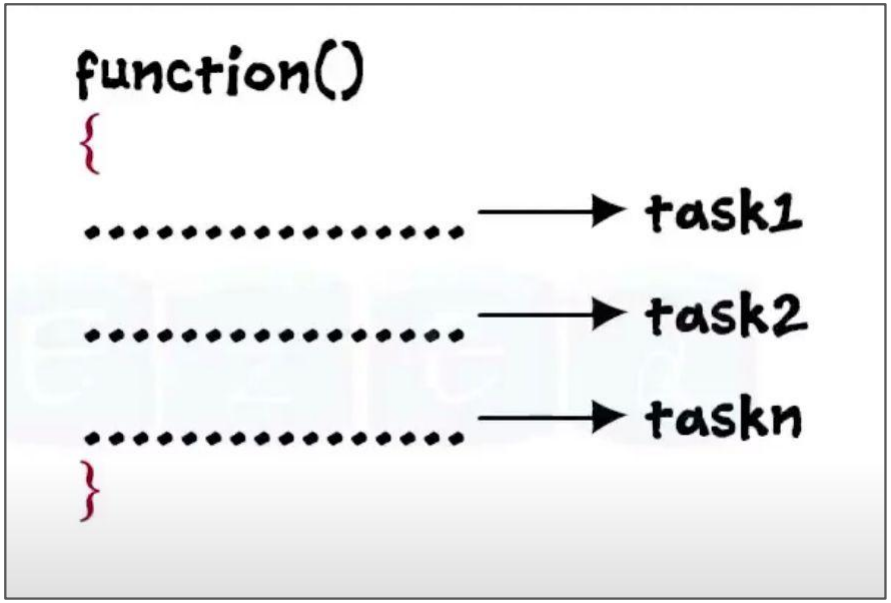
Functions

What is a Function?

A function is set of instructions that are grouped together to achieve a specific outcome.

Examples:

- Function to calculates the sum of two numbers
- Function to outputs a receipt to a screen
- Function to downloads a photo from the internet



Why do we need Functions?

Using functions in a program:

- Makes your program more modular, thus, easier to manage and maintain
- Makes it easier to tackle smaller tasks individually.
- Allows you to reuse logic within your program

Defining and Calling Functions

Using a Function (Methods)

To add a function to your program, you must do two things:

1. Define the function (method) (Function Definition)
2. Use the function (method) in your program (Calling the function)

```
func sayHello() {
    print("Hello, how are you?")
    print("This is the way!")
}

// .. some code
sayHello()
// .. some more code
```

Defining a Function

The function definition consistions

- A **header** that specifies the function's name and function parameters
- A **function body** which is a collection of statements that are performed when the function is called

Function Header

Optional: Function Parameters go here

Function Body

```
func sayHello() {
    System.out.println("Hello, how are you?")
    System.out.println("This is the way!")
}
```

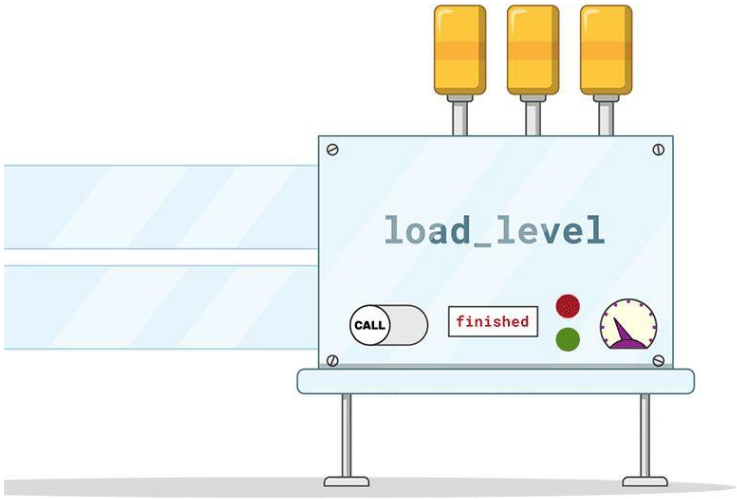
Function Parameters

Functions can accept parameters

Sometimes, you need to send **one or more pieces of data** need to a function

- Parameters are “inputs” that you provide to the function
- The function uses the “inputs” to complete its task.
- Multiple arguments can be passed sequentially into a **parameter list.**

Parameters can be used to customize the behaviour of a function (method)



Passing Arguments to Functions

To pass arguments to a function:

1. Declare a parameter within the function header
2. Use the parameter within the function body

```
func showPokemon(poke:String) {
   print("Your favorite pokemon is "+ poke")
}

func doMath(a:Int, b:Int)  {
   let sum = a + b
   print("The sum of your numbers is "+sum)
   print("The difference of your numbers is "+(a-b))
}
```

Arguments vs. Parameters

The value of the function parameter is set when the function is called. In the example:

- **showPokemon** is a function that accepts a parameter called *poke*
- We call the showPokemon using an argument of ***Charizard***

```
showPokemon(poke:"Charizard")
```

```
func showPokemon(poke:String) {
    print("Your favorite pokemon is \(poke)")
}
```

Function Parameters can Accept Default Values

A function parameter can be set to a default value.

