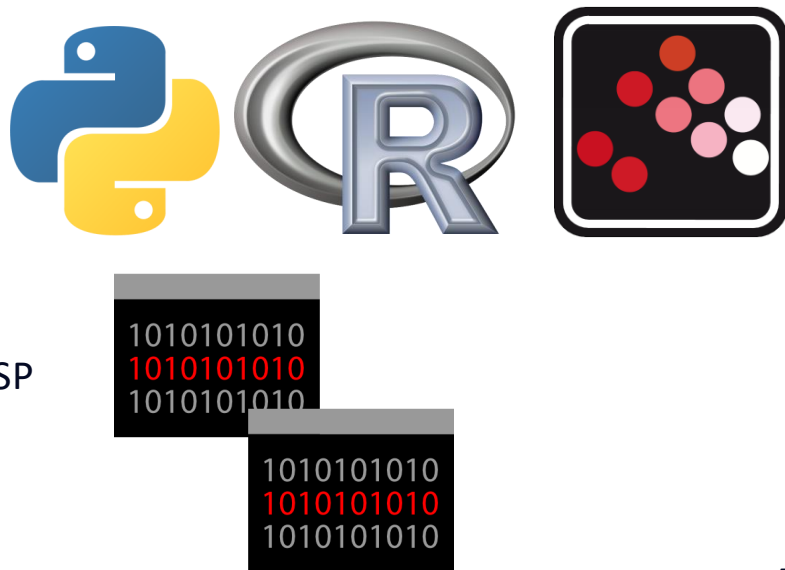


# Linguagens de Programação para **Machine Learning**

**Prof. Dr. Diego Bruno**

Education Tech Lead na DIO

Doutor em Robótica e *Machine Learning* pelo ICMC-USP



# Vamos começar a programar...

Prof. Dr. Diego Bruno

*Machine Learning*

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Linguagens de Programação

Vamos trabalhar inicialmente com as linguagens:



*Python*



*R*



*Scilab*

# Paradigmas de Programação

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# O que são paradigmas de programação?

Um paradigma de programação determina a visão que o programador possui sobre a estruturação e a execução do programa.

Por exemplo, em programação orientada a objetos, os programadores podem abstrair um programa como uma coleção de objetos que interagem entre si.

# Quais os paradigmas?

Os paradigmas destas linguagens são importantes para entendermos melhor nossa forma de pensar sobre nossos problemas de computação:

Lógica;  
Funcional;  
Imperativa;  
Orientada a Objetos.



# Paradigma de Programação Imperativa

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

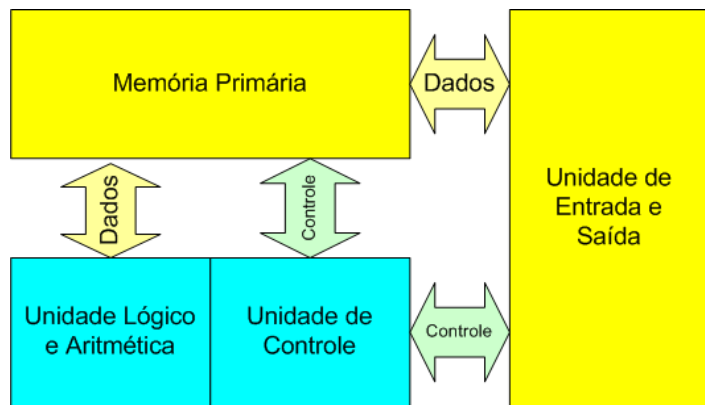
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Programação Imperativa

O paradigma de programação que descreve a computação como ações, enunciados ou comandos que mudam o estado (variáveis) de um programa.



