Identifying runtimes

- Note: typically care about worst-case performance, so $O(f(n))$ notation is most commonly used outside of academic settings
- For-loops are typically $O(n)$, while-loops vary depending on when the loop is broken
- Nested for-loops are typically n^k where k is the number of nested loops
- Accessing array elements by index is $O(1)$
- Searching for an element in an (unsorted) array is $O(n)$
- Binary search is $O(\log(n))$

Example: Fibonacci

```
def fibRecursive(n):
    if n <= 1:
        return 1
    return fibRecursive(n-1) + fibRecursive(n-2)

def fibIterative(n):
    a, b = 1, 1
    for _ in range(n):
        a, b = b, a+b
    return a
```

The recursive version runs in $O(2^n)$ time, but the iterative version is $O(n)$.