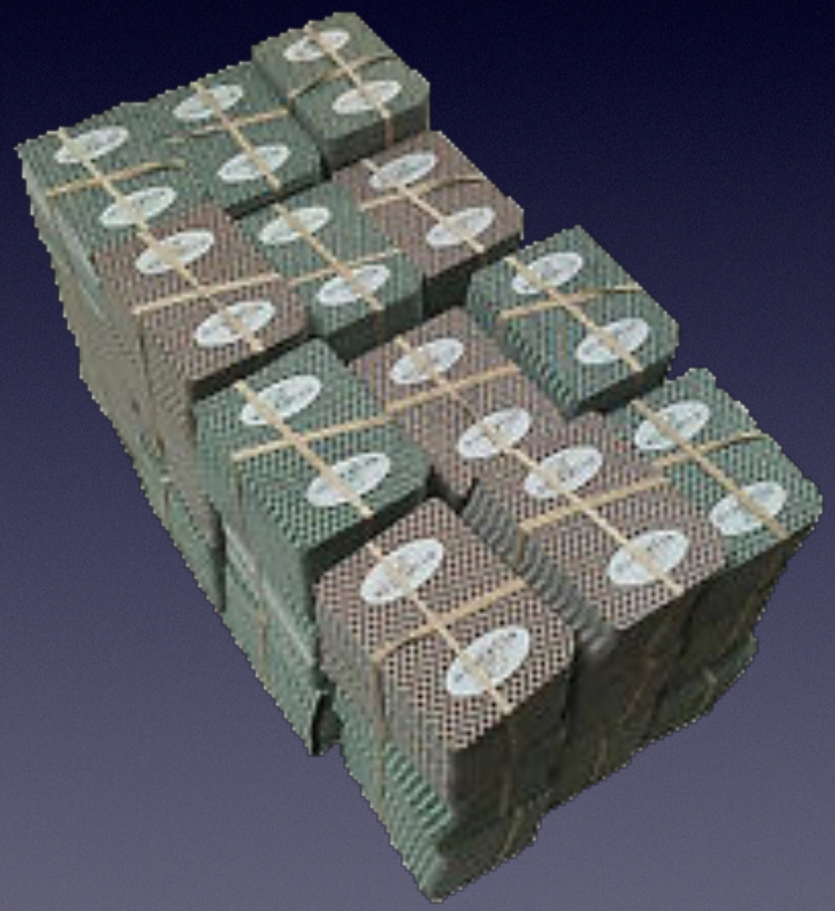
Python:
40% Faster for Free

Brendan J. Herger
13herger@gmail.com
MS Analytics Candidate, USF

Problem

Problem



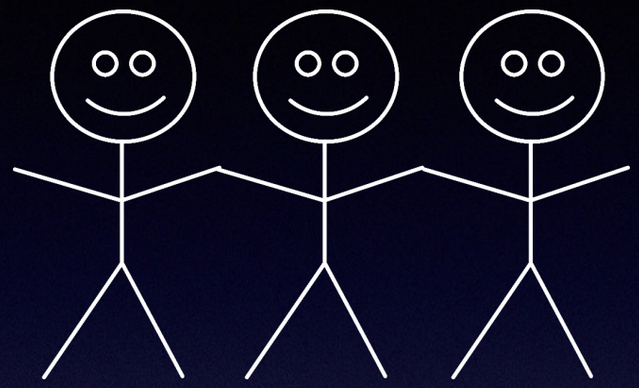
Problem

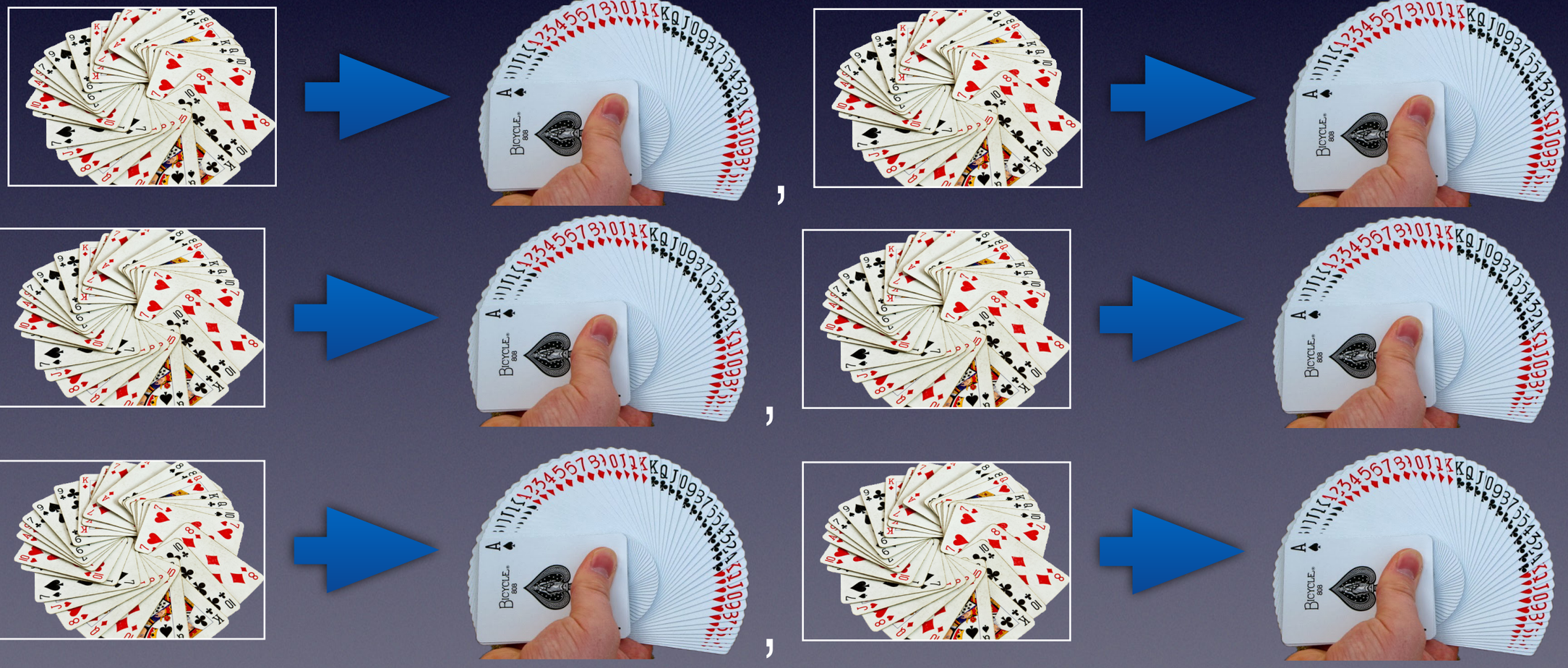
,

. . .

Solution

Solution





Now for Python

Sorting

```
for deck in decks:
    sorted_deck = sorted(deck)
    sorted_decks.append(sorted_deck)
```

```
sorted_decks = map(sorted, decks)
```

```
sorted_decks = bhUtilties.multi_map(sorted, decks)
```

Timing


For Loop
385
Multi-threaded
167
0
100
200
300
400
Time (Seconds)

Sorting

```
for deck in decks:
    sorted_deck = sorted(deck)
    sorted_decks.append(sorted_deck)
```

```
sorted_decks = bhUtilties.multi_map(sorted, decks)
```

Implement

What's Happening?

- Python breaks up task
- Each task is run
- Python re-joins task

When to Use

- When to use: Acting on list elements separately
    - i.e.: reading external files, get word count
- When not to use: any time else

Multithreading Costs

- Overhead
  - splitting up task, rejoining results
  - Can make multithreaded code *slower*

Recap

- Large speed up
- Easy to implement
- Only appropriate in certain cases

Thank you

Brendan J. Herger
13herger@gmail.com
MS Analytics Candidate, USF

Slides, code available at:
https://github.com/bjherger/Multiprocessing-Python