Enumerations

- Enumerations are a *data type* that store a predefined collection of values.
- Used when you need a custom data type, but don't require the complexity of a struct or class
- Enumerations improve clarity and readability of their code. Also reduce errors.
- Syntax of an enum:

```
enum Directions {
   case North
   case East
   case South
   case West
}
```

- Using an enum

```
// non-optional
let position:Direction = Direction.South

// optional
var position2:Direction? = Direction.East
position2 = nil
```

- Enumerations can be assigned a raw value

```
enum Directions:Int {
   case North = 12
   case East  = 3
   case South = 6
   case West  = 9
}

enum Directions:Character {
   case North = "N"
   case East  = "E"
   case South = "S"
   case West  = "W"
}
```

- After an enumeration is assigned raw values, the values can be accessed using the .rawValue property

```
print(Directions.North.rawValue)
```

- If no raw value is assigned to a String and Int enum, then Swift will automatically assign a raw value to each of the enum's cases

  Int s will start at 0

```
enum Directions:Int {
   case North              // will be assigned 0
   case East               // will be assigned 1
   case South              // will be assigned 2
   case West               // will be assigned 3
}
```

  Strings will be assigned the name of the case

```
enum Directions:String {
   case North              // will be assigned "North"
   case East               // will be assigned "East"
   case South              // will be assigned "South"
   case West               // will be assigned "West"
}
```

- Int enums can be assigned a starting value. Subsequent cases will be the next sequential number.

```
enum Directions:Int {
   case North = 55
   case East               // will be assigned 56
   case South              // will be assigned 57
   case West               // will b assigned 58
}
```