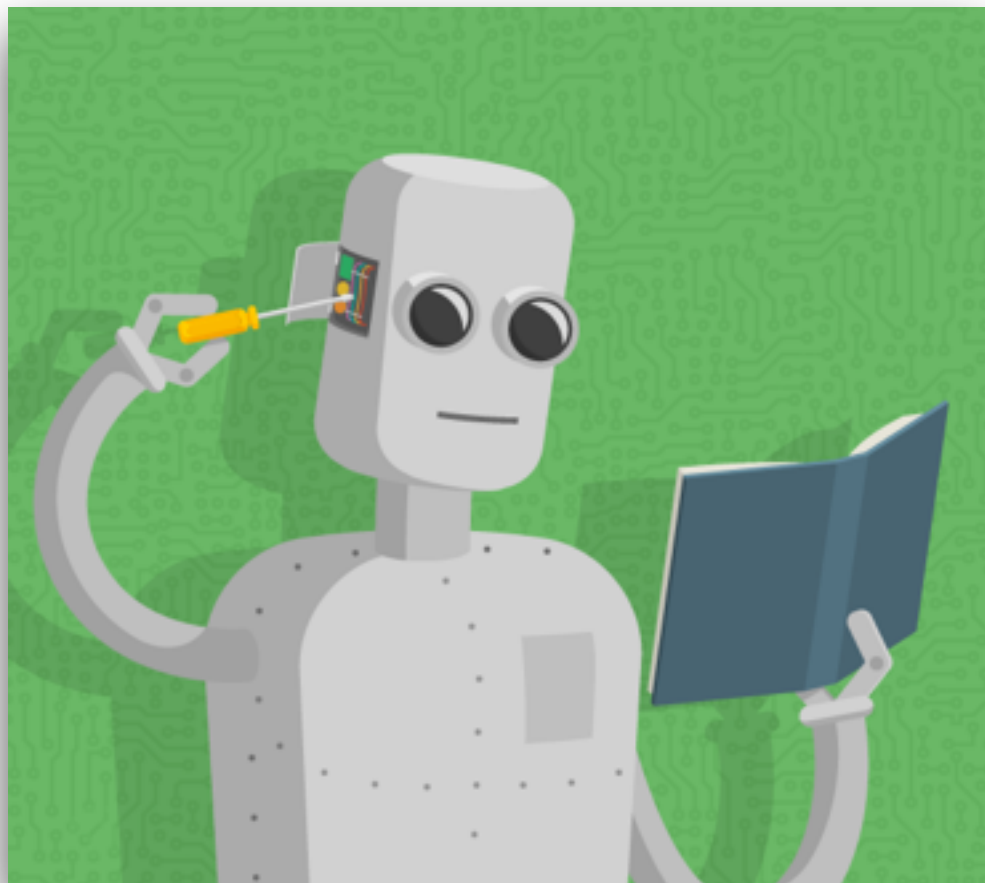




UNIFOR
ENSINANDO E APRENDENDO

Disciplina de Inteligência Artificial (Aula 2)



