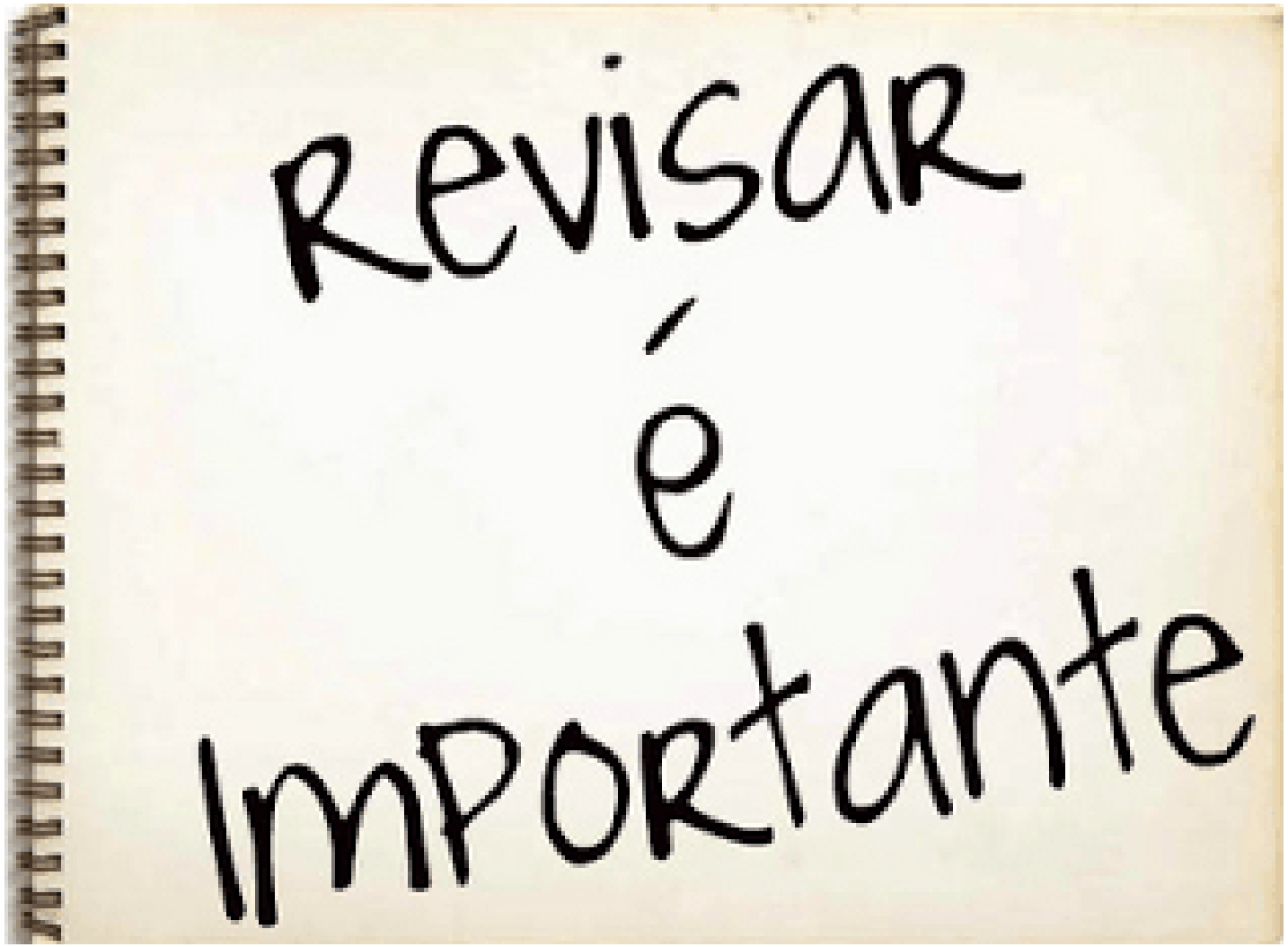
CENTRO UNIVERSITÁRIO
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA – IESB

Aprendizagem de Máquina

PABLO COELHO FERREIRA

Antes de começarmos...

.



pablo.coelho@gmail.com

Geoffrey Hinton e outros

- Em 2006 publicou um artigo (https:/homl.info/136) demonstrando como treinar uma rede neural profunda capaz de reconhecer algarismos escritos a mão com precisão >98%.

- Chamou-se a técnica de **Deep Learning** (aprendizagem profunda).

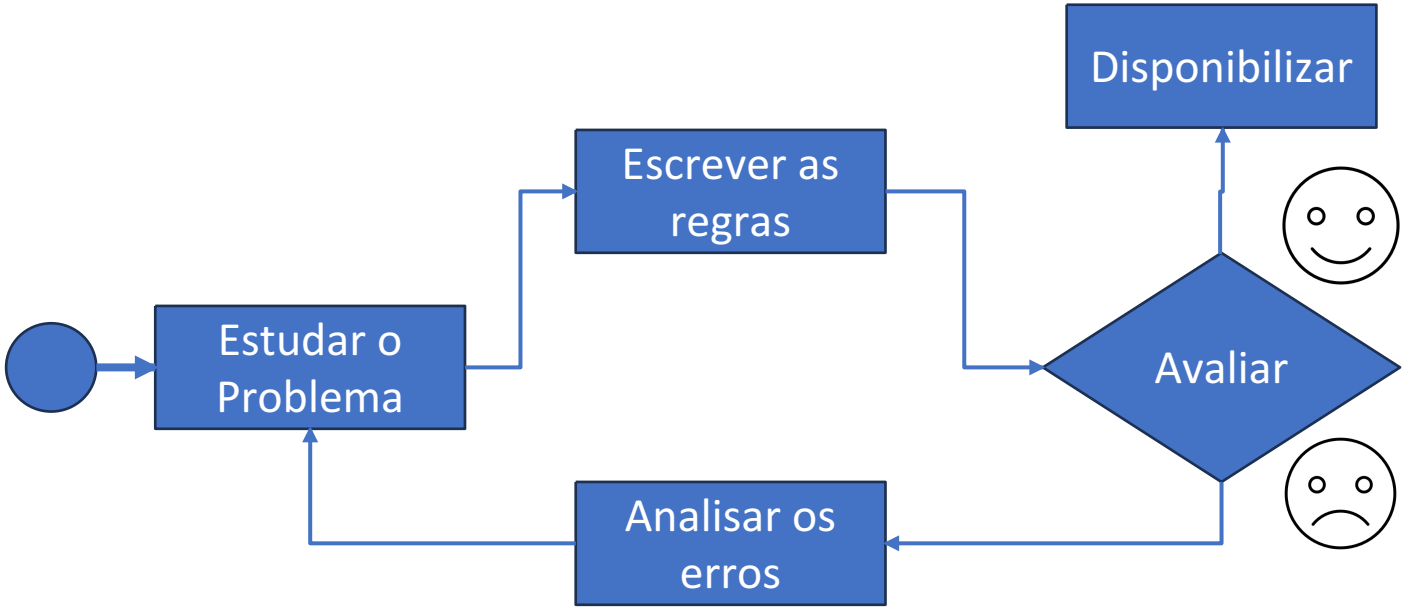
- Uma rede neural profunda é um modelo (bastante) simplificado do nosso córtex cerebral, constituído por pilhas de camadas de neurônios artificiais.

