NumPy

- Poucas funções/keywords (ferramentas)
- Funções/Keywords abrangentes
- Mais fácil memorizar
- Demanda mais criatividade para combinar Keywords para resolver um problema

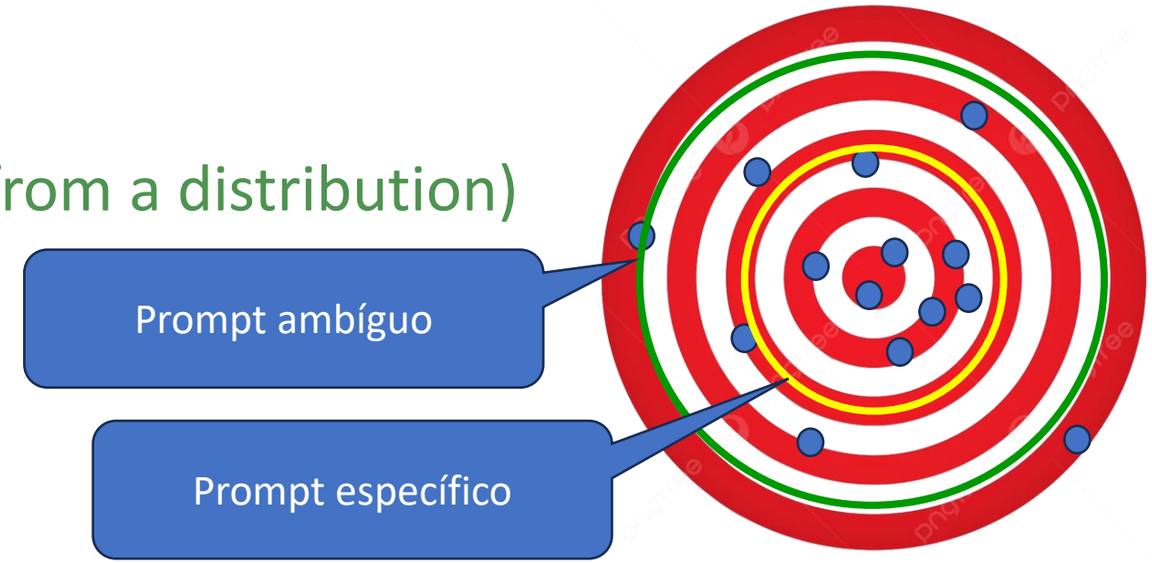
- Muitas funções (ferramentas)
- Funções específicas
- Mais difícil memorizar
- Demanda menos criatividade e mais leitura/consulta