Introdução ao Python

Apresentação



Francisco Nauber Bernardo Gois

Analista aprendizado de máquina
no Serviço Federal de Processamento
de Dados

Doutorando em Informática Aplicada

Mestre em Informática Aplicada

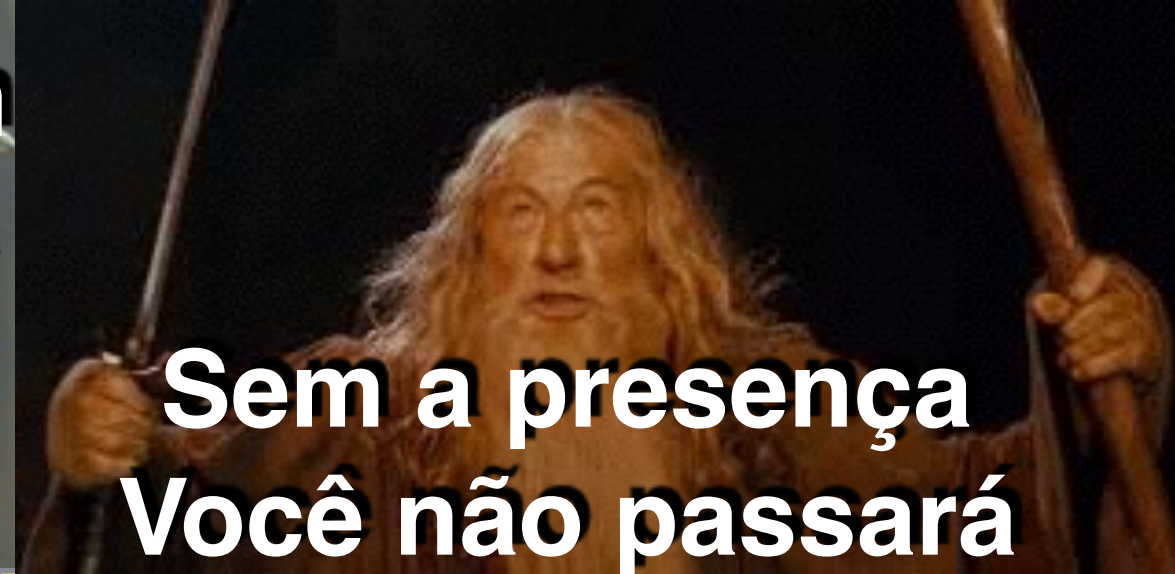
Especialista em desenvolvimento WEB

**Jovem Padawan
procure na aula**



**ao telefone
não falar**

**Sem a presença
Você não passará**



**Para melhor
desempenho na
aula**

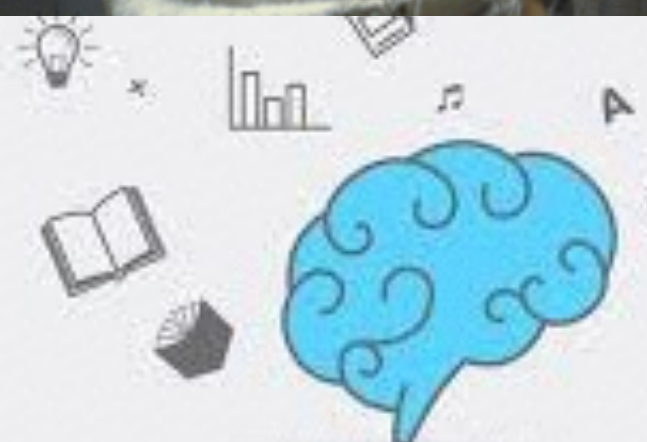
**Cuidado
com o
Horário**

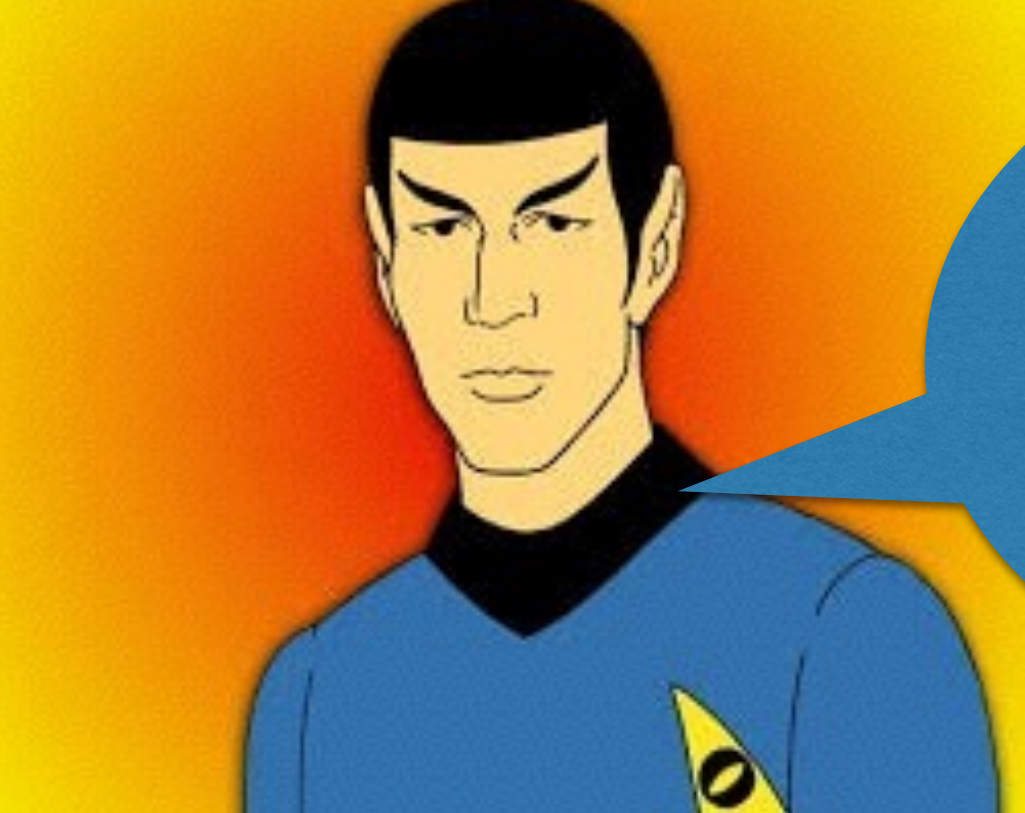
**Não
teremos
pontuação fora dos
trabalhos e provas
da disciplina**

**Buscar
aprendizado
ao invés de
pontos**



**Procure
não
conversar
durante a
aula**





OS trabalhos
deveram ser
entregues uma
semana antes
da prova

Um cadeira longa
e prospera

Não teremos
pontos após a prova
não adianta pedir



Cronograma da Disciplina



**Introdução a
Inteligência
Artificial**



**Introdução
Aprendizado
de Máquina**

**Introdução
ao Python**



O que vimos na aula passada

- **Introdução ao Inteligência Artificial.**
- **Cronograma da disciplina.**
- **Como será a composição da nota.**
- **Principais áreas da inteligência artificial**

Objetivo da Aula

- **Introdução ao Python**
- **Conhecer o Numpy**
- **Conhecer conceitos de busca**

Numpy Cheat Sheet

PYTHON PACKAGE

CREATED BY: ARIANNE COLTON AND SEAN CHEN

NUMPY (NUMERICAL PYTHON)

What is NumPy?

Foundation package for scientific computing in Python

Why NumPy?

- Numpy 'ndarray' is a much more efficient way of storing and manipulating "numerical data" than the built-in Python data structures.
- Libraries written in lower-level languages, such as C, can operate on data stored in Numpy 'ndarray' without copying any data.

N-DIMENSIONAL ARRAY (NDARRAY)

What is NdArray?