Aprendizagem de Máquina (AM)

- A AM foi introduzida no reconhecimento ótico de caracteres (OCR), mas sua primeira grande aplicação foi o filtro anti-spam na década de 1990.

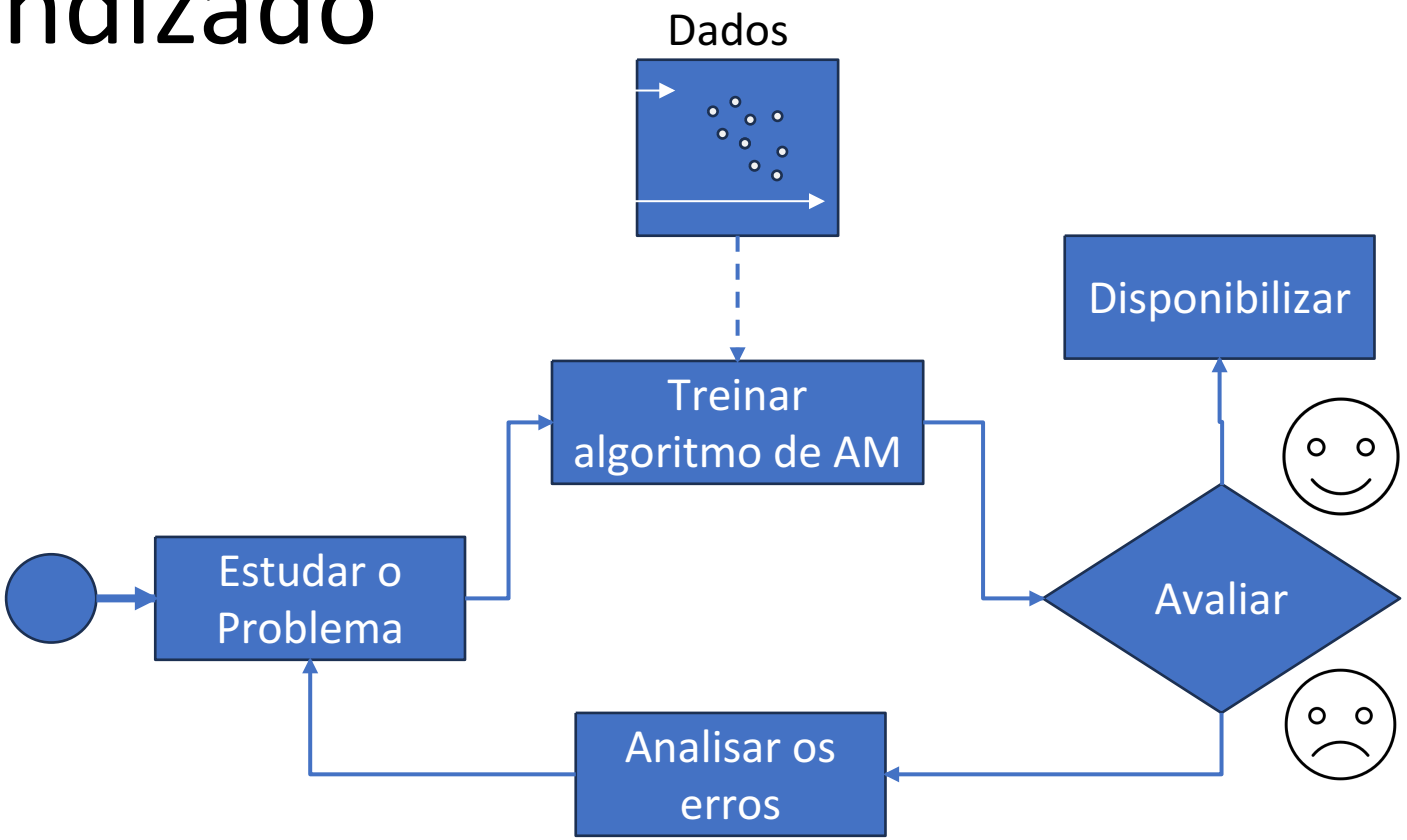
- Mas antes de avançarmos nas aplicações, vamos entender melhor o que é aprendizagem de máquina.

Por que usar o Aprendizado de Máquina
Forma tradicional de escrever algoritmos



pablo.coelho@gmail.com

Aprendizado



pablo.coelho@gmail.com

Alguns conceitos adicionais

- **agente** – É uma entidade que percebe seu ambiente e age em conformidade de acordo com o ambiente.
- **estado** – uma dada configuração do agente e do seu ambiente.
- **estado inicial** – estado como o agente inicia.