If you do not specify an argument when calling the function, the function will use the provided default value.

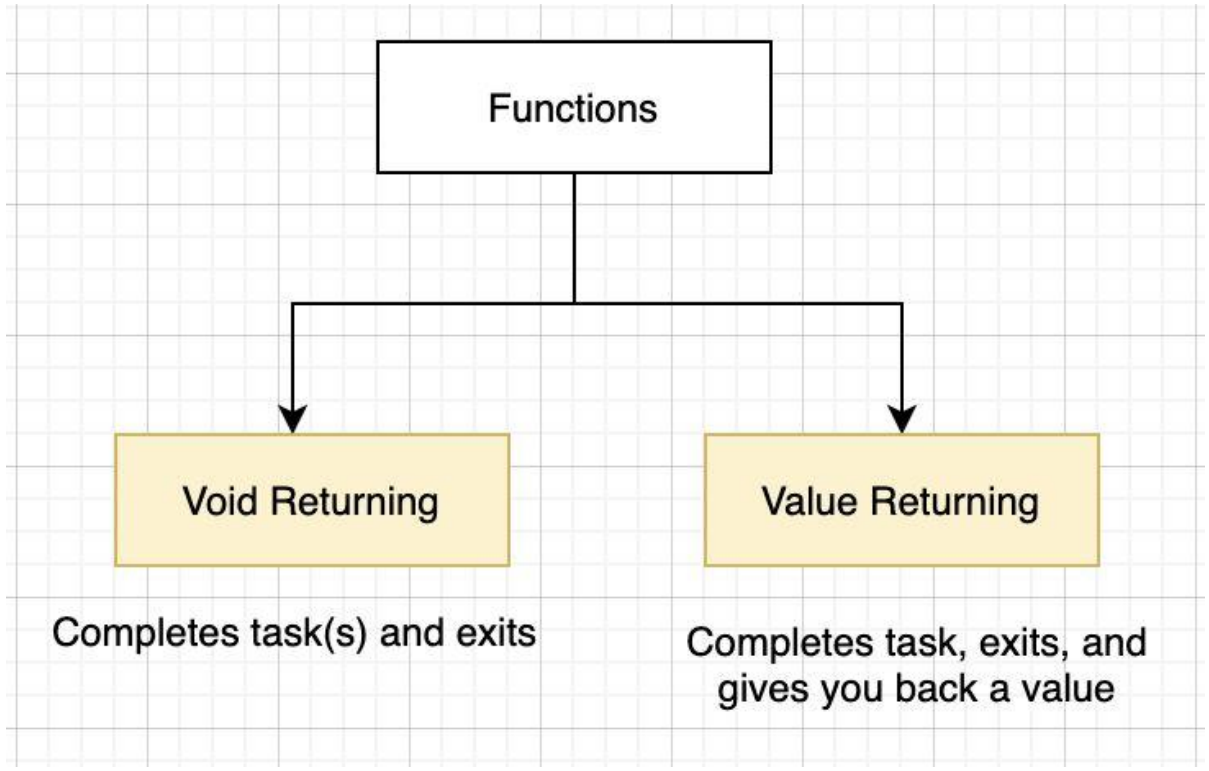
```
func showPokemon(poke:String="Pokemon") {
    print("Your favorite pokemon is \(poke)")
}


// calling the function with no poke argument
showPokemon()
```

Value Returning Functions

Types of Functions

There are two types of functions:

- Value returning functions
- Void functions



Value Returning Function

To create a value returning function:

use the **return** keyword in the function body

In the function header, specify the *type* of data that is being returned

```
func addNumbers()->Int {
    let a = 25;
    let b = 75;
    let total = a + b - 30 * 4;
    return total;
}
```

Summary of Functions

Summary of Functions

In programming, you can have 4 types of functions:

1. Void function
2. Void function that accepts parameters
3. Value returning function
4. Value returning function that accepts parameters

See code examples for syntax.