Fast and space-efficient multidimensional array (container for homogeneous data) providing vectorized arithmetic operations

Create NdArray	<code>np.array(seq1)</code> # seq1 - is any sequence like object, i.e. [1, 2, 3]
Create Special NdArray	<code>1, np.zeros(10)</code> # one dimensional ndarray with 10 elements of value 0 <code>2, np.ones(2, 3)</code> # two dimensional ndarray with 6 elements of value 1 <code>3, np.empty(3, 4, 5) *</code> # three dimensional ndarray of uninitialized values <code>4, np.eye(N) or np.identity(N)</code> # creates N by N identity matrix
NdArray version of Python's range	<code>np.arange(1, 10)</code>
Get # of Dimension	<code>ndarray1.ndim</code>
Get Dimension Size	<code>dim1size, dim2size, .. = ndarray1.shape</code>
Get Data Type **	<code>ndarray1.dtype</code>
Explicit Casting	<code>ndarray2 = ndarray1.astype(np.int32) ***</code>

- Cannot assume `empty()` will return all zeros. It could be garbage values.

- ** Default data type is 'np.float64'. This is equivalent to Python's float type which is 8 bytes (64 bits); thus the name 'float64'.
- *** If casting were to fail for some reason, 'TypeError' will be raised.

SLICING (INDEXING/SUBSETTING)

- Slicing (i.e. `ndarray1[2:6]`) is a 'view' on the original array. **Data is NOT copied**. Any modifications (i.e. `ndarray1[2:6] = 8`) to the 'view' will be reflected in the original array.
- Instead of a 'view', explicit copy of slicing via :
`ndarray1[2:6].copy()`
- Multidimensional array indexing notation :
`ndarray1[0][2]` or `ndarray1[0, 2]`

* Boolean indexing :

```
ndarray1[(names == 'Bob') | (names == 'Will'), 2:]
```

'2:' means select from 3rd column on

- Selecting data by boolean indexing **ALWAYS** creates a copy of the data.
- The 'and' and 'or' keywords do NOT work with boolean arrays. Use & and |.

* Fancy indexing (aka 'indexing using integer arrays')

Select a subset of rows in a particular order :

```
ndarray1[ [3, 8, 4] ]
```

```
ndarray1[ [-1, 6] ]
```

negative indices select rows from the end

- Fancy indexing **ALWAYS** creates a copy of the data.

NUMPY (NUMERICAL PYTHON)

Setting data with assignment :

```
ndarray1[ndarray1 < 0] = 0 *
```

- If ndarray1 is two-dimensions, `ndarray1 < 0` creates a two-dimensional boolean array.

COMMON OPERATIONS

1. Transposing

- A special form of reshaping which returns a 'view' on the underlying data without copying anything.

```
ndarray1.transpose() or  
ndarray1.T or  
ndarray1.swapaxes(0, 1)
```

2. Vectorized wrappers (for functions that take scalar values)

- `math.sqrt()` works on only a scalar

```
np.sqrt(seq1) # any sequence (list,  
ndarray, etc) to return a ndarray
```

3. Vectorized expressions

- `np.where(cond, x, y)` is a vectorized version of the expression 'x if condition else y'

```
np.where([True, False], [1, 2],  
[2, 3]) => ndarray (1, 3)
```

- Common Usages :

```
np.where(matrixArray > 0, 1, -1)  
=> a new array (same shape) of 1 or -1 values
```

```
np.where(cond, 1, 0).argmax() *  
=> Find the first True element
```

- `argmax()` can be used to find the index of the maximum element. Example usage is find the first element that has a "price > number" in an array of price data.

4. Aggregations/Reductions Methods (i.e. mean, sum, std)

Compute mean	<code>ndarray1.mean()</code> or <code>np.mean(ndarray1)</code>
Compute statistics over axis *	<code>ndarray1.mean(axis = 1)</code> <code>ndarray1.sum(axis = 0)</code>

- axis = 0 means column axis, 1 is row axis.

5. Boolean arrays methods

Count # of 'Trues' in boolean array	<code>(ndarray1 > 0).sum()</code>
If at least one value is 'True'	<code>ndarray1.any()</code>
If all values are 'True'	<code>ndarray1.all()</code>

Note: These methods also work with non-boolean arrays, where non-zero elements evaluate to True.

6. Sorting

Inplace sorting	<code>ndarray1.sort()</code>
Return a sorted copy instead of inplace	<code>sorted1 = np.sort(ndarray1)</code>

7. Set methods

Return sorted unique values	<code>np.unique(ndarray1)</code>
Test membership of ndarray1 values in [2, 3, 6]	<code>resultBooleanArray = np.in1d(ndarray1, [2, 3, 6])</code>

- Other set methods : `intersect1d()`, `union1d()`, `setdiff1d()`, `setxor1d()`

8. Random number generation (np.random)

- Supplements the built-in Python random * with functions for efficiently generating whole arrays of sample values from many kinds of probability distributions.

```
samples = np.random.normal(size = (3, 3))
```

- Python built-in random **ONLY** samples one value at a time.

Created by Arianne Colton and Sean Chen

www.datasciencefree.com

Based on content from 'Python for Data Analysis' by Wes McKinney

Updated: August 18, 2016

Introdução ao Python

```
>>> a=1
```

```
>>> b=2
```

```
>>> a == b # == testa se a é igual a b  
False
```

```
>>> a != b # != testa se a é diferente de b  
True
```

```
>>> a <> b # <> também testa se a é diferente de b  
True
```

```
>>> a > b # > testa se a é maior que b  
False
```

Introdução ao Python

>>> 23 # É o mesmo que 2 elevado a 3 (dois ao cubo).**

8

>>> 2(3+6) # Dois elevado a 9**

512



Variáveis

```
>>> valor1='Boa tarde!'
```

```
>>> valor1
```

```
'Boa tarde!'
```

```
>>> type(valor1)
```

```
<type 'str'>
```



Variáveis

```
>>> palavra='termodinâmica'  
>>> palavra  
'termodin\xe2mica'
```




```
>>> print palavra  
'termodinâmica'
```