Exemplo – Estado inicial

Exemplo de diferentes configurações de um ambiente

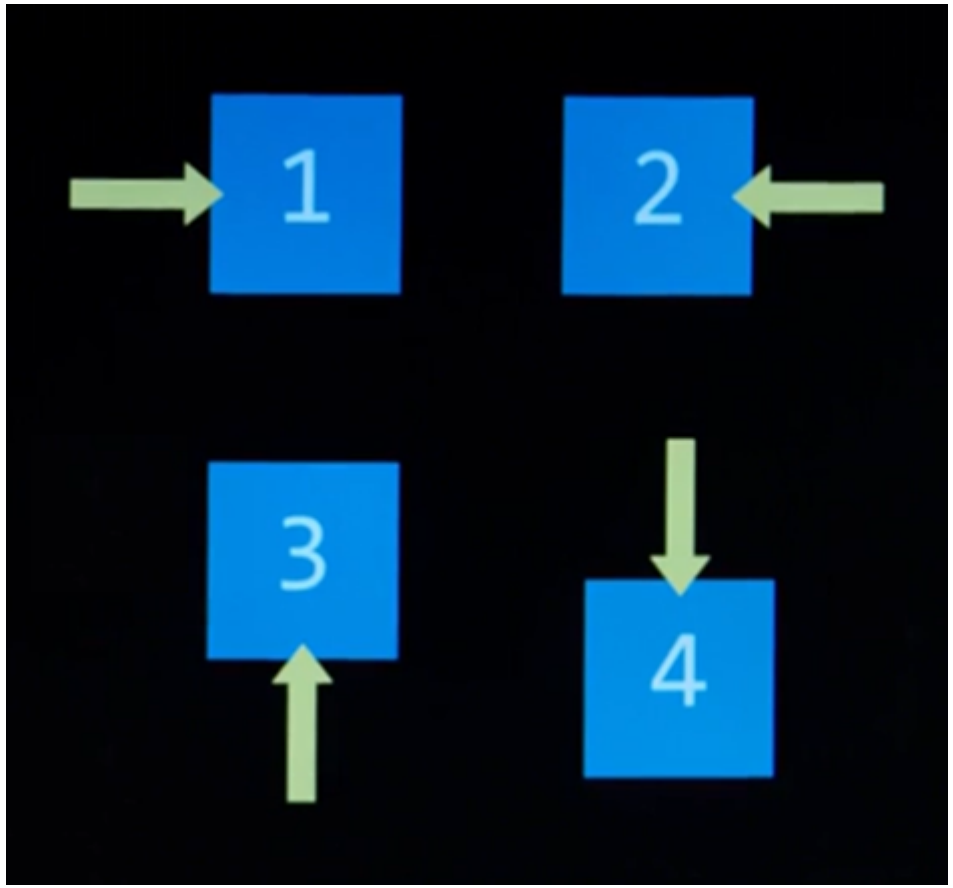
Matriz 4x4 com diversas formas de ordenação.

pablo.coelho@gmail.com

Alguns conceitos adicionais

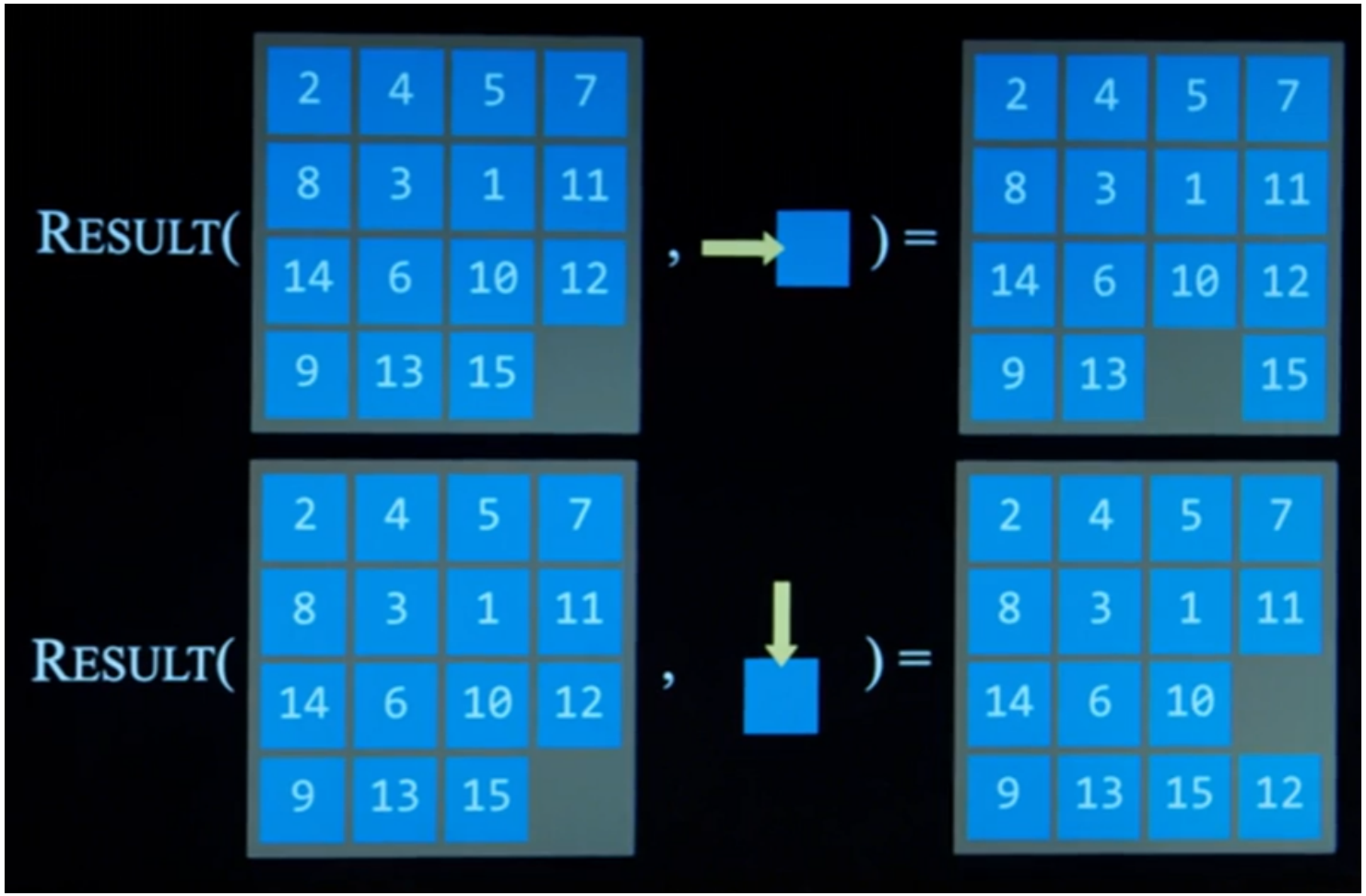
- **ações** – escolhas que podem ser feitas em um estado.
  - É uma função **Actions(s)** que recebe uma entrada (s - estado) e retorna todas as ações que podem ser feitas naquele estado.
- **modelo de transição** – descrição do estado resultante sobre a aplicação de qualquer ação possível sobre qualquer estado.
  - É uma função **Result(s,a)** que retorna o estado resultado da ação "a" no estado "s".

