Processing Tasks With Stacks And Queues: Takeaways

Syntax

- Adding an element to the "top" of a stack:

```
stack = [1,2]
stack.insert(0,3)
```

- Removing an element from the "top" of a stack:

```
stack.pop(0)
```

- Creating a stack:

```
class Stack():
    def __init__(self):
        self.items = []
    def push(self, value):
        self.items.insert(0, value)
    def pop(self):
        return self.items.pop(0)
    def count(self):
        return len(self.items)
```

- Adding an element to the bottom of a queue:

```
queue = [1,2]
queue.append(3)
queue.append(4)
```

- Removing the top element of a queue:

```
queue.pop(0)
```

- Creating a queue:

```
class Queue():
    def __init__(self):
        self.items = []
    def push(self, value):
        self.items.append(value)
    def pop(self):
        return self.items.pop(0)
    def count(self):
        return len(self.items)
```

Concepts

- Worker pools work well when you have a fixed amount of work; they do not well when you have work that keeps coming in.

- A stack is a data structure that takes in new elements. Stacks are a way to implement the theoretically more efficient method of prioritization by following a particular order in which the operations are performed.
- A queue is a first in first out system, where the tasks that arrived first get processed first.
- Queues are more "fair" than a stack — all tasks get the same priority, and none of them get processed early.
- Queues are generally best when you want all tasks processed at about the same pace, and queues usually have a fairly low maximum wait time for processing tasks.
- Stacks are generally best when you are okay waiting around while tasks finish processing. Stacks have a fairly high maximum wait time.
- When adding more elements to stacks:
    - Items added to a stack towards the end are processed much faster than items added towards the beginning.
    - Some stack tasks are finished almost immediately after they're added.
    - The worst-case queue time in a stack is equivalent to waiting for every single task to be processed first.
- When adding more elements to queues:
    - Items added to a queue towards the end are processed more slowly than items added earlier (this depends strongly on the throughput of the task processor).
    - Only the first item added to a queue is processed instantly (given that tasks are added faster than they can be processed).
    - The worst-case queue time for a queue depends on the throughput of the task processor.

Resources

- Stack Data Structure
- Queue Data Structure