



Swift

```
var group = ["Fabricio", "Angelo", "Hemili"]  
print(group[0])
```

História:



Vikram Adve



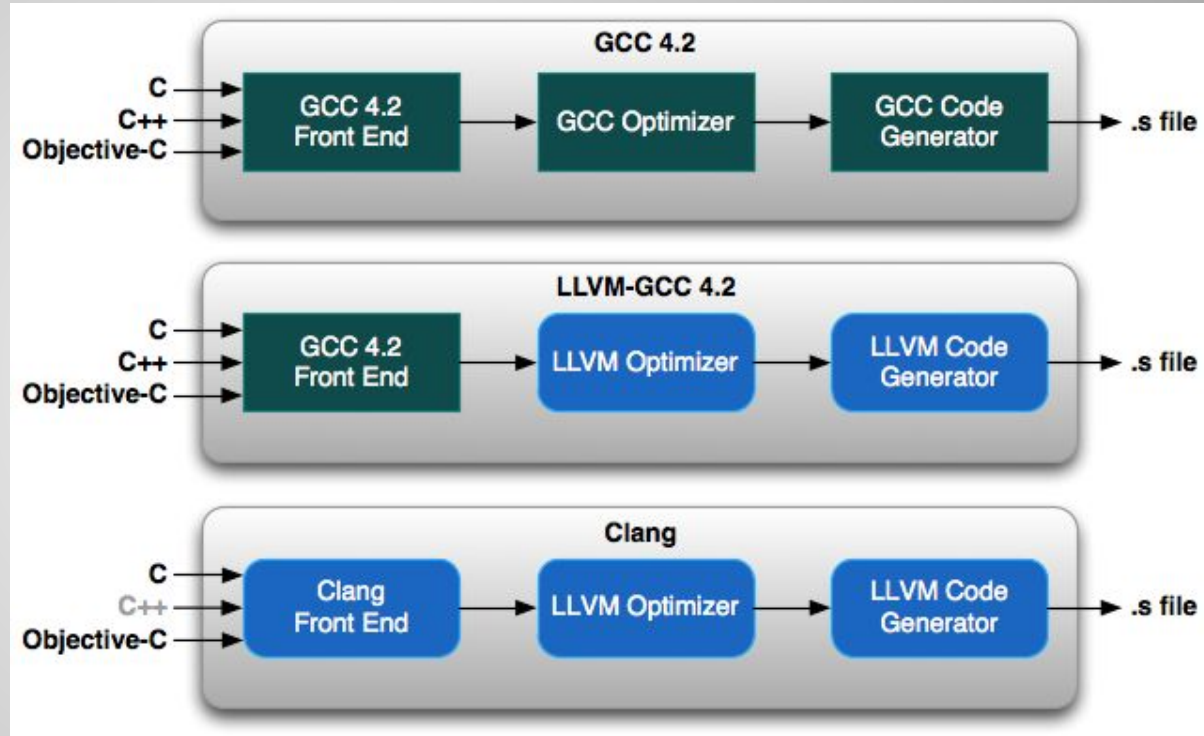
Chris Lattner

O **LLVM** começou como um projeto de pesquisa na Universidade de Illinois , com o objetivo de fornecer uma estratégia moderna de compilação baseada em SSA, capaz de suportar a compilação estática e dinâmica de linguagens de programação arbitrárias.
Final dos anos 2000.

Compilador Clang

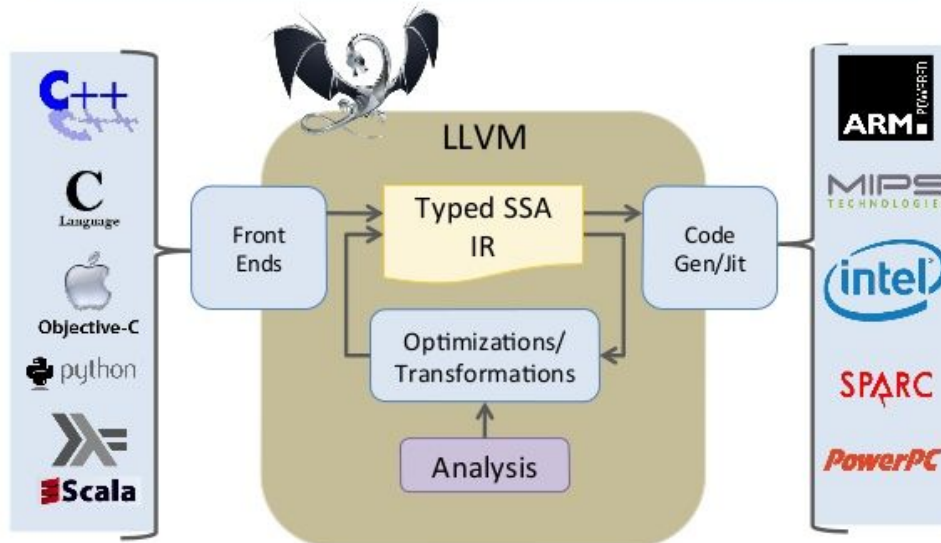
Clang é um front-end de um compilador para as linguagens C, C++, Objective-C e Objective-C++ que utiliza o LLVM como back-end