Variáveis

```
>>> palavra[2]
```

```
'r'
```

```
>>> 2*palavra[0]
```

```
'tt'
```



Listas

```
>>> lista=[1,2,3]  
>>> lista  
[1, 2, 3]
```

Agora
vamos
criar um
conjunto
de valores



Listas

```
>>> lista[0]
```

```
1
```

```
>>> lista[0]+lista[2]
```

```
4
```

Agora
vamos
criar um
conjunto
de valores



Introdução ao Numpy



NumPy é o pacote fundamental para a computação científica com o Python. Contém, entre outras coisas:

Um poderoso objeto de matriz N-dimensional

Funções sofisticadas (transmissão)

Ferramentas para integrar C / C ++ e código Fortran



Arrays

```
>>> a = np.array([1, 4, 5, 8], float)
>>> a
array([ 1.,  4.,  5.,  8.])
>>> type(a)
<type 'numpy.ndarray'>
```

Vamos usar o Numpy



Arrays

```
>>> a[:2]
array([ 1., 4.])
>>> a[3]
8.0
>>> a[0] = 5.
>>> a
array([ 5., 4., 5., 8.])
```

Exibindo dados de
um array



Vídeo

<https://youtu.be/cAICT4AI5Ow>

Vídeo Informativo



Arrays

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.]])
>>> a[0,0]
1.0
>>> a[0,1]
2.0
```

Criando array do
tipo float



Arrays

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> a
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.]])
>>> a[0,0]
1.0
>>> a[0,1]
2.0
```