Arquitetura de Von Neumann

Este paradigma foi projetado para a arquitetura de computadores prevalente



**Assembly Language**



# Paradigma de Programação Lógica

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

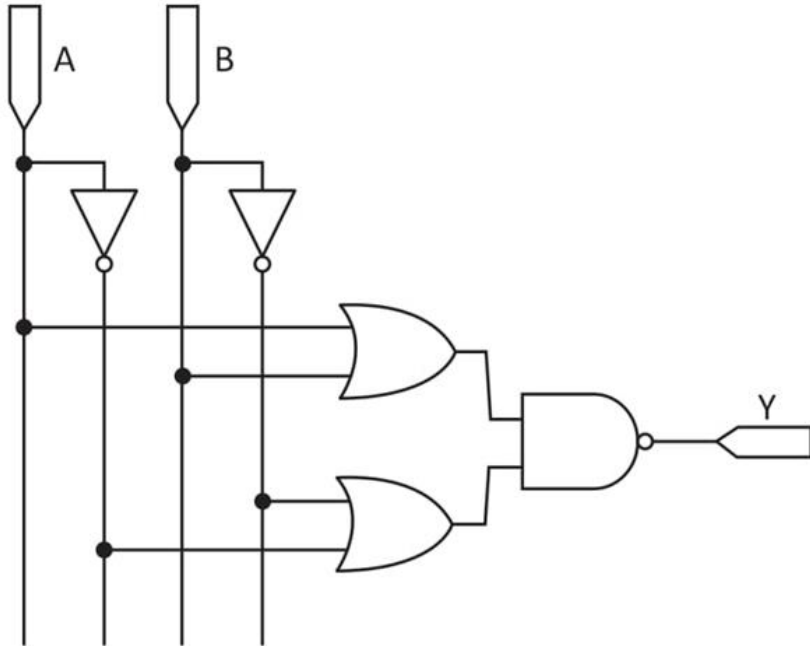
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Programação Lógica

O sentido da **programação lógica** é trazer o estilo da lógica matemática à programação de computadores.



# Programação Lógica

Considere o seguinte banco de dados:

**gosta(maria, flores).**

**gosta(maria, pedro).**

**gosta(paulo, maria).**

Se fizermos a pergunta:

**?- gosta(maria, X).**

estaremos perguntando “Do que Maria gosta?”.

Prolog responde: **X = flores**

# Paradigma de Programação Funcional

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

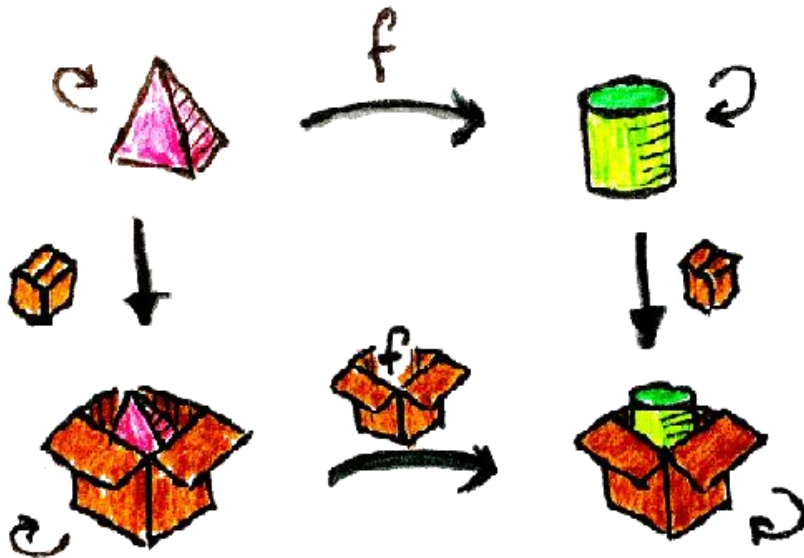
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Programação Funcional

Programação funcional é um paradigma de programação que trata a computação como uma avaliação de funções matemáticas.



$$2+2 \times 3 = ?$$

# Programação Funcional

Programação funcional é um paradigma de programação que trata a computação como uma avaliação de funções matemáticas.

## Scheme

```
1 ((lambda (x) (+ x x)) (* 3 4))
```

Nesse caso, seria isso que aconteceria:

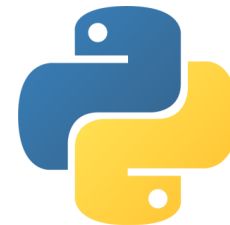
$3 * 4 = 12;$

$x = 12;$

$x + x = 12 + 12 = 24;.$



Linguagem funcional



Suporte para funcional

# Paradigma de Programação Orientada a Objetos

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

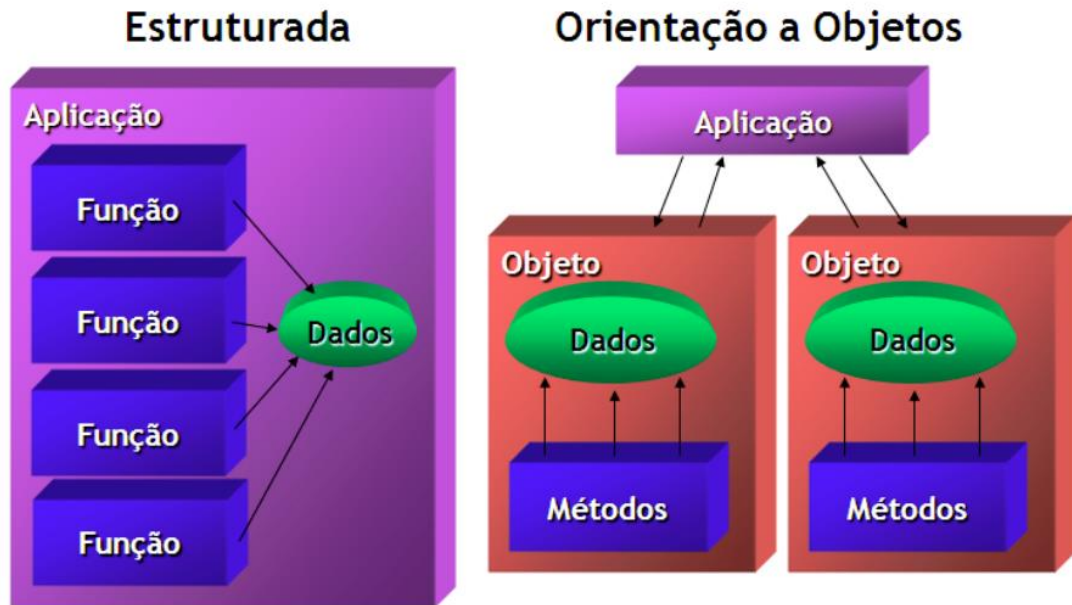
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Programação Orientada a Objetos

Na programação Orientada a Objetos temos como objetivo transformar nosso problema do mundo real em partes para o computador.



Poliformismo

Herança

Encapsulamento

Abstração



# Paradigma de Programação Multi- paradigma

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):

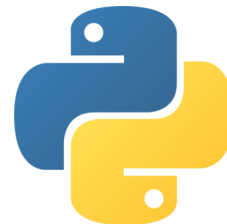
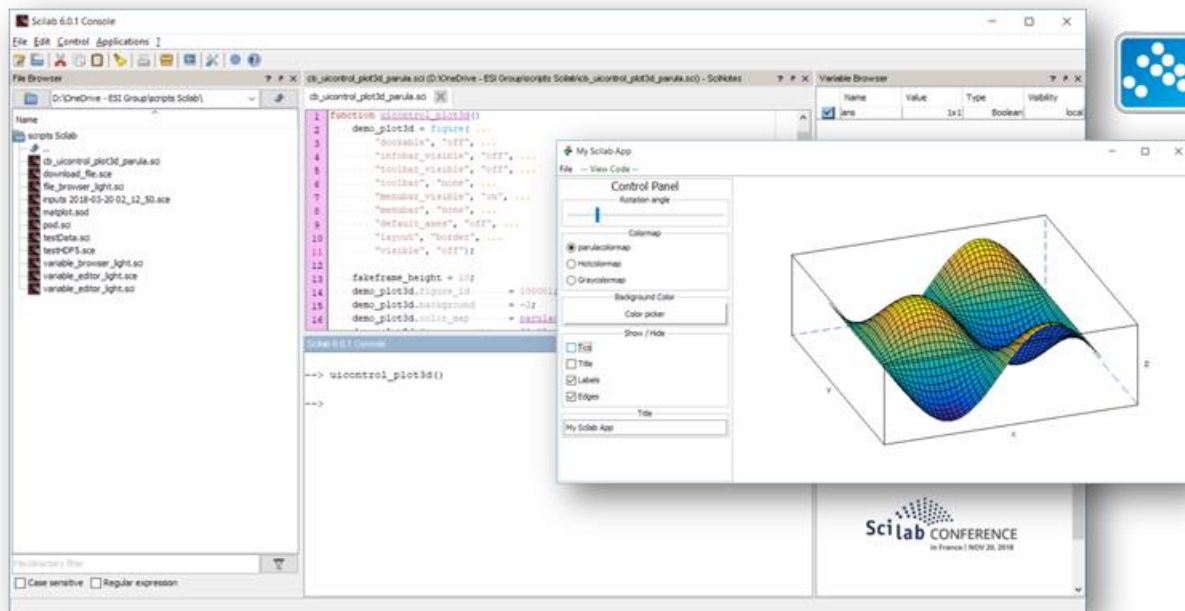
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```

# Programação Orientada a Objetos

*Scilab* (laboratório de matriz) é um ambiente de computação numérica multi-paradigma.



Multi-paradigma

# Obrigado!

Prof. Dr. Diego Bruno

## *Machine Learning*

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense

class MyModel(Model):
    def __init__(self, hidden_units, outputs, **kwargs):
        super(MyModel, self).__init__(**kwargs)
        self.dense = Dense(hidden_units, activation='sigmoid')
        self.linear = LinearMap(hidden_units, outputs)

    def call(self, inputs):
        h = self.dense(inputs)
        return self.linear(h)

my_model = MyModel(64, 12, name='my_custom_model')
```