Opcionais:

* Em Swift pode-se declarar uma variável com o uso de um sinal de interrogação ("?") após o tipo para dizer ao compilador que ela irá aceitar o valor nil além de um valor do tipo especificado.

Laços:

```
* let fator = 2
  var auxiliar = 1

  for i in 0..<10{
      print (auxiliar)
      auxiliar += fator
}
```

String:

* Strings constantes (declaradas com let) são por padrão imutáveis, ou seja, não podemos realizar operações nessas strings que as alterem (porém, perceba que podemos criar novas strings a partir delas, atribuindo seu valor). Strings declaradas com var, por sua vez, são mutáveis e podem ser alteradas em memória diretamente, sem a necessidade de declararmos uma nova String.

Array:

* Quando queremos trabalhar com vetores, que são coleções de dados indexadas por inteiros de 0 à N-1 (onde N é o tamanho da coleção), utilizamos os Arrays.

* Adicionar elementos a um Array, porém como podemos alterar uma posição que não seja a última ou trocar um elemento em um Array? Para isso, utilizamos o índice da seguinte forma "array[índice]" (chamada de indexação do Array) para nos auxiliar. "array[índice]=" pode ser utilizada para alterar um elemento na posição índice, enquanto "array[índice]" pode ser utilizada para buscar o elemento de array que esteja na posição índice.

Dicionário:

* Quando queremos trabalhar com coleções;

* A grande diferença entre um array e um dicionário é que enquanto indexamos Arrays com inteiros, dicionários são indexados com quaisquer objetos, por exemplo, Strings. Em um formato de chave e valor, tanto o tipo das chaves como o tipo dos valores devem ser pré-definidos, já que a tipagem é estática e forte.

```
 var carros = ["Fuxca": 25000, "Goool": 20000, "Oniix": 40000]

 for (chave, valor) in carros {
     print("O carro \(chave) custa \(valor) reais")
 }
```

Enum:

* Os Enums nos ajudam a explicitar tipos importantes para o nosso programa;

```
* enum Bussola {
    case Norte, Sul, Leste, Oeste
  }

  var direcao = Bussola.Norte // inferencia de tipo: Bussola
      print(direcao)

  direcao = .Leste // atribuindo novo valor
          print(direcao)
```

```
 enum Estacao {
     case Outono
     case Inverno
     case Verão
     case Primavera
 }

 var atual = Estacao.Outono
 print(atual)
```

Funções:

* No que diz respeito a retornos, as funções podem ser declaradas de duas maneiras distintas: a primeira quando ela tem algum tipo de retorno, e a segunda quando ela não deve retornar nada (void em outras linguagens). Além disso, as funções podem ou não conter parâmetros internos que serão levados em conta na sua execução.

```
 func funcaoSemParamESemRetorno() {
     print("Nada será retornado")
 }

 func funcaoSemParamEComRetornoInt() -> Int {
     return 0
 }

 func funcaoComParamEComRetornoInt(param: Int) -> Int {
     return param + 1
 }
```

```
 let vetor = [1, 2, 3, 4]

 func duplicador(i: Int) -> Int {
     return i * 2
 }

 let vetorDuplicado = vetor.map(duplicador)

 print (vetorDuplicado)



 func buscarLatitudeLongitude() -> (String, Double, Double) {
     return ("Campinas", -22.002, -25.012)
 }

 let (cidade, lat, lng) = buscarLatitudeLongitude()
 // Agora, temos: cidade = "Campinas, lat = -22.002 e lng = -25.012


 print(cidade)
 print(lat)
 print(lng)
```