Type Aliases !

Most problem domains need specific data types

- A data type communicates a value's attributes and capabilities
  - eg: if you have a value of type int, you know you can do arithmetic with it
  - So far, you've used built-in data types
- Your programs will need to **model** more specific and complex concepts
- **Type Aliases** allow us to assign a more meaningful name to a data type
- **Composite Data Types** allow us to combine many values into a single type

Type Aliases

- An alias allows you to use a contextually meaningful name for broader type. Ex:

```
Percent = float
cointoss_odds: Percent = 0.5
```

- The above example establish `Percent` as a type alias of `float`
  - Notice you can declare a variable whose type is Percent (actually: `float`)
  - You can also declare parameters and return types as `Percent`!
  - You've "invented" a "new" type!

- When in doubt, ***do not*** alias simple primitive types: `int`, `float`, `bool`, `str`
  - This technique is most useful for more complex types such as `Tuple[float, float, float]`

- Custom type names are identifiers that begin new words with `CapitalLetters`
  - This is called `CamelCasing`