Em 2005 Lattner, foi contratado pela Apple inc. Iniciou o projeto CLang.



LLVM Compiler Infrastructure

[Lattner et al.]



Infraestruttura LLVM

Swift

Swift é uma linguagem de programação desenvolvida pela Apple para desenvolvimento no iOS, macOS, watchOS, tvOS e Linux. Swift foi desenvolvida para manter compatibilidade com a API Cocoa e com código existente em Objective-C. O compilador usa a infraestrutura do LLVM



<i>Hello World</i> em Objective-C	<i>Hello World</i> em Swift
<pre>#include <stdio.h> int main(int argc, const char * argv[]) { // insert code here... printf("Hello, World!\n"); return 0; }</pre>	<pre>println("Hello World")</pre>

IDE's



XCode



AppCode

VSCode



Atom

Framework Web

3



VAPOR

Pode ser usado para criar APIs RESTful, aplicativos da Web e aplicativos em tempo real usando o WebSockets

```
import Vapor

let app = try Application()
let router = try app.make(Router.self)

router.get("hello") { req in
    return "Hello, world."
}

try app.run()
```

Valores e tipos de dados

- Possui Tipagem Estática.
- Fortemente tipada.
- Possui Inferência de tipo.
- É Case-sensitive.
- Apesar da inferência de tipo, é possível fazer “type annotations”

```
var red, green, blue: Double
```


Swift Playground

< Issuing Commands >

Goal: Use Swift commands to tell Byte to move and collect a gem.


Your character, Byte, loves to collect gems but can't do it alone. In this first puzzle, you'll need to write Swift **commands** to move Byte across the puzzle world to collect a gem.

- 1 Look for the gem in the puzzle world.
- 2 Enter the correct combination of the `moveForward()` and `collectGem()` commands.
- 3 Tap Run My Code.

```
moveForward()  
moveForward()  
moveForward()  
collectGem()
```

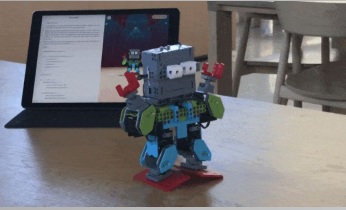
0/1

1x



Stop

Hint



Swift Playgrounds



Princípios básicos da linguagem

```
// Declaracao variaveis
var idade: Int = 24

// Declaracao variaveis constantes
let nome: String = "Lucas"

// Exibir no console
print("\nSou \(nome) tenho \(idade) anos \u{1F604}")
```

fabricio@ubuntu1804: ~

Arquivo Editar Ver Pesquisar Terminal Ajuda

~/Tutorial-Swift on master !1 ?5

> swift basico_linguagem.swift

Sou Lucas tenho 24 anos 😊

```
// if's aninhados

var salario = 3_000.00
var aliquotaImpostoRenda: Float

if salario <= 1_787.77 {
    aliquotaImpostoRenda = 0.0
} else if salario <= 2_679.29 {
    aliquotaImpostoRenda = 7.5
} else if salario <= 3_572.43 {
    aliquotaImpostoRenda = 15.0
} else if salario <= 4_463.81 {
    aliquotaImpostoRenda = 22.5
} else {
    aliquotaImpostoRenda = 27.5
}

print("Alíquota: \(aliquotaImpostoRenda)%")
```