Exemplos de ações possíveis



pablo.coelho@gmail.com

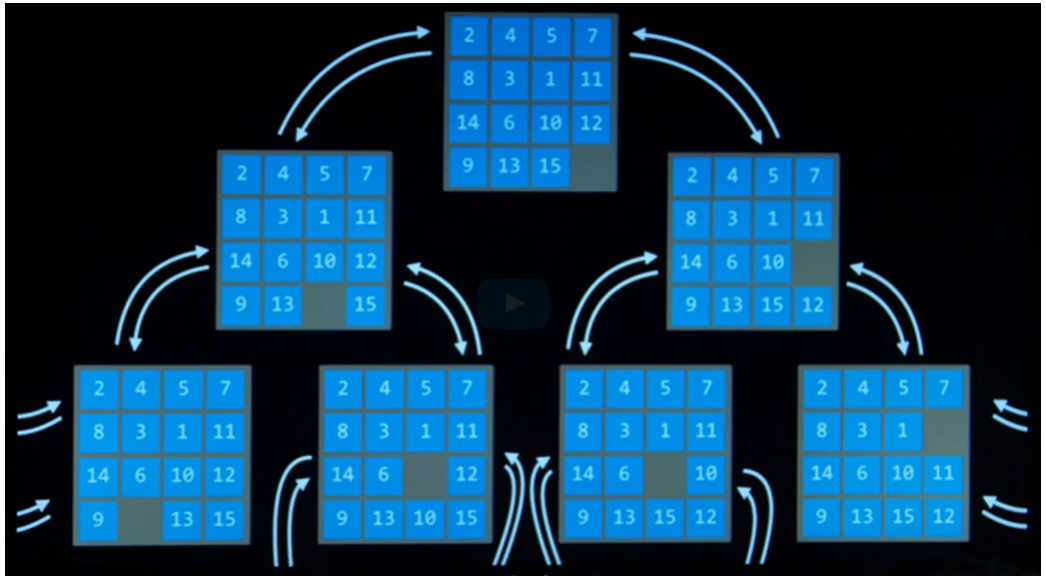
Exemplo de modelo de transição



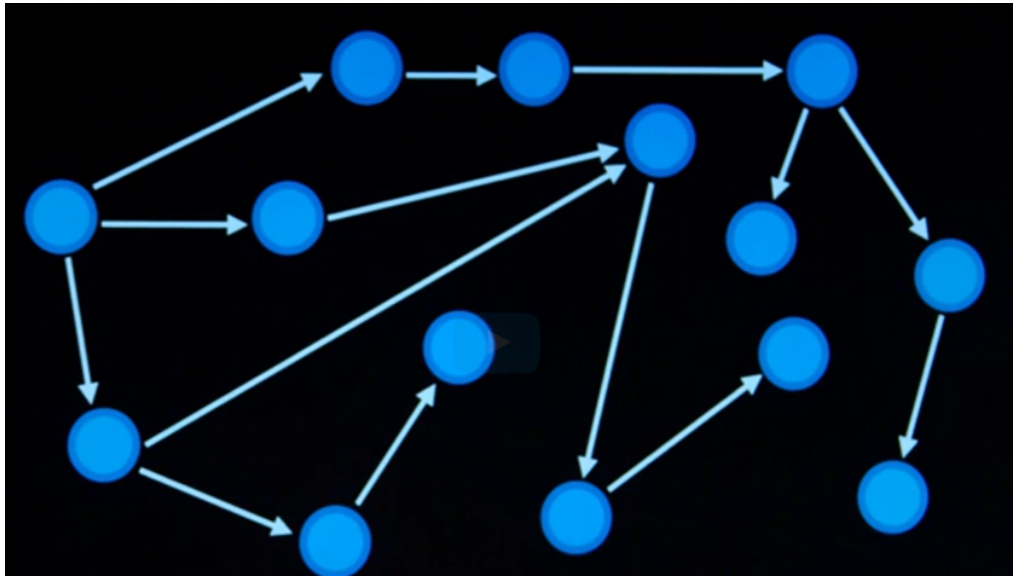
pablo.coelho@gmail.com

Alguns conceitos adicionais

- **Espaço de estados** – escolhas que podem ser feitas em um estado.
  - É uma função **Actions(s)** que recebe uma entrada (s - estado) e retorna todas as ações que podem ser feitas naquele estado.



