Introducción a swift

Librerías básicas Import Foundation

Opcionalidad

Una variable puede ser opcional, no tiene valor.

```
var optionalString = "Hello"
print(optionalString == nil)//Esto sería un False

var optionalName: String? = "patata"
var greeting = "Hola"
if let name = optionalName{
   greeting = "Hola, \(name)"
}
```

Podemos lidiar con los valores nulos utilizando doble interrogación:

```
   let nickname: String? = nil
   let fulname = "patata"
   //con la doble interrogación indicamos que si nickname es null, se
muestra fulname en su lugar
   let informalGreeting = "Hola \(nickname ?? fulname)"
   print(informalGreeting)
```

Switches

```
let vegetable = "pepper"
switch vegetable {
  case "celery":
    print("Me da que esto no es, pero a saber")
  case "cucumber", "Watercress":
    Print("Esto le da miedo a youtube")
  case ket x Where x.hasSuffix("pepper")://si acaba en pepper se ejecuta
    print("THIS")
  default:
    print("Verdura")
}
```

Programación Funcional

Swift tiene programación funcional. Ejemplos:

```
Long result = numetos.stream().filter(num.patata > 10)//Filtro
```

```
let interestingNumbers = [
  "prime":[2, 3, 5, 7, 11, 13]
  "Fibonacci":[1,1,2,3,5,8,]
  "Square":[1,4,9,16,25]
]

var largest = 0
for(_, numbers) in interestingNumbers{
  for number in numbers{
    if number > largest {
      largest = number
    }
  }
}

print largest
```

Bucles

```
var n = 2
while n < 100{//Bucle Repeat
  n*=2
}

print(n)

var m=2
repeat{//Bucle repeat While
  m*=2
} while m<100
print(M)
```

Funciones

```
func saludo(persona: String, day:String)->String{
 return "Hola \(persona) hoy es \(dia)"
}

saludo(persona: "Pancho", dia: "Martes")
```

Si ponemos un _ en la declaración de una función ya no es necesario indicar el tipo antes de la entrada de esta, también podemos cambiar el nombre de variable poniendo delante como queremos que sea:

```
func saludo(_ persona String, en day:String)->String{// persona con el _ y
```

```
day con en para indicar el día
  return "Hola \(persona) hoy es \(dia)"
}

saludo("Pancho", en: "Martes")
```

También se pueden anidar funciones dentro de funciones, estas funciones anidadas no se pueden usar fuera de la función que esamos implementando:

```
func prueba()->String{
  func anidada(){
    print("Hola, soy una funcion anidada")
  }
  return anidada()
}
```