Switch case

```
// Range
let quantidadeFilhos = 4
switch quantidadeFilhos {
case 0...3:
    print("Um carro comum serve")
case 4...8:
    print("Você vai precisar de uma Kombi")
default:
    print("Compre um ônibus")
}
```

```
// continue
let stringEntrada = "texto a ser alterado"
var stringSaida = ""
for caracter in stringEntrada {
    switch caracter {
    case "a", "e", "i", "o", "u", " ":
        continue
    default:
        stringSaida.append(caracter)
    }
}
print("\nRemove vogais e espacos -> \(stringSaida)")
```

fabricio@ubuntu1804: ~

Arquivo Editar Ver Pesquisar Terminal Ajuda

~/Tutorial-Swift on master !1 ?5

> swift basico_linguagem.swift

Remove vogais e espacos -> txtsrldr

For-in

```
// for-in em ranges  
let tabuada = 7  
for multiplicador in 1...10 {  
    print("\(tabuada) x \(multiplicador) = \(tabuada * multiplicador)")  
}
```

fabricio@ubuntu1804: ~

Arquivo Editar Ver Pesquisar Terminal Ajuda

~/Tutorial-Swift on master !1 ?5

at 18:09:

➤ swift basico_linguagem.swift

```
--> 7 x 1 = 7  
--> 7 x 2 = 14  
--> 7 x 3 = 21  
--> 7 x 4 = 28  
--> 7 x 5 = 35  
--> 7 x 6 = 42  
--> 7 x 7 = 49  
--> 7 x 8 = 56  
--> 7 x 9 = 63  
--> 7 x 10 = 70
```

While – funções

- While:

```
while semestre == true {  
    print("socorro")  
}
```

- Repeat-while:

```
repeat {  
    print("socorro")  
} while semestre == true
```

```
func cumprimenta(pessoa: String) -> String {  
    let cumprimento = "Olá, " + pessoa + "!"  
    return cumprimento  
}
```


Array

```
// Declarações

let paises1 : Array<String> = ["Mexico", "Brasil", "Argentina"]

let paises2: [String] = ["Chile", "Colombia", "Canada"]

let paises3 = ["Franca", "Irlanda", "Inglaterra"]

let todos_Paises = paises1 + paises2 + paises3

var array_Vazio : [String] = []

print("\nTodos Paises \n(todos_Paises)")
|

// Acessa primeiro indice
let primeiro_Valor = todos_Paises[0]
print("\nPrimeiro Item do Array \n(primeiro_Valor)")
```

Array

```
// Array

// cria um array
var cores : Array<String> = ["amarelo", "azul"]

// add uma cor ao array com append
cores.append("vermelho")
print("\nAdicionado vermelho", cores)

// add um vetor ao array
cores += ["preto", "branco"]
print("\nAdicionado um vetor ao array ", cores)

// o insert substitui a cor da posição 1 pelo roxo
cores.insert("roxo", at: 1)
print("\nSubstituido uma cor ", cores)
// remove item da posicao 2
cores.removeAt(2)
```


Paradigmas

```
// Imperativo (principal)
let numbers = [1, 2, 3, 4, 5]
var evenNumbers = [Int]()
for i in 0..
```

```
// Funcional (poderoso)
let evenNumbers = [1, 2, 3, 4, 5].filter {
    (number) -> Bool in
        if number % 2 == 0 {
            return true
        } else {
            return false
        }
}
```

TDD Swift

- **XCTAssert**: resultado é verdadeiro (equivalente a `XCTAssertTrue()`);
- **XCTAssertTrue**: resultado é verdadeiro;
- **XCTAssertFalse**: resultado é falso;
- **XCTAssertEqual**: dois valores são iguais;
- **XCTAssertEqualWithAccuracy**: dois valores são iguais de acordo com um fator de precisão;
- **XCTAssertNotEqual**: um valor é diferente de outro;
- **XCTAssertNotEqualWithAccuracy**: dois valores são diferentes de acordo com um fator de precisão;
- **XCTAssertGreaterThan**: um valor é maior que o outro;

```
var idade = 18

// valida se idade for maior que zero
assert(idade >= 0, "Idade não pode ser menor que zero")

print("\nTeste passou ")
```

Map

```
var alunosNotas2 = [4, 5, 7, 9, 6, 10, 3]

// Eleva ao quadrado todos elementos
alunosNotas2 = alunosNotas2.map {$0 * 2}

print("\nNotas com map utilizando metodo sugar $ -> \$(alunosNotas2)")
```

```
// Uma das variações de sintaxes do map, utilizando closures

var alunosNotas1 = [4, 5, 7, 9, 6, 10, 3]

alunosNotas1 = alunosNotas1.map({(nota:Int) -> Int in
|   return nota + 1
})

print("\nNotas com map -> \$(alunosNotas1)")
```