Pode ser representada por um grafo



pablo.coelho@gmail.com

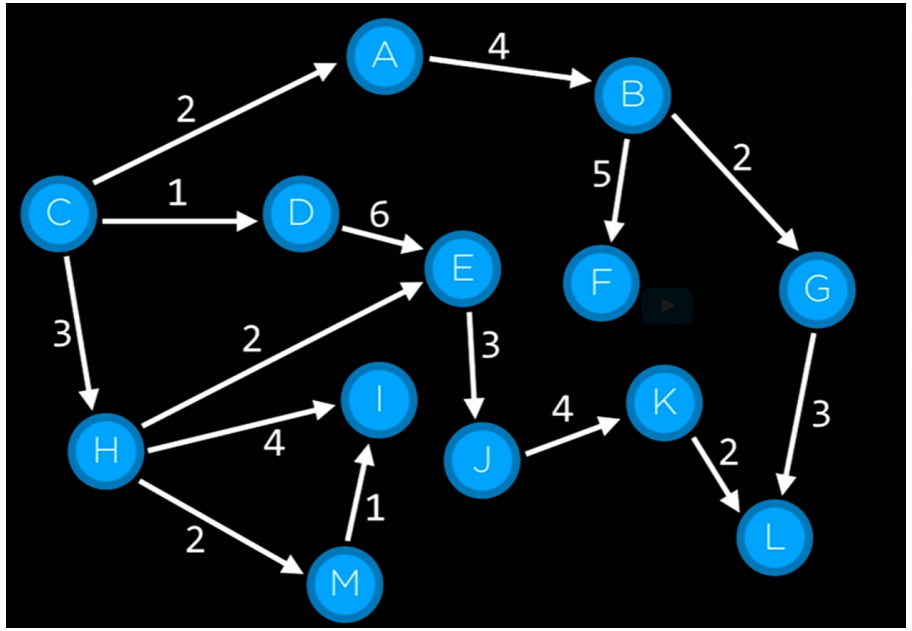
Alguns conceitos adicionais

- **objetivo da tarefa** – Objetivo desejado. Pode ser único ou ter diversas soluções possíveis.
  - Realiza-se o **Teste do Objetivo** para verificar se o, ou um, resultado esperado foi atingido após uma ação.

- **solução** – conjunto de ações que a partir do estado inicial leva até o objetivo da tarefa.
  - Solução ótima – solução de menor custo para sua implementação.

Alguns conceitos adicionais

- **custo da solução** – custo numérico associado ao custo de uma determinada solução.



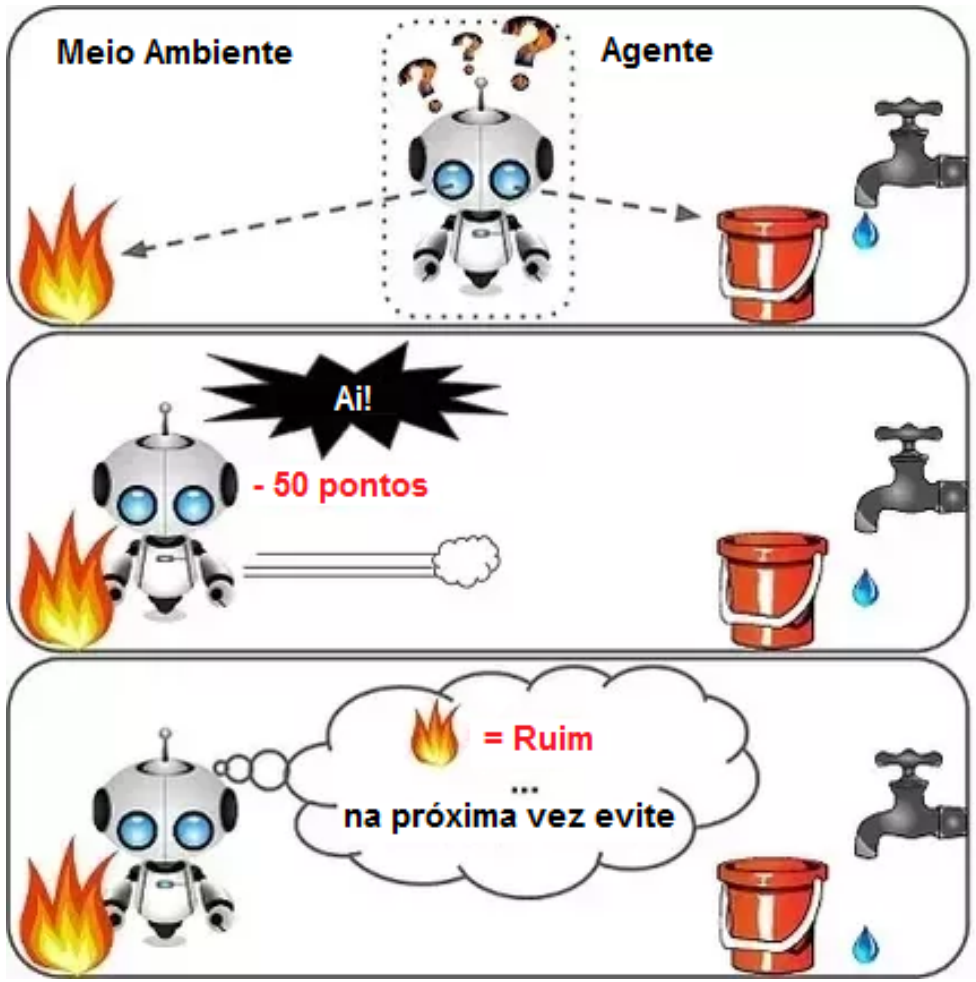
Alguns conceitos adicionais

- **nó** – estrutura de dados que mantêm o registro do:
  - estado do nó
  - pai - nó que o gerou
  - ação - ação aplicada ao pai que causou a origem do nó
  - Custo do caminho – custo desde a origem até o nó

Resumindo...