https://numpy.org/doc/stable/search.html

Professor: Alex Pereira

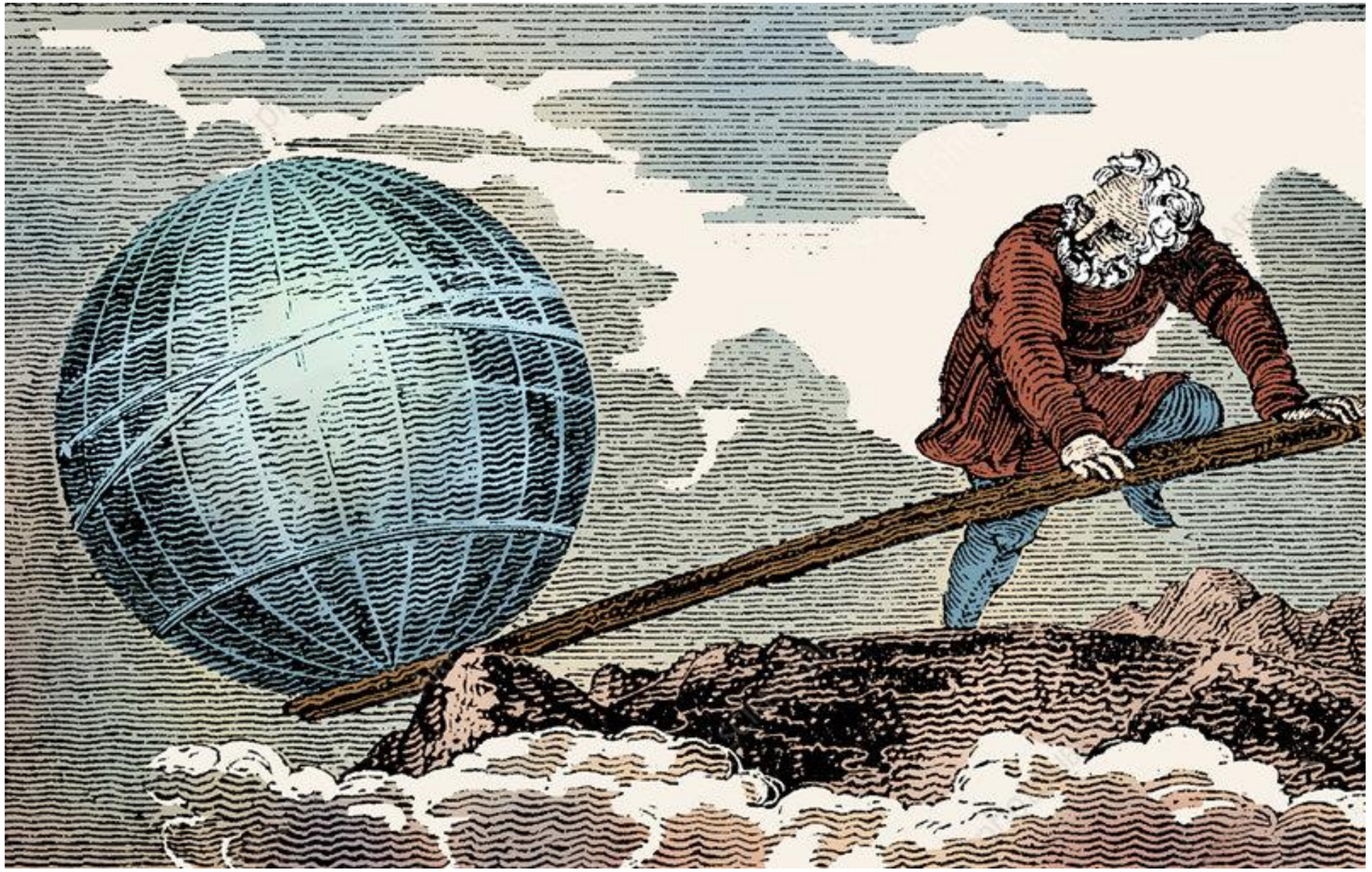
Por que um mesmo prompt pode passar ou não na validação dos exercícios

- LLMs são não determinísticos
  - Ou seja, a mesma pergunta repetida duas vezes pode gerar respostas distintas
    - ✓ Cite a 1a marca de shampoo que te vem a mente
      - 10x = 5x Pantene, 4x seda, 1x Head & Shoulders
  - Constroem texto a partir de uma distribuição de probabilidade do próximo token
    - ✓ 3Blue1Brown
      - Amostra uma distribuição (sample from a distribution)
        - Analogia



Expectativas quanto ao auxílio da IA

- Juiz usa inteligência artificial para fazer decisão e cita jurisprudência falsa.
  - Fonte. Nov. 2023.
- Não ler o código que a IA gerou
  - está no mesmo grupo de comportamentos indesejados, e
  - é improdutivo.
- Teste seu código
  - Com as entradas fornecidas;
  - E confira o resultado;
  - Se o resultado não está correto,
    - ✓ Inspecione o código para descobrir a causa



"Give me a place to stand and I will move the earth.“ (Arquimedes)

Me dê um lugar para apoiar e moverei a Terra

A IA é uma alavanca

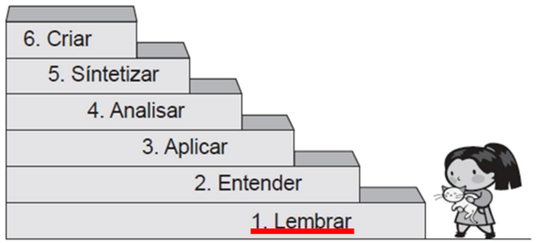
- A alavanca propicia mover objetos pesados com menos esforço físico,
  - mas ainda requer **compreensão** de onde posicionar o ponto de apoio e como aplicar a força corretamente,
- A IA ajuda os programadores a gerar código mais rapidamente,
  - mas ainda exige que eles **entendam** os fundamentos da programação
- A codificação (build) foi terceirizada para a IA
  - A especificação, não.

