **native machine code** at runtime
  - may not work for some functions/modules - these are still run on the interpreter
  - **significantly enhances** speedups provided by optimized numerical codes

# Reach out to me!



: ongchinhwee



: @ongchinhwee



: hweecat



: <https://ongchinhwee.me>

And check out my slides on:



[hweecat/talk\\_jit-numba](#)

@ongchinhwee