- Um problema para IA tem:
  - estado inicial
  - ações que modificam os nós
  - modelos de transição
  - teste de objetivo
  - solução
  - custo do caminho para solução

Aprendizado por esforço



1. Observar
2. Selecionar ação usando política
3. Ação!
4. Obter recompensa ou penalidade
5. Política de atualização (etapa de aprendizado)
6. Repetir até que uma política ótima seja encontrada

pablo.coelho@gmail.com

Dever de casa

Criar um notebook no Colab.

Copiar o código base para o seu notebook.

Criar um arquivo CSV com 5 entradas com os seguintes campos:

Nome, data de nascimento (mês, dia, ano), dia do cadastro (ano, mês e dia) e hora de cadastro.

Ler o arquivo.

Imprimir o registro N que o usuário informar via prompt, em uma única linha, concatenada e com a data no padrão brasileiro.

pablo.coelho@gmail.com

Dever de casa

- Exemplo
  - Arquivo CSV

  nome, dataNasc,dataCadastrado,horaCadastro

  Pablo Coelho, 01/26/1974, 2025/02/17, 20:30

  Mariana Ferreira, 26/04/1995, 2025/02/17, 20:31

  (...)

  Saída (caso o usuário indique o registro 1):

  Registro 1: Nome: Pablo Coelho; Data de Nascimento: 26/01/1974; Data de Cadastro: 17/02/2025 as 20:30 horas

Aula 02

pablo.coelho@gmail.com

Python, NumPy, Pandas e Matplotlib

- numpy.org
- pandas.pydata.org
- matplotlib.org

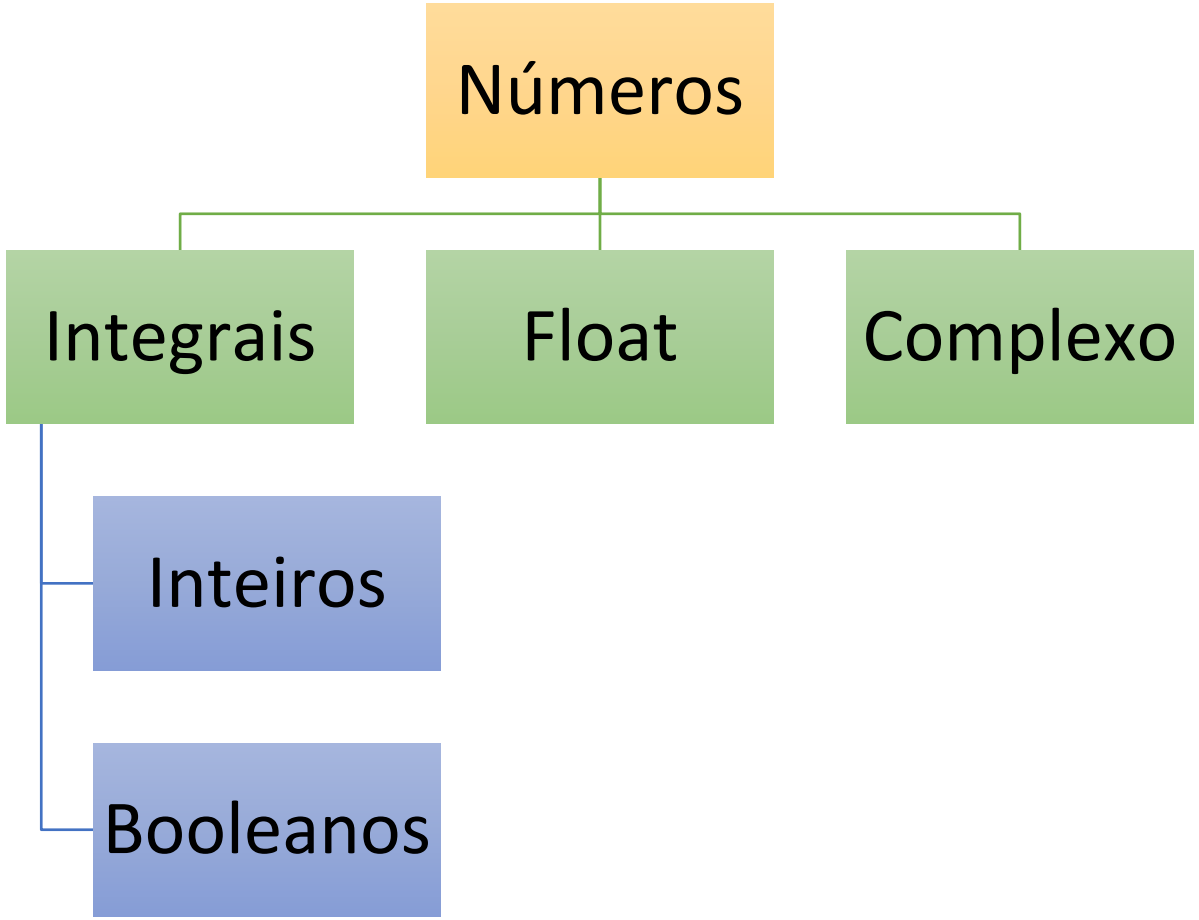
- HOJE PYTHON (1)