Não use como uma muleta

Que te deixa mais lento do que sem ela

Corrigir/melhorar o seu processo de trabalho

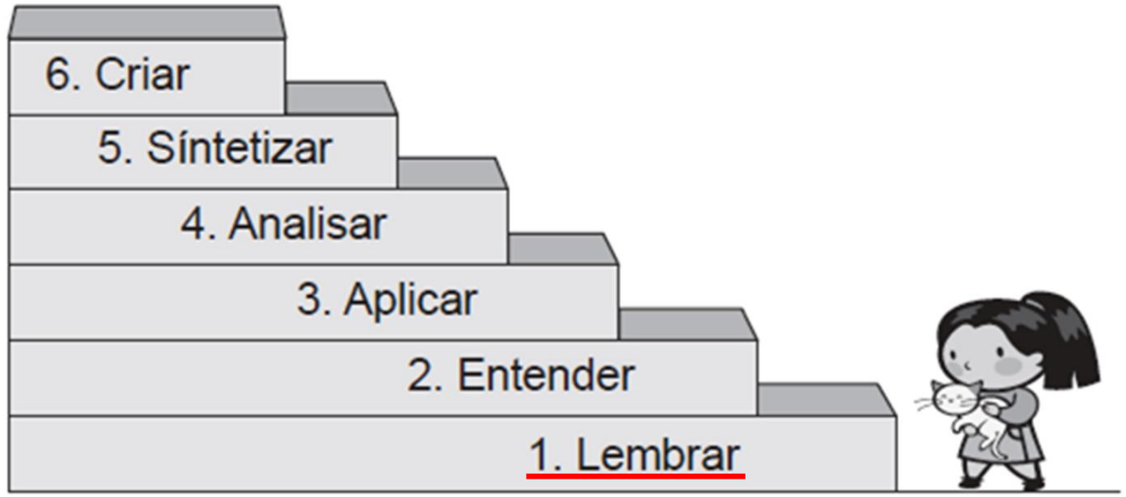
- O que você faz quando um prompt não foi validado?
  - Lê o código?
  - Lê a mensagem de erro?
  - Realiza uma busca ativa por um termo desconhecido?
    - ✓ Quantas buscas ativas você fez em média em cada questão?
    - ✓ Quantos novos termos você aprendeu?
  - Itera repetidamente sobre o prompt sem buscar entender mais sobre o problema?



- O que você faz ao se deparar com um termo desconhecido
  - Enxergue como uma oportunidade que o mundo te oferece
    - ✓ Pois você não sabia que não sabia.
- Princípios aplicáveis
  - Dê-me seis horas para derrubar uma árvore e passarei as quatro primeiras afiando o machado.
  - A definição de insanidade é fazer a mesma coisa repetidamente e esperar resultados diferentes.
- Atitude de pesquisa vs Terminar o exercício o mais rápido possível
  - Uma oportunidade única. Prof., monitores e um ambiente controlado

Conceitos novos de Python abordados na aula

- try
- except
- Pass
- sorted vs sort
- Sentença inline
- List comprehension
- HTTP
- JSON
- Zip
- Conjunto
- tupla



Funções Lambda (Anônima)

- São funções definidas em uma única sentença
- O resultado é o retorno de um valor
- São definidas pelo keyword lambda
  - Não há um significado especial para o termo lambda
    - ✓ a não ser "Estamos definindo uma função anônima"
- A função é dita anônima porque não damos um nome a ela

```
def exemplo(x):
     return x * 2

# Função lambda equivalente
equiv_anon = lambda x: x * 2
```

