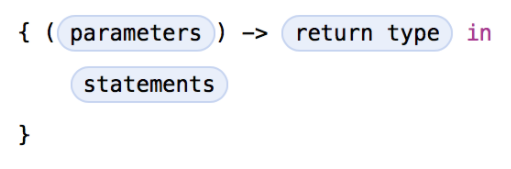
**Swift Closures**

Closures are blocks of functionality that can be passed around our code.

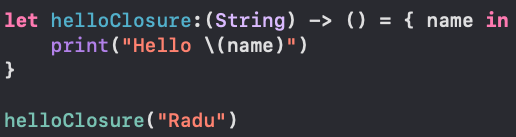
They are of a type that makes it possible to assign them to variables.

Closures can accept parameters and return values. It also contains a set of statements which executes after you call it and can be assigned to a variable/constant as functions.

Closure syntax:

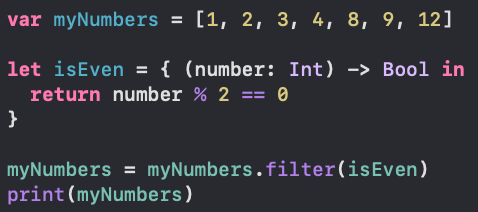


Example:



What do we use closures for? We use closures to pass instructions around in the code, for example: as a parameter in a function.

Example of passing a closure as a parameter:



**Swift Strong Reference**

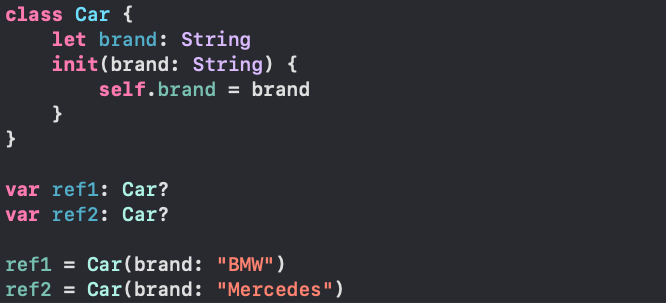
Strong reference is a normal reference to an object.

It protects the referenced object from getting deallocated by ARC, meaning that any object that has a strong reference will not be deallocated.

Every strong reference to an object will increase the retain count. When an object’s retain count is zero, the system is allowed to deallocate it from memory.

By default, all references to an object are strong references.

Example of strong references:



**Swift Weak Reference**

A weak reference is a reference that does not increase the retain count by one, therefore it does not protect an object from getting deallocated by the ARC.

On deallocation, a weak reference will be set to nil.

Why should you use a weak variable, considering that it creates a nil reference?

If object Car has a strong variable reference to object Garage and object Garage has a strong reference to object Car, none of them will ever be deallocated from memory because they keep each other alive. Therefore, a solution for this would be to make one of the references weak: object Garage has a weak reference to object Car. This guarantees that Garage can not be destroyed while the Car still exists, but object Car.

**Swift ARC**

ARC stands for Automatic Reference Counting and is used to track and manage the usage of the application’s memory.