pablo.coelho@gmail.com

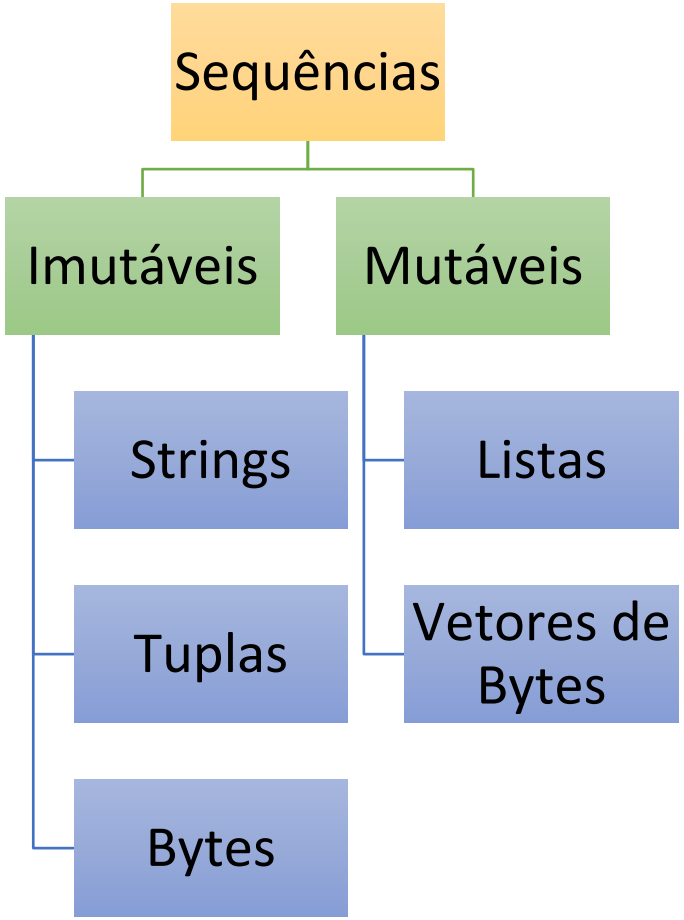
Python

- Linguagem de alto nível (mais próxima a linguagem falada)
- Interpretada
- Tipagem dinâmica
- Criada em 1991 por Guido van Rossum
- Hoje é mantida pela comunidade

Variáveis numéricas



Sequências



pablo.coelho@gmail.com

Resumo

Nome	Tipo	Descrição
Inteiro	int	Todos os números inteiros: 2 300 100000
Ponto Flututante	float	Todos números com decimais: 2.102 403.98
Strings	str	Sequencias de caracteres: “oi” “Pablo” “Janeiro”
Listas	list	Sequencia ordenada de objetos:
Dicionários	dict	Sequencia de pares sem ordem
Tuplas	tup	Sequencia fixa de objetos
Conjuntos	set	Coleção de objetos sem ordem
Booleanos	bool	Verdadeiro ou falso

Operações com Strings

- Pulando linha: \n
  - print (“Hello \n World”)

- SLICE
  - [start:stop:step], traduzindo [começo:fim:passos]
  - START = Início do slice
  - STOP = é até onde vai o slice (mas não inclui o próprio. Para um elemento antes)
  - STEP = é o tamanho do passo que você vai dar (um pulo)

Aspas simples ou duplas

- Usar " ou ' ?
  - São similares. O uso de um, permite utilizar o outro dentro da String.
  - Exemplo:
    - print ("Hello 'world' !") #vai imprimir: Hello 'world' !
    - print ('Hello "world" !') #vai imprimir: Hello"world" !
  - Outra forma de dar o mesmo resultado:
    - print ('Hello \'world\' !')
    - print ("Hello \"world\" !")

Aspas simples ou duplas

•Para strings que se estendem por várias linhas, você pode usar três aspas simples (''' ''') ou três aspas duplas (""" """). Ambas as opções funcionam de forma similar para strings multilinha.

```
texto = '''Este é um exemplo
de string em múltiplas
linhas.'''
```

Acessando caracteres em uma String

- strMinhaString = 'Pablo é professor de ADS'
- print (strMinhaString[0]) #Vai imprimir: P
- print (strMinhaString[1:5]) #Vai imprimir: ablo
- print (strMinhaString[0:5] + '\n' + strMinhaString[6:17]) #Vai imprimir:

Pablo

é Professor

(em linhas separadas)

Acessando caracteres em uma String

- strMinhaString = 'Pablo é professor de ADS'

- print (strMinhaString[5:]) #Vai imprimir: é professor de ADS
- print (strMinhaString[:5]) #Vai imprimir: Pablo
- print (strMinhaString[-6]) #Vai imprimir: d
- print (strMinhaString[::]) #Vai imprimir: Pablo é professor de ADS
- print (strMinhaString[::2]) #Vai imprimir: Pboépoesrd D

Acessando caracteres numa string

- strMinhaString = '123456789'

- print (strMinhaString[1:7:2]) #Vai imprimir: 246
- print (strMinhaString[::-1]) #Vai imprimir: 987654321

Manipulando caracteres numa string

- strMinhaString = 'Pablo é professor de ADS'
- strMinhaString[0] = 'C' #Vai dar erro!
- srtMinhaNovaString = strMinhaString[0] + 'aulo'
- print (srtMinhaNovaString) #vai Imprimir Paulo

Manipulando strings

- strMinhaString = 'Pablo'
- strMinhaString = strMinhaString + 'é professor'
- print (strMinhaString) #vai Imprimir Pablo é Professor

Manipulando strings

- strMinhaString = 'z'
- strMinhaString = strMinhaString * 10
- print (strMinhaString) #vai Imprimir zzzzzzzzzz

Manipulando strings

- print (2+3) #vai Imprimir 5
- print ('2'+'3’) #vai Imprimir 23

Manipulando Strings

- strMinhaString = 'Pablo é professor de ADS'
- strMinhaString = strMinhaString.upper()
- print (strMinhaString) #vai Imprimir PABLO É PROFESSOR DE ADS
- O método lower deixa todas as letras em minúsculas.

Montando uma lista a partir de uma string

- strMinhaString = 'Pablo é professor de ADS'
- strMinhaString = strMinhaString.split()
- print (strMinhaString) #vai imprimir ['Pablo', 'é', 'professor', 'de', 'ADS']
- O resultado é uma lista (vetor) com os itens separados pelo caracter dentro dos parêntesis do split. Se não tiver nenhum utilizará o espaço.

Outro exemplo de SPLIT

- strMinhaString = 'calabreza, portuguesa, marguerita, frango catupiri, bacon com frango, pepperoni'

- strMinhaString = strMinhaString.split(',')

- print (strMinhaString)

- ['calabreza', ' portuguesa', ' marguerita', ' frango catupiri', ' bacon com frango', ' pepperoni']

Exercícios de sala

1) Repita todos os exemplos.