Funções Lambda (Anônima)

```
def exemplo(x):
     return x * 2

def apply_to_list(some_list, f):
     return [f(x) for x in some_list]

ints = [4, 0, 1, 5, 6]
r1 = apply_to_list(ints, lambda x: x * 2)
r2 = apply_to_list(ints, exemplo)
print(r1, r2)
```

```
[8, 0, 2, 10, 12]  [8, 0, 2, 10, 12]
```

Função Lambda para ordenar uma lista

- Criar uma função lambda para ordenar uma lista de strings
  - pelo tamanho dos seus elementos
    - ✓ do menor para o maior

```
strings = ['fo', 'card', 'bar', 'aaa2222a', 'abasb']
strings.sort(key=lambda x: len(x))
print(strings)
```

```
['fo', 'bar', 'card', 'abasb', 'aaa2222a']
```

NumPy (Numerical Python)

Pacote Python para computação numérica

Documentação oficial: https://numpy.org/doc/stable/search.html

Recursos do Numpy

- Computação eficiente com array multi-dimensional
  - armazena dados numa região contínua de memória;
  - As funções numpy escritas em C podem operar diretamente na memória.
- Funções matemáticas eficientes/rápidas em arrays
  - sem a necessidade de escrever loops (laços)
- Possui funções de
  - Álgebra linear, geração de números aleatórios, entre outros

Criando ndarrays (array de N dimensões)

```
import numpy as np
data1 = [6, 7.5, 8, 0, 1]
arr1 = np.array(data1)
print(arr1)
data2 = [[1, 2, 3, 4], [5, 6, 7, 8]]
arr2 = np.array(data2)
print(arr2)
print(np.zeros(5)) # Função que cria um array de zeros
```

```
[6.  7.5 8.  0.  1. ]
[[1 2 3 4]
 [5 6 7 8]]
[0. 0. 0. 0. 0.]
```

Alguns atributos dos ndarrays

```
# O tipo do dado é inferido. Mas também pode ser especificado
arr1 = np.array([6, 7.5, 8, 0, 1])
print(arr1.dtype) # tipo do dado
arr2 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]], dtype=np.int32)
print(arr2.shape) # Forma/dimensões do array
print(arr2.ndim) # Quantidade de dimensões
```

```
float64
(2L, 4L)
2
```

Alguns tipos de dados Numpy

Tipo básico	Tipo Numpy disponív.	Comentários
Boolean	bool	Tamanho de 1 byte
Integer	int8, int16, int32, int64, int128, int	int tem o tamanho do int padrão da plataforma C
Unsigned Integer	uint8, uint16, uint32, uint64, uint128, uint	uint tem o tamanho do uint padrão da plataforma C
Float	float32, float64, float, longfloat	Float é sempre de precisão dupla (64 bits). longfloat é um float maior cujo tamanho depende da plataforma.
String	str, Unicode	Unicode é sempre UTF32

Conversão de tipos (cast)

```
num_str = np.array(['1.25', '-9.6', '42'], dtype=np.string_)
arr_float = num_str.astype(float) # Converte para float
print(arr_float)
arr1 = np.array([3.7, -1.2, -2.6, 0.5, 12.9, 10.1])
arr1_int = arr1.astype(np.int32) # Converte para int32
print(arr1_int)
```

```
[ 1.25 -9.6  42. ]
[ 3 -1 -2  0 12 10]
```

Array bi-dimensional

```
data = np.array([[ 0.78, 1.87, 0.82], [-1.60, -0.01, -0.02]])
print(data)
print(data.shape) # Forma/dimensões do array
print(data.ndim) # Quantidade de dimensões
```

```
[[ 0.78  1.87  0.82]
 [-1.6  -0.01 -0.02]]
(2, 3)
2
```

Curiosidade: Qual o dado bi-dimensional mais comum na computação ?

