have a *formula* for `0 + n` in general, we can only use `add_zero` and `add_succ` once we know whether `n` is `0` or a successor. The `induction` tactic splits into these two cases.

The base case will require us to prove `0 + 0 = 0`, and the inductive step will ask us to show that if `0 + d = d` then `0 + succ d = succ d`. Because `0` and successor are the only way to make numbers, this will cover all the cases.

See if you can do your first

Next

**Theorem** `zero_add` : For all natural numbers $n$, we have $0 + n = n$.

Active Goal

Objects:

**n** : ℕ

Goal:

```
0 + n = n
```

```
induction n with d hd
```

Retry

Active Goal    Goal 2

Goal:

```
0 + 0 = 0
```

```
rw [add_zero]
```

Retry

Active Goal    Goal 2

Goal:

```
0 = 0
```

```
rfl
```

Retry

```
intermediate goal solved! 🎉
```

Active Goal

Objects:

**d** : ℕ

Assumptions:

**hd** : `0 + d = d`

Goal:

## add_succ

```
(a d : ℕ) : a + MyNat.succ d = MyNat.succ (a + d)
```

`add_succ a b` is the proof of `a + succ b = succ (a + b)`.