Filter

```
// Verifica se a nota é >= a 5  
  
var alunosNotas2 = [4, 5, 7, 9, 6, 10, 3]  
alunosNotas2 = alunosNotas2.filter { $0 >= 5 }  
  
print("\nAlunos aprovados \n(alunosNotas2) ")
```

Reduce

```
var alunosNotas = [4, 5, 7, 9, 6, 10, 3]
soma = 0
soma = alunosNotas.reduce (0, {$0 + $1})

print("\nReducao com reduce na lista \$(soma)")
```

```
// Reduz toda lista somando com um valor padrao ja definido "10"
let items = [2.0,4.0,5.0,7.0]
let total = items.reduce(10.0, +)
print("\nValor reduzido + com valor padrao defifido --> \$(total)")

// lista de strings
let codes = ["abc","def","ghi"]
let text = codes.reduce("", +)
print("\nUnindo strings da lista --> \$(text)")
```

Closures -> lambdas

```
// Closures
let numeros = [4, 10, 2, 9, 3, 0]

func decrescente(v1: Int, v2: Int) -> Bool {
    return v1 > v2
}

let numerosDecrescentes = numeros.sorted(by: decrescente)
print(">> \(numerosDecrescentes)")
```

```
let numerosOrdenados = numeros.sorted(by: { $0 > $1 } )
print("-> \(numerosOrdenados)")
```

~/Tutorial-Swift on mast

> swift Closures.swift

→ Funcao 🤖
↪ [10, 9, 4, 3, 2, 0]

→ Closures 🤖🤖
↪ [10, 9, 4, 3, 2, 0]

```
{ ( parameters ) -> return type in
    statements
}
```

POO



Struct - Class

```
1 struct Resolution {  
2     var width = 0  
3     var height = 0  
4 }  
5 class VideoMode {  
6     var resolution = Resolution()  
7     var interlaced = false  
8     var frameRate = 0.0  
9     var name: String?  
10 }
```

```
1 let someResolution = Resolution()  
2 let someVideoMode = VideoMode()
```

```
1 print("The width of someResolution is \$(someResolution.width)")  
2 // Prints "The width of someResolution is 0"
```


Class

```
//Criamos um classe com o métodos inicializador
class Empresa{

    var cnpj: String = String()
    var nomeFantasia: String = String()
    var faturamentoAnual: Double = Double()

    init(cnpj: String, nomeFantasia: String, faturamentoAnual: Double){
        self.cnpj = cnpj
        self.nomeFantasia = nomeFantasia
        self.faturamentoAnual = faturamentoAnual

        print("Iniciando a classe Empresa: Nome: \$(self.nomeFantasia)")
    }
}
```

Encapsulamento

public – Permite acesso a qualquer outro elemento.

internal – Permite acesso apenas dentro da própria classe e nas classes herdeiras.

private – Permite acesso apenas dentro da classe na qual foi declarada.

Herança

```
//Classe Pai / SuperClasse (Humano)
class Humano {
    var nome: String = ""
    var idade: Int = 0

    func andar(){
        print("O humano está andando")
    }
}
```

```
//Classe Filha / SubClasse (Filha)
class Atleta : Humano {
    var habilidade: String()

    init(var habilidade: String()){
        self.habilidade = habilidade
    }

    // sobescrita da classe Pai
    override andar(){
        print("O atleta esta andando")
    }
}
```

Herança

```
class Pessoa {  
    private var nome: String?  
    public var telefone: Int  
    internal let endereço = "Vitória"  
  
    init (telefone: Int, nome: String) {  
        self.telefone = telefone  
        self.nome = nome  
    }  
  
    func getName() -> String? {  
        return nome  
    }  
  
    func setName(nome: String) {  
        self.nome = nome  
    }  
  
    func info()-> [String] {  
        var str: [String] = []  
        //Pode ser nula:  
        if let nome = self.nome {  
            str.append(nome)  
        }  
        let telefone = self.telefone  
        str.append(String(telefone))  
        return str  
    }  
}
```

```
class Estudante: Pessoa {  
    let universidade = "UFES"  
  
    init (nome: String, telefone: Int, endereço: String, qtdCafé: String) {  
        super.init(nome:nome, telefone:telefone, endereço:endereço)  
        super.qtdCafé = "Alta!"  
    }  
  
    override func info () -> [String] {  
        var str: [String] = []  
        if let nome = self.nome {  
            str.append(nome)  
        }  
        let telefone = self.telefone  
        str.append(String(telefone))  
        str.append(universidade)  
        return str  
    }  
}
```