Operações vetorizadas em ndarrays

- Funções e operações em várias dimensões sem loops
  - Mais eficientes (preferíveis)

```
data = np.array([[ 3, 4, 8], [10, -4, -2]])
print(data)
print(data * 10)
print(data + data)
```

```
[[ 3  4  8]
 [10 -4 -2]]
[[ 30  40  80]
 [100 -40 -20]]
[[ 6  8 16]
 [20 -8 -4]]
```

Aritmética com NumPy Arrays

```
arr = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

print(arr * arr2)

print(arr - arr2)

print(1 / arr)

print(arr ** 0.5)

print(arr2 > arr)
```

```
[ 4 10 18]
[-3 -3 -3]
[1.         0.5        0.33333333]
[1.         1.41421356 1.73205081]
[ True  True  True]
```

Outros métodos de criação de array

```
print(np.arange(10)) #cria array de números sequenciais
print(np.arange(2, 10, dtype=float))
print(np.zeros(5))
print(np.ones((2,3)))
```

```
[0 1 2 3 4 5 6 7 8 9]
[2. 3. 4. 5. 6. 7. 8. 9.]
[0. 0. 0. 0. 0.]
[[1. 1. 1.]
 [1. 1. 1.]]
```

Comparação de Desempenho: Numpy array vs list

```
my_arr = np.arange(10000000) #cria array de números sequenciais
my_list = list(range(10000000))
# %time mede o tempo tomado pela execução da linha
%time my_arr2 = my_arr * 2
%time my_list2 = [x * 2 for x in my_list]
print(my_arr2[1:5])
print(my_list2[1:5])
```

```
Wall time: 32 ms
Wall time: 1.15 s
[2 4 6 8]
[2, 4, 6, 8]
```

Slicing (fatiar) NumPy Arrays

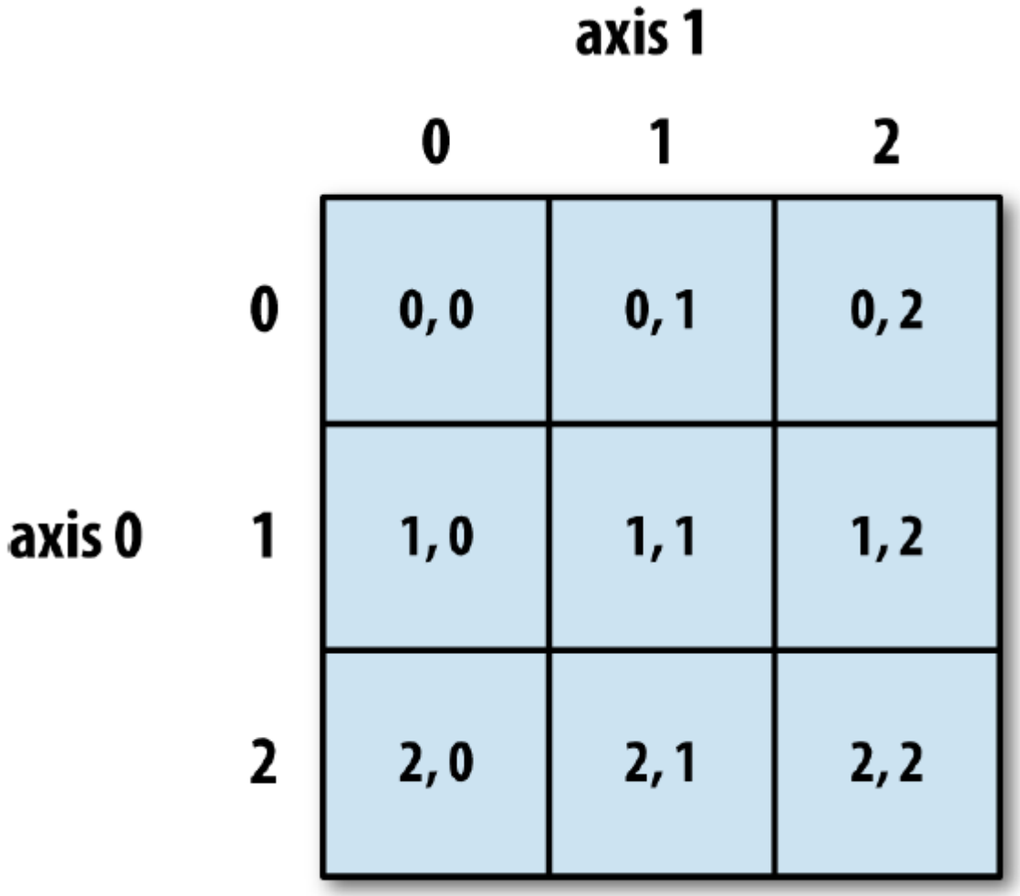
- Nas listas os slices (cortes) são cópias;
- Nos arrays numpy os slices são views (visualizações)
  - para copiar, use a função: .copy() , exemplo: arr[1:4].copy()

```
arr = np.arange(10)
print(arr)
arr_slice = arr[5:8]
arr_slice[1] = 12345
print(arr)
li2 = list(range(10)) # O equivalente com uma lista
list_slice = li2[5:8]
list_slice[1] = 12345
print(li2)
```