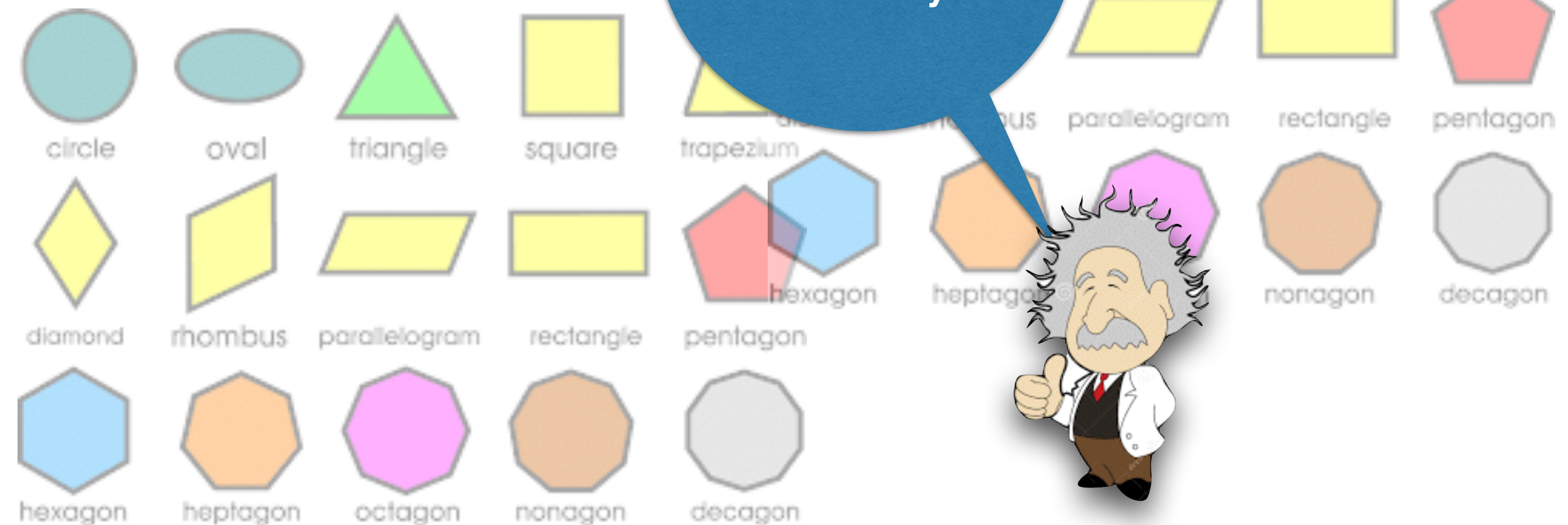
Criando array do
tipo float



Shape de um array

```
>>> a.shape  
(2, 3)
```

Imprimindo o
Shape de
um array



Tipo de um array

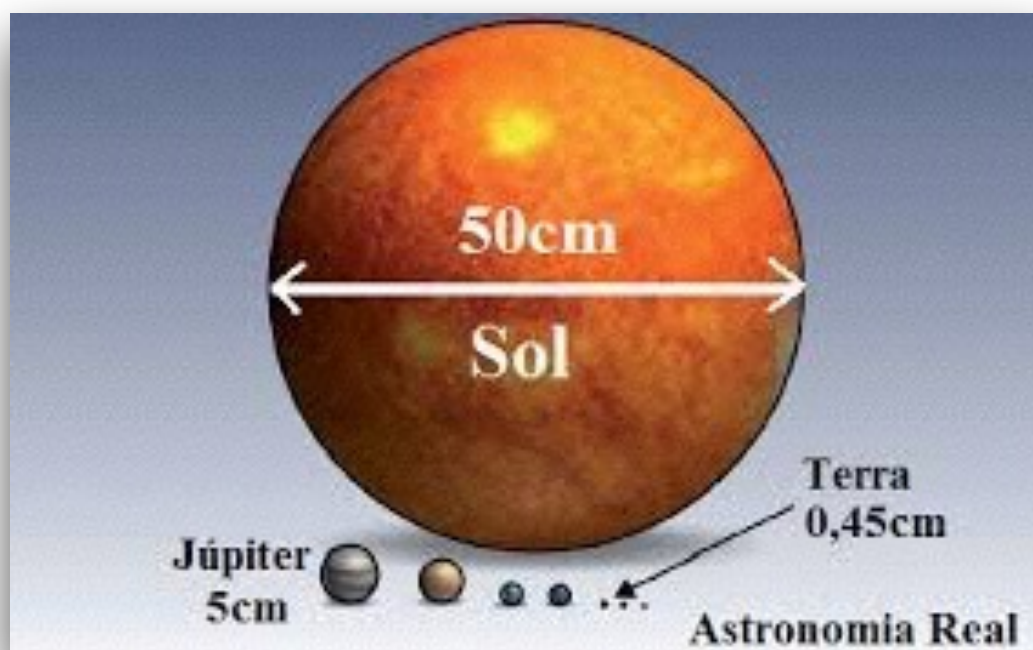
```
>>> a.dtype  
dtype('float64')
```

Imprimindo o
tipo de
um array



Tamanho de um array

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> len(a)
2
```



Imprimindo o
tamanho de
um array

Testando se elemento está na array

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]], float)
>>> 2 in a
True
>>> 0 in a
False
```

Testando se
elemento está
na array



Reshaping

```
>>> a = np.array(range(10), float)
>>> a
array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])
>>> a = a.reshape((5, 2))
>>> a
array([[ 0., 1.],
       [ 2., 3.],
       [ 4., 5.],
       [ 6., 7.],
       [ 8., 9.]])
>>> a.shape
(5, 2)
```

Reshaping



Clonando array

```
>>> a = np.array([1, 2, 3], float)
>>> b = a
>>> c = a.copy()
>>> a[0] = 0
>>> a
array([0., 2., 3.])
>>> b
array([0., 2., 3.])
>>> c
array([1., 2., 3.])
```



Clonando um array



Transformando em Lista

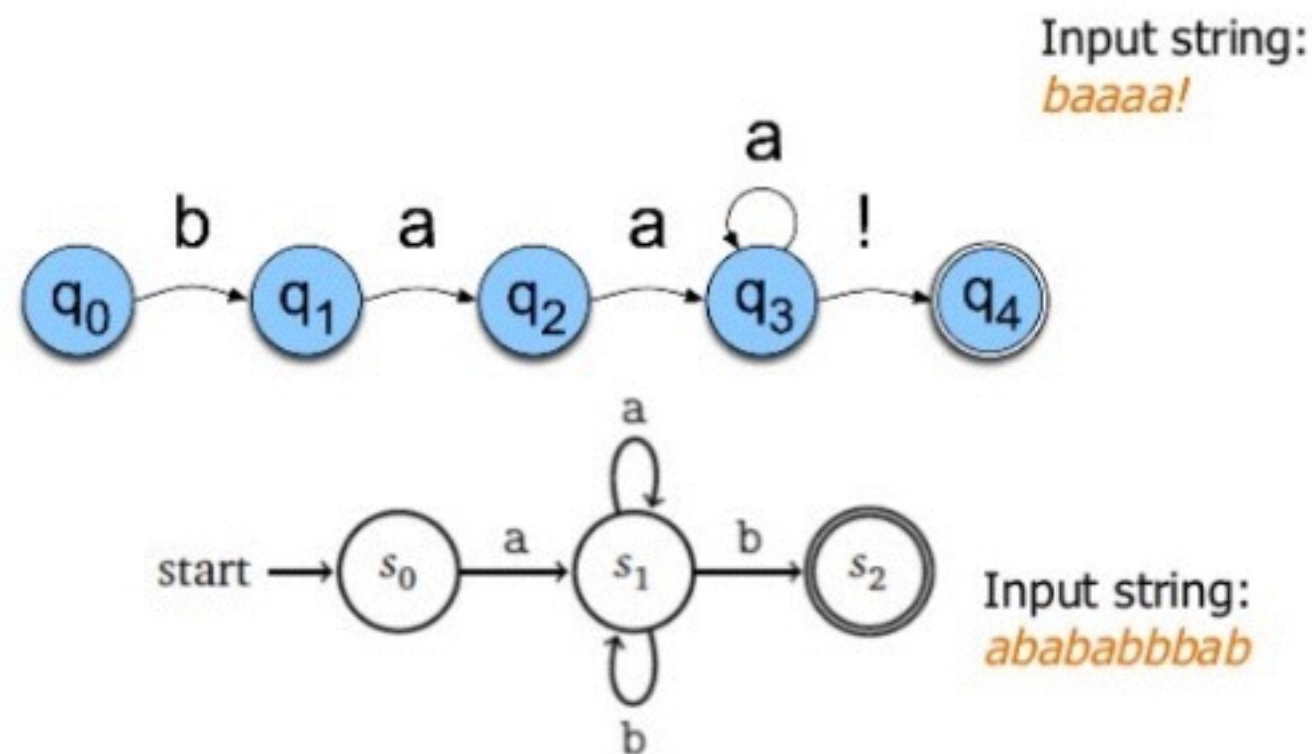
```
>>> a = np.array([1, 2, 3], float)
>>> a.tolist()
[1.0, 2.0, 3.0]
>>> list(a)
[1.0, 2.0, 3.0]
```