   Altere, quando possível, as variáveis de cada exemplo, acrescentando 1 E subtraindo 1. Antes de executar o código pense na saída. Execute, veja se bateu com o que vc pensou.

   ANOTE o score de erros e acertos.

1) Crie um programa que:
   A) Carregue na variável: strMeuNome, o seu nome completo.
   B) Imprima apenas seu primeiro nome.
      A) Usando SLICE
      B) Usando SPLIT

pablo.coelho@gmail.com

f-strings

- Insere variáveis em uma string

- Sintaxe: (f'TEXTO QLQ {<nome variável} ')

- Exemplo:

- strMinhaString = 'Pablo'
- print (f'Meu nome é: {strMinhaString}.') #vai Imprimir: Pablo

Inserindo variáveis no texto

- strVariavel1 = 'Pablo'
- strVariavel2 = 'Janeiro'
- strVariavel3 = 'Aquário'

- print (f'Meu nome é: %s. Nasci em %s e sou do signo de %s.' %(strVariavel1, strVariavel2, strVariavel3))

Parametros do f-strings

- %s – Substitui por uma string.

- %r – Representação de uma variável. Trás a estrutura dessa variável, no caso de uma string o valor vem entre aspas.

- \t – Representa uma tabulação (TAB)

- %x.yf – Formatação de ponto flutuante, onde X indica quantos caracteres antes do ponto (incluindo) e Y quantas casas decimais.

Exercício de sala

- Testar as saídas:

- print ('O nome é %s.'%'Pablo')
- print ('O nome é %r.'%'Pablo')

- print ('O nome é %s.'%'Pablo \t Coelho')
- print ('O nome é %r.'%'Pablo \t Coelho')

Exercício (continuação)

- print ('O Pablo tem R$ %s na carteira.' %3.75)
- print ('O Pablo tem R$ %d na carteira.' %3.75)

- print ('Ponto flutuante: %5.2f' %(13.144))
- print ('Ponto flutuante: %1.0f' %(13.144))
- print ('Ponto flutuante: %1.5f' %(13.144))
- print ('Ponto flutuante: %10.2f' %(13.144))
- print ('Ponto flutuante: %25.2f' %(13.144))

Tuples

- São idênticas as listas (vetores) mas seus valores são constantes (imutáveis).

- Uma tupla é feita com parêntesis ao invés de colchete.

- Dois métodos: count e index

Exercício de sala

- Crie uma Tuple com as cinco primeiras letras do alfabeto, em minúsculas, em ordem, mas repetindo a letra ‘a’ três vezes.

- Imprima o tamanho da Tuple.

- Imprima quantas letras 'a' existem na Tuple.

- Imprima a posição da letra 'c'.

- myTuple = ('a', 'a', 'a', 'b','c', 'd', 'e’)
- print (len(myTuple))
- print (myTuple.count('a'))
- print (myTuple.index('c'))

Momento concurso

Considerando que, em um projeto desenvolvido em Python, há uma lista inserida como num_lista = [3, 4, 8, 5, -2], julgue o item seguinte, assumindo que, nos comandos em análise, o sinal >>> indica cada linha de comando, e que a tecla Enter foi pressionada após cada linha de comando.

O valor da expressão $3^4 + 8x5^{-2}$ pode ser encontrado executando a linha de comando a seguir.

```
>>> 3 ** 4 + 8 * 5 ** num_lista[4]
```

- ◯ Certo
- ◯ Errado

Ano: 2025 **Banca:** CESPE / CEBRASPE **Órgão:** TRF - 6ª REGIÃO **Prova:** CESPE / CEBRASPE - 2025 - TRF - 6ª REGIÃO - Analista Judiciário – Área: Apoio Especializado – Especialidade: Estatística

pablo.coelho@gmail.com

“Causos” – podem ou não ter acontecido

- Falamos da próxima onda. IA, Robótica e biotecnologia.
- Falemos de como estar a frente. Empreender.
- 1º passo – formalizar-se como empresário
  - Abrir um CNPJ (programação NÃO pode ser MEI – Microempreendedor individual)
- Na próxima aula falaremos de oportunidades para empresas recém abertas.

pablo.coelho@gmail.com

Dever de casa

Desenvolver um programa em Python que receba uma frase inserida pelo usuário e realize diversas operações de análise e formatação, demonstrando os conceitos vistos em aula.

Requisitos:

1. Entrada do Usuário:
   - Solicitar ao usuário que insira uma frase.
   - Verificar se a entrada não está vazia e tratar o erro caso necessário. (crie uma nova função no bloco de funções)

2. Análise da Frase:
   - Contagem de Caracteres: Calcular o número total de caracteres da frase (incluindo espaços).
   - Contagem de Palavras: Dividir a frase em palavras (utilizando o método `split()`) e determinar quantas palavras existem.
   - Maior Palavra: Identificar a palavra com o maior número de caracteres dentre as encontradas.

pablo.coelho@gmail.com

Dever de casa

3. Manipulação e Formatação:

- Inversão da Frase:

- Inverter a frase por meio dos caracteres (utilizando slicing com `[::-1]`);

- Inverter a ordem das palavras e reconstruir a frase.

- Alteração de Caixa: Converter a frase para todas as letras maiúsculas e para todas as letras minúsculas.

- Tupla de Palavras: Criar uma tupla contendo todas as palavras da frase.

4. Saída Formatada:

- Utilizar f-strings para exibir os resultados de forma organizada, apresentando as seguintes informações:

- Número de caracteres da frase.

- Número de palavras.

- A palavra com maior comprimento.

- A frase invertida (por caracteres e por palavras).

- A frase em maiúsculas e minúsculas.

- A tupla formada pelas palavras da frase.

pablo.coelho@gmail.com


Thats all Folks!