Slicing (fatiar) de arrays bidimensionais

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr2d)
print(arr2d[2])
print(arr2d[0][2]) # Tanto faz
print(arr2d[0, 2]) # Tanto faz
print(arr2d[:, 1])
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[7 8 9]
3
3
[2 5 8]
```



Outras funções numpy

- np.sum() – Soma os elementos de um array
  - Todos os elementos ou os elementos em cada um dos eixos
    - ✓ axis=0, soma ao longo das colunas
    - ✓ axis=1, soma ao longo das linhas

```
1 ar = np.array([[0, 1], [0, 5]])
2 print(ar)
3 print(np.sum(ar))
4 print(np.sum(ar, axis=0))
5 print(np.sum(ar, axis=1))
```

```
[[0 1]
 [0 5]]
6
[0 6]
[1 5]
```

Outras funções numpy

- np.mean() – Calcula a media dos elementos de um array
- np.std() – Calcula o desvio padrão dos elementos de um array
- np.any() – Retorna um booleano (ou um array de booleanos) informando se há pelo menos um elemento True.

```
1 arr2 = np.array([[True, False], [True, True]])
2 print(arr2)
3 print(np.any(arr2))
4 print(np.any(arr2, axis=0))
```

```
[[ True False]
 [ True  True]]
True
[ True  True]
```

Outras funções numpy

- np.argmax() – Retorna os índices dos valores máximos ao longo de um eixo
  - Analogamente np.argmin() retorna os índices dos valores mínimos

```
1  a = np.array([[10, 11, 12],
2                [13, 14, 15]])
3  print(a)
4  print(np.argmax(a))
5  print(np.argmax(a, axis=0))
6  print(np.argmax(a, axis=1))
```

```
[[10 11 12]
 [13 14 15]]
5
[1 1 1]
[2 2]
```

Filtrar um Array baseado numa condição

- A condição que você deseja filtrar (especificação de requisito)
  - é colocada entre colchetes

```
1  arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
2  print(arr)
3  print(arr < 4)
4  print(arr[arr < 4])
```

```
[0 1 2 3 4 5 6 7 8]
[ True  True  True  True False False False False False]
[0 1 2 3]
```

Operações lógicas com Array Numpy

- & faz a operação AND e | faz a operação OR
  - **sempre** use os parenteses para fazer operações lógicas com arrays

```
1  arr1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
2  selecao = (arr1 > 2) & (arr1 <5)
3  print(selecao)
4  print(arr[selecao])
```

```
[False False False  True  True False False False False]
[3 4]
```

```
1  selecao2 = (arr1 < 2) | (arr1 > 5)
2  print(selecao2)
3  print(arr1[selecao2])
```

```
[ True  True False False False False  True  True  True]
[0 1 6 7 8]
```

Prática no Colab Notebook

- Escolham por onde começar: Teoria, Warmup ou Exercícios;
  - As soluções dos warmups já estão publicadas;
  - As soluções dos exercícios extra serão disponibilizadas ao final do dia;
- É esperado que não terminem todos os exercícios durante a aula;
  - Façam o restante ao longo da semana.
  - https://numpy.org/doc/stable/search.html