team treehouse: error handling

types of errors

- what are **errors**?
  * an **error** is **any code** that **produces** an **incorrect** or **unexpected result**, or **causes** your **program** to **behave** in an **unintended manner**
- let's start with simplest of errors

domain errors

- common example occurs often when we want to *convert between strings* and *numbers*

compiler errors

- these are the **errors** that the **swift compiler** raises as it **parses** your **code** and **prevents** that **code** from **running**
  * since *optionals* have *great compiler support*, **domain errors**, simple ones at least, can be **considered compiler errors** in *swift* because **optionals handle** them
- there are **errors** that the **compiler cannot catch for you**, and they only **occur** when you **run your program** and **hit** a **particular line** of **code**
  * these are called **runtime errors** and we can **react** to them in **two ways** depending on **what kind of error occured**
    ** **recoverable errors**
    ** **unrecoverable errors** or **failures**
  * these are of **two sub types**
    ** **logic errors** that can occur from a *programmer mistake*
    ** and **universal errors** where we **cannot anticipate** the **error occuring**
      ** it's something that the *system generates* for example

modelling errors

- in *swift* **errors** are **represented** by **values of types** *conforming* to the **error type protocol**
- in swift code, we have a *special construct*, the **do clause**, that **handles** the **behaviour** of *what* we're *trying to achieve*
    * and then we have a *seperate construct*, the **catch clause** that **handles** the **error logic** for us
- *start getting into* the **mentality** of **modelling errors** in your code rather than *leaving them as a problem to solve when your app* **crashes**, and it *will most definitely crash*
- when the **error is throw** from **inside** a **do clause**
    * the **error** is **propogated** to its **outer scope** and is **handled** by a **catch clause**
- the **defer statements** are **executed** in the **reverse order** they are *written*
- to **execute statements** as we **leave** the **current scope**, we use **defer** statement