Polimorfismo

Generic

```
// Generics
func minhaFuncao<T, U>(a: T, b: U) {}

func imprimeElementos<T>(a: [T]) {
    for elemento in a {
        println(elemento)
    }
}

struct MinhaColecao <T> {
    let itens:[T]

    init(itens: [T]) {
        self.itens= itens
    }
    //...
}
```

Polimorfismo

Sobrecarga

```
// Sobrecarga
struct Vector2D {
    var x = 0.0, y = 0.0
}

extension Vector2D {
    static func + (left: Vector2D, right: Vector2D) -> Vector2D {
        return Vector2D(x: left.x + right.x, y: left.y + right.y)
    }
}
```

Polimorfismo

Sobrescrita

```
// sobrescrita
class Pessoa{
    var nome: String

    init(var nome: String){
        self.nome = nome
    }

    func andar(){
        print("Estou andando")
    }
}
```

```
class Funcionario: Pessoa{
    var salario: Double

    init(var salario: Double){
        self.salario = salario
    }

    override func andar(){
        print("Funcionario continua andando")
    }
}
```

Polimorfismo

Herança Múltiplas -> NÃO

```
protocol A {  
    func aFunctionName()  
}  
  
protocol B {  
    func bFunctionName()  
}  
  
class Some{  
    // ...  
}  
  
class someClass: Some, A, B {  
    //...  
}
```


Extensões

Servem para adicionar funcionalidades de uma forma organizada a classes, enumerados, estruturas ou protocolos.

```
extension Pessoa{  
    |    var saudacao: String {return "Ei " + self.getNome()!}  
    |  
    }  
  
var pessoa = Pessoa (nome:"Nat",telefone:12345678,endereço:"Vitória")  
print([pessoa.saudacao])
```

Protocolos

Os protocolos prometem que uma classe particular implemente um conjunto de métodos.

```
protocol AddStrings{  
  func toString() -> String  
}  
  
extension String:  
  AddStrings{  
    func toString() ->  
      String{  
        return self  
      }  
  }  
}  
  
var aux: AddStrings
```

Protocols

```
protocol Animal {  
    var patas: Int  
}  
  
protocol Pet: Animal {  
    var nome: String  
}  
  
class Cachorro: Pet {  
    //devemos ter o var nome e var patas na classe Dog  
    var nome: String  
    var patas: Int  
  
    init(nome: String, patas: Int) {  
        self.name = nome  
        self.noOfLegs = patas  
    }  
}
```

Atualmente ...



Chris Lattner

```
import TensorFlow
import Python

let np = Python.import("numpy")

let array = np.arange(100).reshape(10, 10) // Create a 10x10 numpy array.
let tensor = Tensor<Float>(numpy: array) // Seamless integration!
```

Swift + TensorFlow

```
let np = Python.import("numpy")
let plt = Python.import("matplotlib.pyplot")
IPythonDisplay.shell.enable_matplotlib("inline")
```

```
let time = np.arange(0, 10, 0.01)
let amplitude = np.exp(-0.1 * time)
let position = amplitude * np.sin(3 * time)
```

```
plt.figure(figsize: [15, 10])
```

```
plt.plot(time, position)
plt.plot(time, amplitude)
plt.plot(time, -amplitude)
```

```
plt.xlabel("time (s)")
plt.ylabel("position (m)")
plt.title("Oscillations")
```

```
plt.show()
```

```
In [1]: %include "EnableIPythonDisplay.swift"
```

```
In [2]: let np = Python.import("numpy")
let plt = Python.import("matplotlib.pyplot")
IPythonDisplay.shell.enable_matplotlib("inline")
```

```
Out[2]: ('inline', 'module://ipykernel.pylab.backend_inline')
```

