Sphero R2D2
Swift Playground
Command Reference

Roll

```
roll(heading: Int, speed: Int)
```
heading: 0 - 360
speed: 80 - 255

Dependencies:
Requires wait() followed by stopRoll() as subsequent commands

```
wait(for: Double)
stopRoll() // stops roll action
```

Stance

```
setStance(R2D2Stance. [bipod | tripod | waddle | stop ])
```

Dependencies:
requires `wait(for: 3)` as next command after `R2D2Stance.waddle`.

```
wait(for: 3.0)
```

Sound

```
playSound(sound: R2D2Sound. [happy | cautious | excited | hello | joyful
| sad | scan | scared | talking])
```

Dome

```
setDomePosition(angle: Int)
```
angle: -100 - 100

Dependencies:
requires „wait(for: Int)“ command as next command

```
wait(for: Double)
```

Lights

```
setFrontPSILed(color: FrontPSIColor. [black | blue | red])
setBackPSILed(color: BackPSIColor. [black | green | yellow])

setHoloProjectorLed(brightness: Int)
setLogicDisplayLeds(brightness: Int)
```
brigntness: 0 - 255

Wait

```
wait(for: Int)
```
for: 0 - 255

R2D2 Playground exercises
Chapter 1: Rolling

In this lecture you are tasked with driving R2D2 along a course.
To fulfill this task you'll need the following commands, which are described in detail above in the command reference:
- setStance()
- roll()
- wait
- play()

Here are some examples of how to concour the task. From a basic linear approach to a solution that uses a array of structs that describe the path that R2D2 will drive along.

1.1 Basic version

```
func escape() {
    setStance(R2D2Stance.tripod)
    roll(heading: 0, speed:80)
    wait(for: 1.0)
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```

Explanation
This code snippet lets R2D2 drive for one second on a straight line. At the beginning he sets the stance and polls it in after he arrived. Then he plays a happy sound.

Task
Play along with the values of R2D2Stance, heading, speed, for and R2D2Sound in order to understand the basics of what R2D2 is capable of.

1.2 Using an array for patrolling in a square

```
func escape() {
    let angle = [0, 90, 180, 270]
    setStance(R2D2Stance.tripod )
    for a in angle {
        roll(heading: a, speed:80)
        wait(for: 1)
    }
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```

Explanation
This code snippet lets R2D2 drive along a path that is defined by an array which contains the heading values. In a for loop the value for heading is read from the array.

Task
Play along with the angle array. Find other values for the angle array. Add more values to the array.

1.3 Using arrays for angle and speed

Good to know:
Arrays always start at index 0.

```
func escape() {
    let angle = [0, 90, 180, 270]
    let speed = [80, 100, 120, 140]
    setStance(R2D2Stance.tripod )
    for i in 0 ... 3 {
        roll(heading: angle[i], speed: speed[i])
        wait(for: 2.0)
    }
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```

Explanation

This code snippet lets R2D2 drive along a path that is defined by an array which contains the heading values. The corresponding speed is defined in a second array named speed. In a for loop the values for heading and speed are read from the arrays.

Task

Play along with the angle and speed arrays. Find other values for the angle and speed array. Add more values to the arrays.

Quiz

- What happens, if you add more values to the angle array but not to the speed array?
- How could you prevent this? *
- Which other value do you have to change if you add more values to the arrays - why?

*compare angle.count to speed.count and loop to the smaller value

---

1.4 Using a two dimensional array for angle and speed

```
func escape() {
    let angle = 0
    let speed = 1
    var way = [[0,80], [90,100], [180,120], [270,140], [45,255]]
    setStance(R2D2Stance.tripod )
    for waypoint in way {
        roll(heading: waypoint[angle], speed: waypoint[speed])
        wait(for: 2.0)
    }
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```

Explanation

This code snippet lets R2D2 drive along a path that is defined by an 2-dimensional array which contains the heading and speed values. In a for loop the values for heading and speed are read from the array.

Task

Play along with the angle array. Find other values for the angle array. Add more values to the array.

Quiz

- What is the advantage of the 2-dimensional array?*

*Heading and speed are defined in pairs as required.

1.5 Using a function to initialize the two dimensional array

```
func makeWay(numberOfWayPoints: Int) -> [[Int]] {
    let angle = 0
    let speed = 1
    var way = Array(repeating: Array(repeating: 0, count: 2), count: numberOfWayPoints)
    var angleValue = 0
    var speedValue = 80
    for i in 0 ... numberOfWayPoints-1 {
        way[i][angle] = angleValue
        way[i][speed] = speedValue
        angleValue += Int(360/numberOfWayPoints)
        speedValue += Int(175/numberOfWayPoints)
    }
    return way
}

func escape() {
    let angle = 0
    let speed = 1
    var way = makeWay(numberOfWayPoints: 6)

    setStance(R2D2Stance.tripod )
    for waypoint in way {
        roll(heading: waypoint[angle], speed: waypoint[speed])
        wait(for: 1.0)
    }
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```

1.6 Using an array of struct

```
struct leg {
    var angle: Int
    var speed: Int
    var wait: Double
}

func makeWay(numberOfWayPoints: Int) -> [leg] {
    var way = [leg]()
    var angleValue = 0
    var speedValue = 80
    var waitValue = 1.0
    var thisLeg = leg(angle: 0, speed: 0, wait: 0.0)

    for i in 0 ... numberOfWayPoints-1 {
        thisLeg.angle = angleValue
        thisLeg.speed = speedValue
        thisLeg.wait = round((waitValue + (Double(i)/Double(numberOfWayPoints)))*10)/10
        angleValue += Int(360/numberOfWayPoints)
        speedValue += Int(175/numberOfWayPoints)
        way.append(thisLeg)
    }
```

```
    return way
}

func escape() {
    var way = makeWay(numberOfWayPoints: 6)

    setStance(R2D2Stance.tripod )
    for waypoint in way {
        roll(heading: waypoint.angle, speed: waypoint.speed)
        wait(for: waypoint.wait)
    }
    stopRoll()
    play(sound: R2D2Sound.happy)
}
```