Syntax in functions.

1. Pattern matching → consists of specifying patterns to which some data should conform and then checking to see if it does.

Deconstructing the data according to these patterns.

↓

when defining functions, we can define separate fn. bodies for different patterns

```haskell
sayMe :: (Integral a) => a -> String
sayMe 1 = "One!"
sayMe 2 = "Two!"
sayMe 3 = "Three!"
sayMe 4 = "Four!"
sayMe 5 = "Five!"
sayMe x = "Not between 1 and 5"
```

↓

factorial fn:

```haskell
factorial :: (Integral a) => a -> a
factorial 0 = 1
factorial n = n * factorial (n-1)
```

→ recursion

↓

important concept in Haskell.

↓

to remember: when making patterns, we should always include a catch-all pattern so that our program doesn't crash if we get some unexpected o/p.