Transformando
em Lista

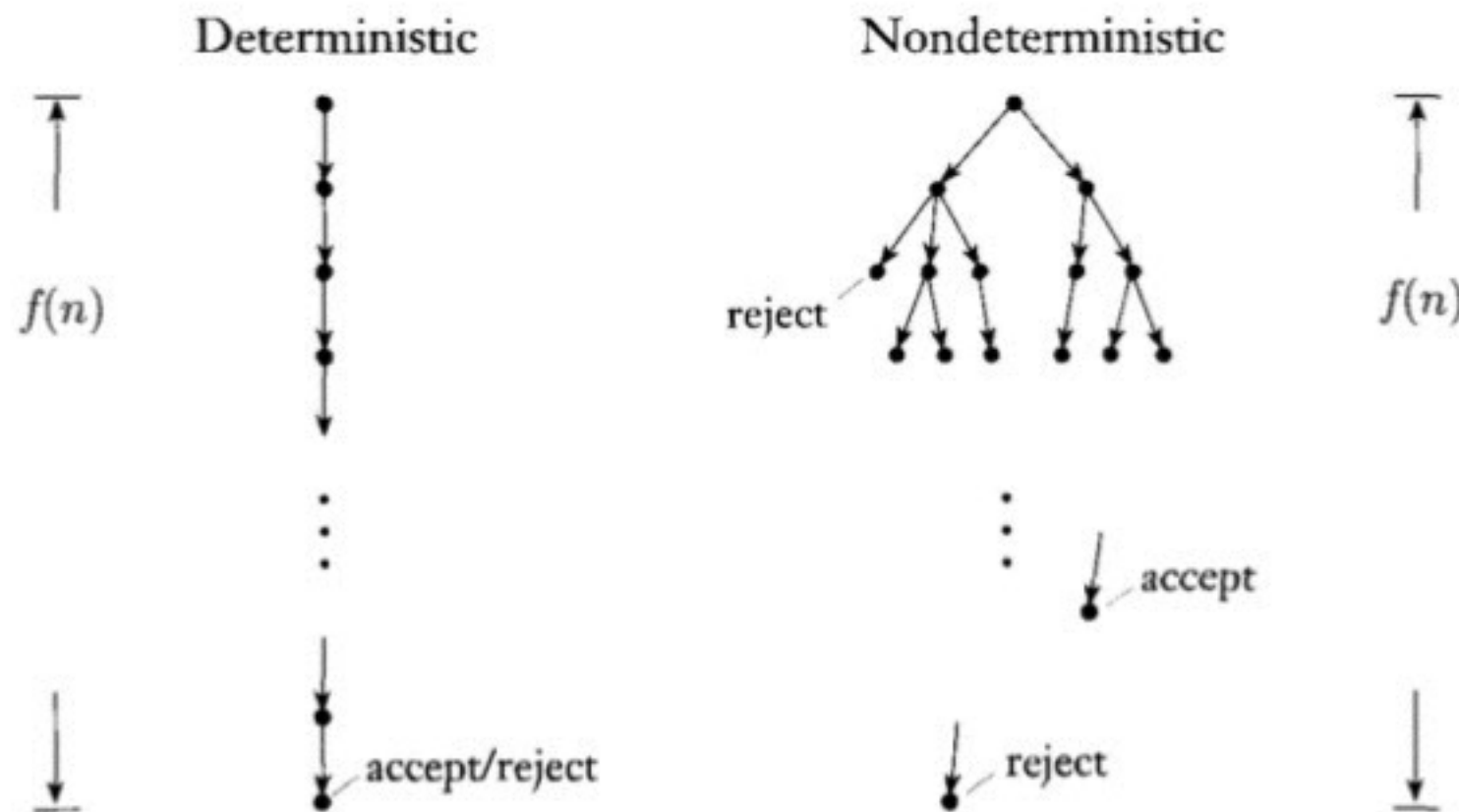


Problema Determinístico e não determinístico

Problema determinístico vs não determinístico



Problema Determinístico e não determinístico

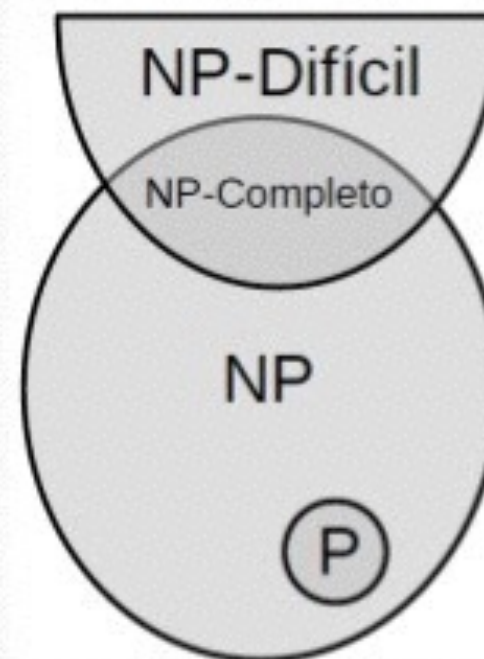
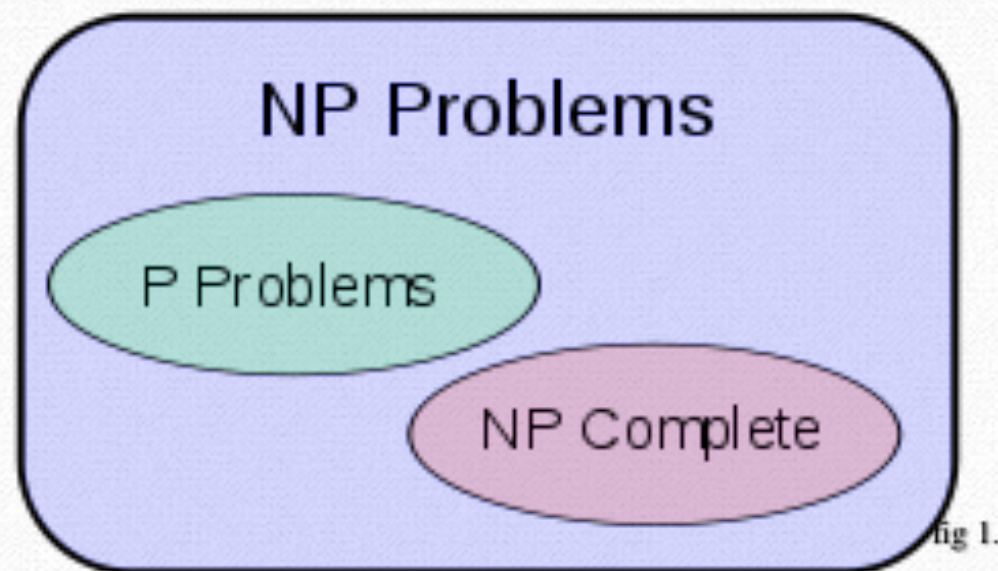


