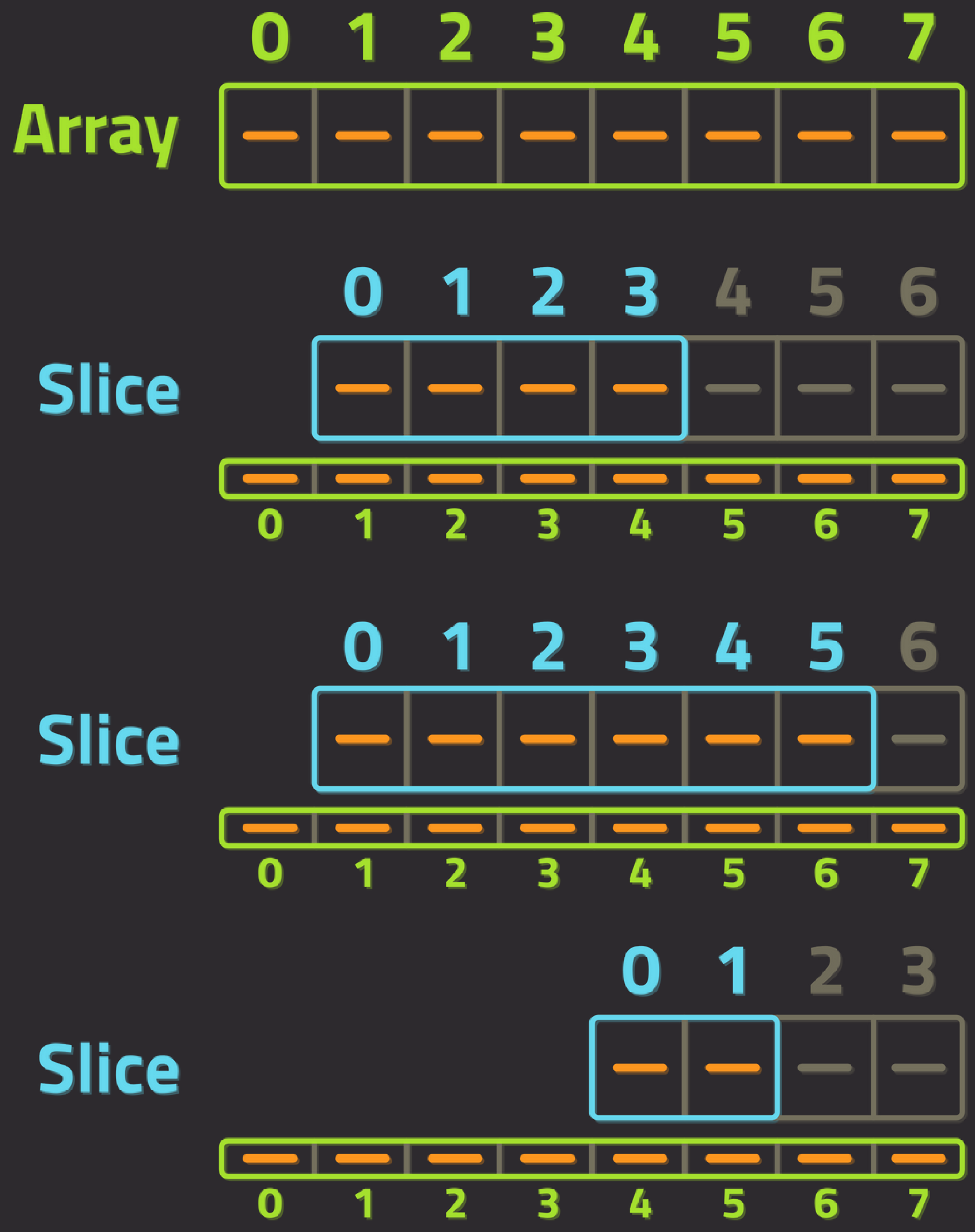
# Slices

## Slices

- Slices are companion types that work with arrays
- They enable a "view" into an array
  - Views are dynamic and not fixed in size
- Functions can accept a slice as a function parameter
  - Any size array can be operated upon via slice

# Visualization



# Creating a Slice

- Slices and an underlying array can be created at the same time

```go
mySlice := []int{1, 2, 3}
```

- Accessing elements in a slice is the same as an array

```go
item1 := mySlice[0]
```

# Slice Syntax

- Slice syntax can create slices from specific elements in an array or other slice

start index (inclusive)

end index (exclusive)

```
slice[a:b]
```

- Omitting **a** means "start at 0"
- Omitting **b** means "to the end"

# Slice Syntax Example

```go
numbers := [...]int{1, 2, 3, 4}

slice1 := numbers[:] // [1, 2, 3, 4]

slice2 := numbers[1:] // [2, 3, 4]
slice3 := slice2[:1]  // [2]

slice4 := numbers[:2] // [1, 2]

slice5 := numbers[1:3] // [2, 3]
```

- It is an error to slice past the array length

# Dynamic Arrays

- Slices can be used to create arrays that can be extended
  - The **append()** function can add additional elements

```
numbers := []int{1, 2, 3}
numbers = append(numbers, 4, 5, 6)
// [1, 2, 3, 4, 5, 6]
```

- 3 dots can be used to extend a slice with another slice

```
part1 := []int{1, 2, 3}
part2 := []int{4, 5, 6}
combined := append(part1, part2...)
```

# Preallocation

- Slices can be preallocated with specific capacities
  - The **make()** function is used to preallocate a slice
- Useful when number of elements is known, but their values are still unknown

```go
slice := make([]int, 10)
```

# Slices to Functions

- Functions parameters which require a slice can work with slices of any size

```go
func iterate(slice []int) {
	for i := 0; i < len(slice); i++ {
		// ..
	}
}
small := []int{1}
big := []int{1, 2, 3, 4, 5, 6, 7}
iterate(small)
iterate(big)
```

# Multidimensional Slices

```go
board := [][]string{
    // type declaration is optional
    []string{"_", "_", "_"},
    {"_", "_", "_"},
    {"_", "_", "_"},
}
board[0][0] = "X"
board[2][2] = "O"
board[1][2] = "X"
board[1][0] = "O"
board[0][2] = "X"
```

## Recap

- Slices are a more convenient way to work with arrays
- Slices can be resized using the **append()** function
- Slices can be created with a special **slice syntax**
- Slice memory can be preallocated using the **make()** function
- Slices always require an underlying array