

Algebraic Data Types (ADTs)

```
def describeTrafficLight(trafficLight: String) =  
  trafficLight match {  
    case "red"      => "Stop!"  
    case "yellow"   => "Slow down..."  
    case "green"    => "Safe to go."  
    case _          => "Invalid." // shouldn't happen  
  }
```

Gotta test the invalid case!

```
assert(describeTrafficLight("abc") == "Invalid.")
```

```
def nextTrafficLight(trafficLight: String) =  
  trafficLight match {  
    case "red"      => "green"  
    case "yellow"   => "red"  
    case "green"    => "yellow"  
    case _          => "Invalid." // shouldn't happen  
  }
```

Gotta test the invalid case!

```
assert(nextTrafficLight("abc") == "Invalid.")
```

```
def saveTrafficLight(trafficLight: String)  
def logTrafficLight(trafficLight: String)  
def trafficLightAsJson(trafficLight: String)  
def trafficLightAsCsv(trafficLight: String)
```

`String` implies that the complete works of Shakespeare is valid input.

Including the Chinese translations!

Let's solve this with types!

TrafficLight



```
graph TD; TrafficLight --> Red; TrafficLight --> Yellow; TrafficLight --> Green;
```

Red

Yellow

Green

```
sealed trait TrafficLight  
case object Red extends TrafficLight  
case object Yellow extends TrafficLight  
case object Green extends TrafficLight
```

...and nothing else!

(unlike traditional inheritance)

```
def describeTrafficLight(trafficLight: TrafficLight) =  
  trafficLight match {  
    case Red      => "Stop!"  
    case Yellow   => "Slow down..."  
    case Green    => "Safe to go."  
  }
```

VS.

```
def describeTrafficLight(trafficLight: String) =  
  trafficLight match {  
    case "red"      => "Stop!"  
    case "yellow"   => "Slow down..."  
    case "green"    => "Safe to go."  
    case _          => "Invalid." // shouldn't happen  
  }
```

Possible inputs: 3 vs. *Infinity*


```
def nextTrafficLight(trafficLight: TrafficLight) =  
  trafficLight match {  
    case Red    => Green  
    case Yellow => Red  
    case Green  => Yellow  
  }
```

VS.

```
def nextTrafficLight(trafficLight: String) =  
  trafficLight match {  
    case "red"    => "green"  
    case "yellow" => "red"  
    case "green"  => "yellow"  
    case _        => "Invalid." // shouldn't happen  
  }
```

Possible inputs: 3 vs. *Infinity*

New traffic light?

```
sealed trait TrafficLight  
  
case object Red extends TrafficLight  
  
case object Yellow extends TrafficLight  
  
case object Green extends TrafficLight  
  
case object Blue extends TrafficLight
```

⚠ Warning:(170, 5) match may not be exhaustive.
It would fail on the following input: Blue
trafficLight match {