Problema Determinístico e não determinístico



Tipos de Problemas

- Para compreender o problema abordado neste trabalho, é necessário conhecer a classe de problemas NP-Completo. Para melhor compreender essa classe e também as classes P e NP, segue uma definição básica de cada uma delas.

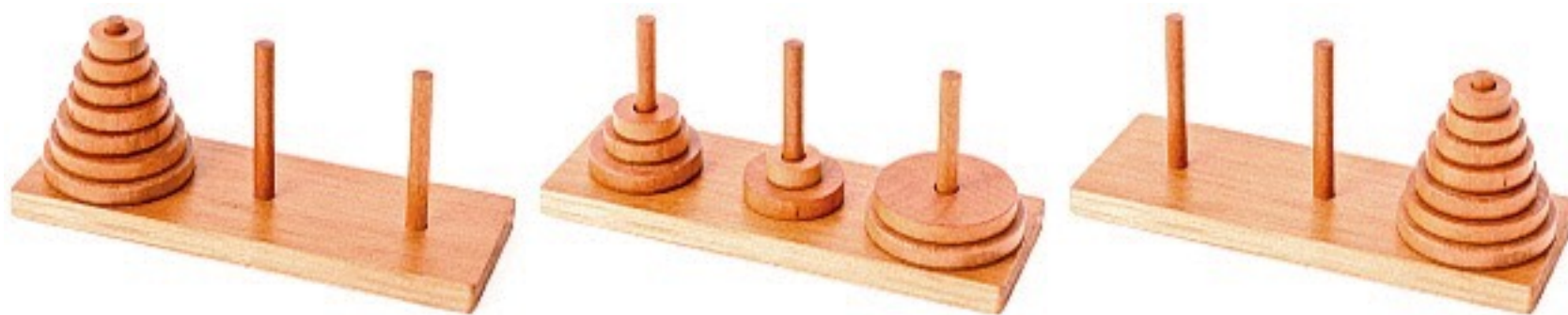


Tipos de Problemas

Problemas P, NP e NP-completo:

O problema P é um problema de decisão (ou seja, a resposta pode ser sim ou não), que possa ser resolvido em tempo polinomial.

O problema NP (Non-Deterministic Polynomial time), ou seja "Tempo polinomial não-determinístico". O Problema NP pode ser um Problema P (na verdade o assunto é mais complexo), com a característica de "não determinístico", ou seja, ele pode ser provado em tempo polinomial.