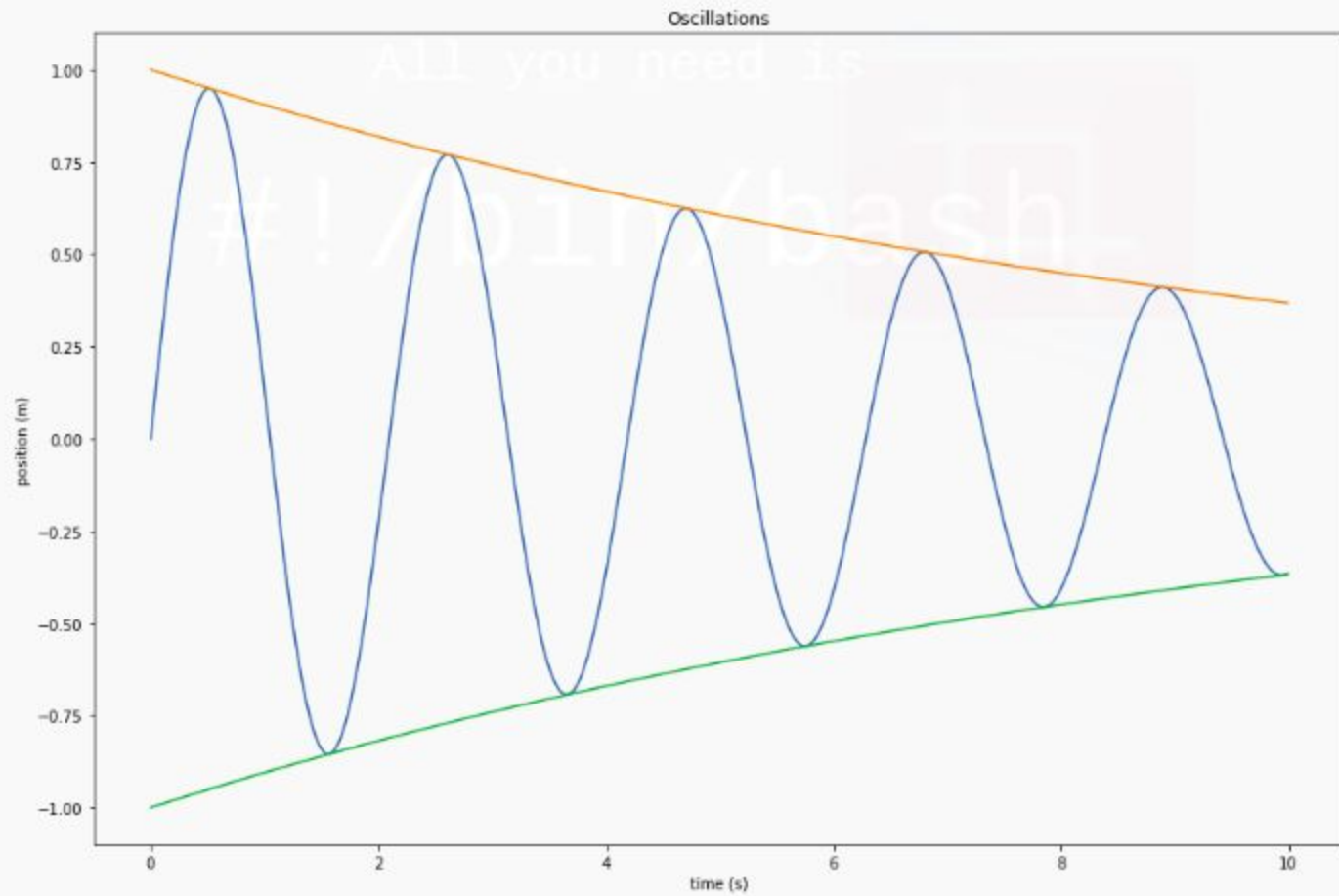
```
In [3]: let time = np.arange(0, 10, 0.01)
let amplitude = np.exp(-0.1 * time)
let position = amplitude * np.sin(3 * time)

plt.figure(figsize: [15, 10])

plt.plot(time, position)
plt.plot(time, amplitude)
plt.plot(time, -amplitude)

plt.xlabel("time (s)")
plt.ylabel("position (m)")
plt.title("Oscillations")

plt.show()
```



Out[3]: None

Links

<https://github.com/fabriciocovalesci/Tutorial-Swift> - Github códigos swift

<https://atp.fm/205-chris-lattner-interview-transcript/> - Entrevista com Chris

<https://ninhodaandorinha.com.br/2017/07/15/testes-unitarios-introducao/> - Testes unitários

<https://www.apple.com/br/swift/playgrounds/> - Swift Playgrounds

<https://www.tensorflow.org/swift> - Swift TensorFlow

<https://api.vapor.codes/> - Doc Vapor