Handling Failure

Let's Try[T]

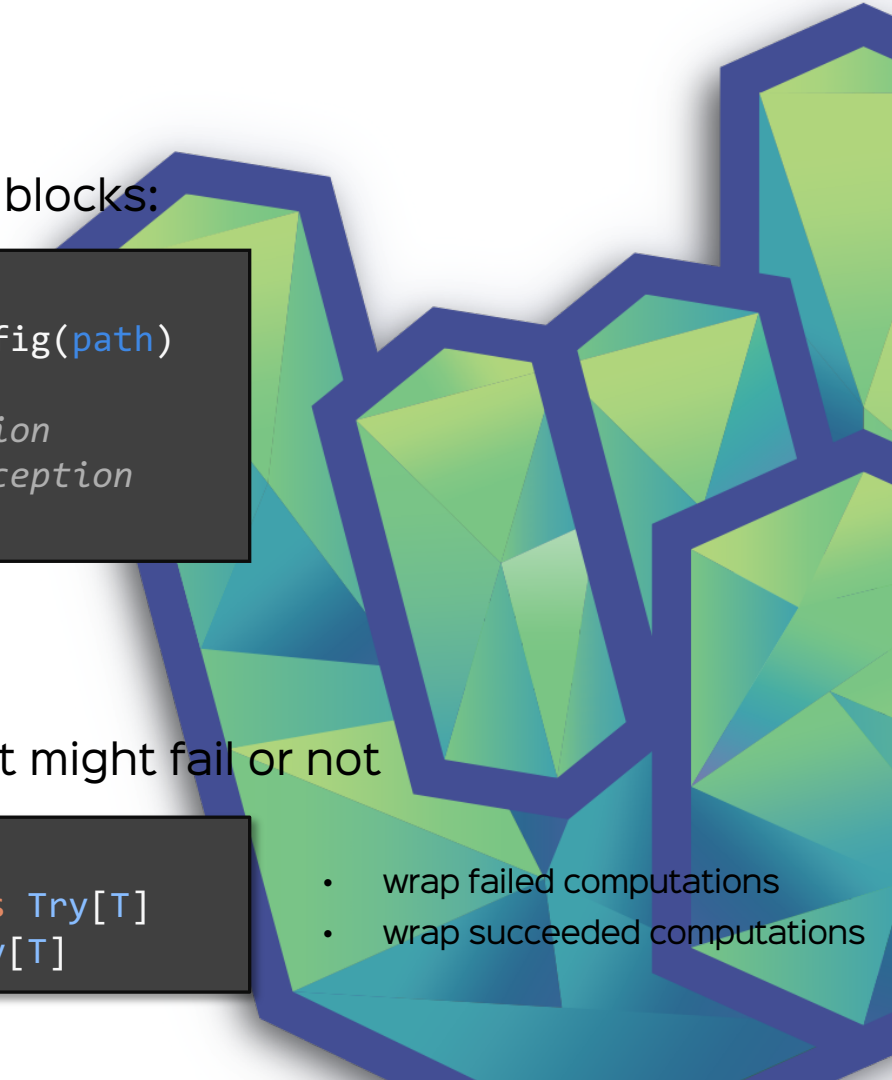
Exceptions are handled inside try-catch blocks:

```
try {
  val config: Map[String, String] = loadConfig(path)
} catch {
  case _: IOException => // handle IOException
  case _: Exception   => // handle other Exception
}
```

- multiple / nested try's make the code hard to follow
- we can't chain multiple operations prone to failure

A Try is a wrapper for a computation that might fail or not

```
sealed abstract class Try[+T]
case class Failure[+T](t: Throwable) extends Try[T]
case class Success[+T](value: T) extends Try[T]
```



- wrap failed computations
- wrap succeeded computations

Takeaways

Use Try to handle exceptions gracefully:

- avoid runtime crashes due to uncaught exceptions
- avoid an endless amount of try-catches

A functional way of dealing with failure

- *map, flatMap, filter*
- *orElse*
- others: *fold, collect, toList,* conversion to Options

If you design a method to return a (some type) but may throw an exception, return a Try[that type] instead.

Scala rocks