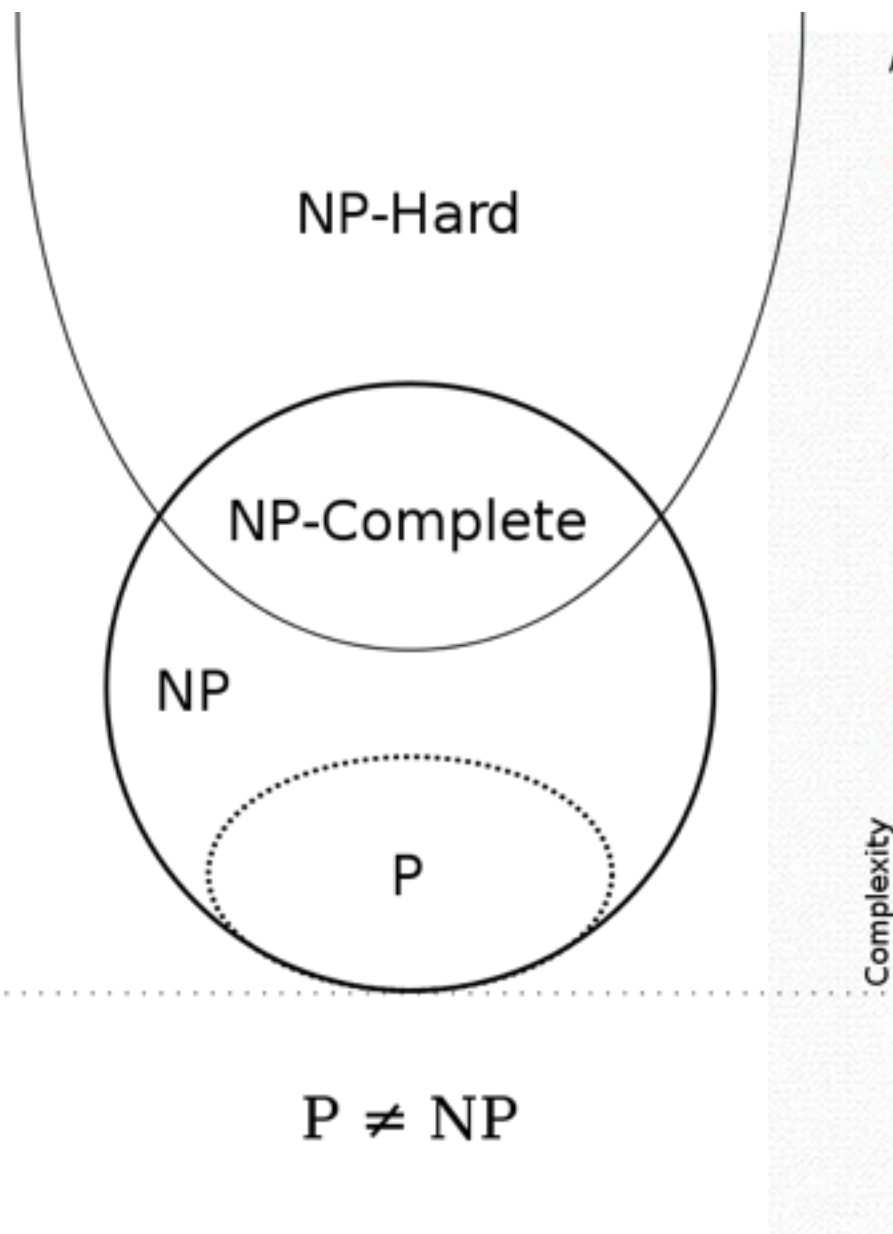


Tipos de Problemas

2.1 A Classe P de problemas

- A classe P engloba todos os problemas decisórios que podem ser resolvidos por um algoritmo de tempo polinomial.
- Um problema cujo algoritmo é de ordem $O(n^k)$ com uma constante k muito grande é considerado eficiente, pois é provável, pelo que se vê ao longo da história dos algoritmos, que logo sejam encontrados novos algoritmos mais eficientes que resolvam tal problema com uma complexidade menor.

Tipos de Problemas



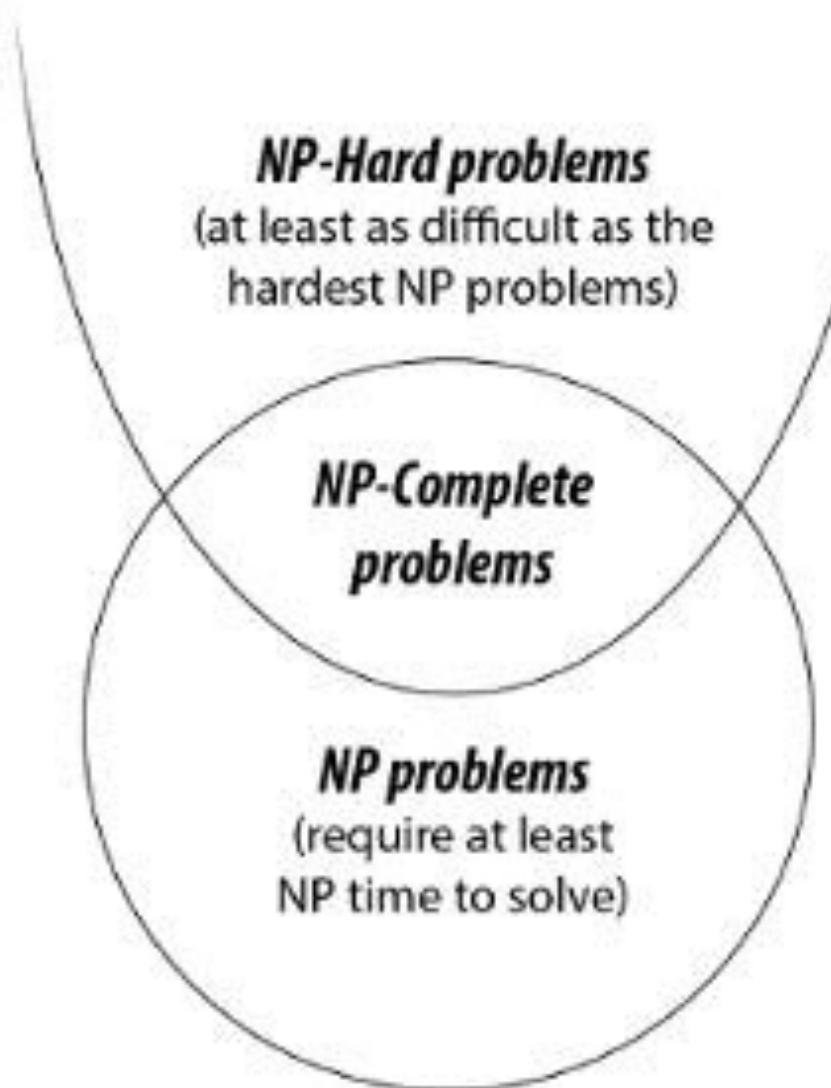
2.2 A Classe NP de problemas

- A classe NP de problemas é o conjunto de todos os problemas de decisão para os quais existem verificadores polinomiais. Esta classe corresponde à classe dos problemas que poderíamos chamar "razoáveis".
- É importante lembrar que todo problema pertencente à classe P também pertence à classe NP, uma vez que se um problema pode ser resolvido em tempo polinomial, ele pode ser igualmente verificado em tempo polinomial.

Tipos de Problemas

Three classes of decision problems

- **P** is the set of decision problems that can be solved in polynomial time.³ Intuitively, P is the set of problems that can be solved quickly.
- **NP** is the set of decision problems with the following property: If the answer is YES, then there is a proof of this fact that can be checked in polynomial time.
- **co-NP** is the opposite of NP. If the answer to a problem in co-NP is NO, then there is a proof of this fact that can be checked in polynomial time.



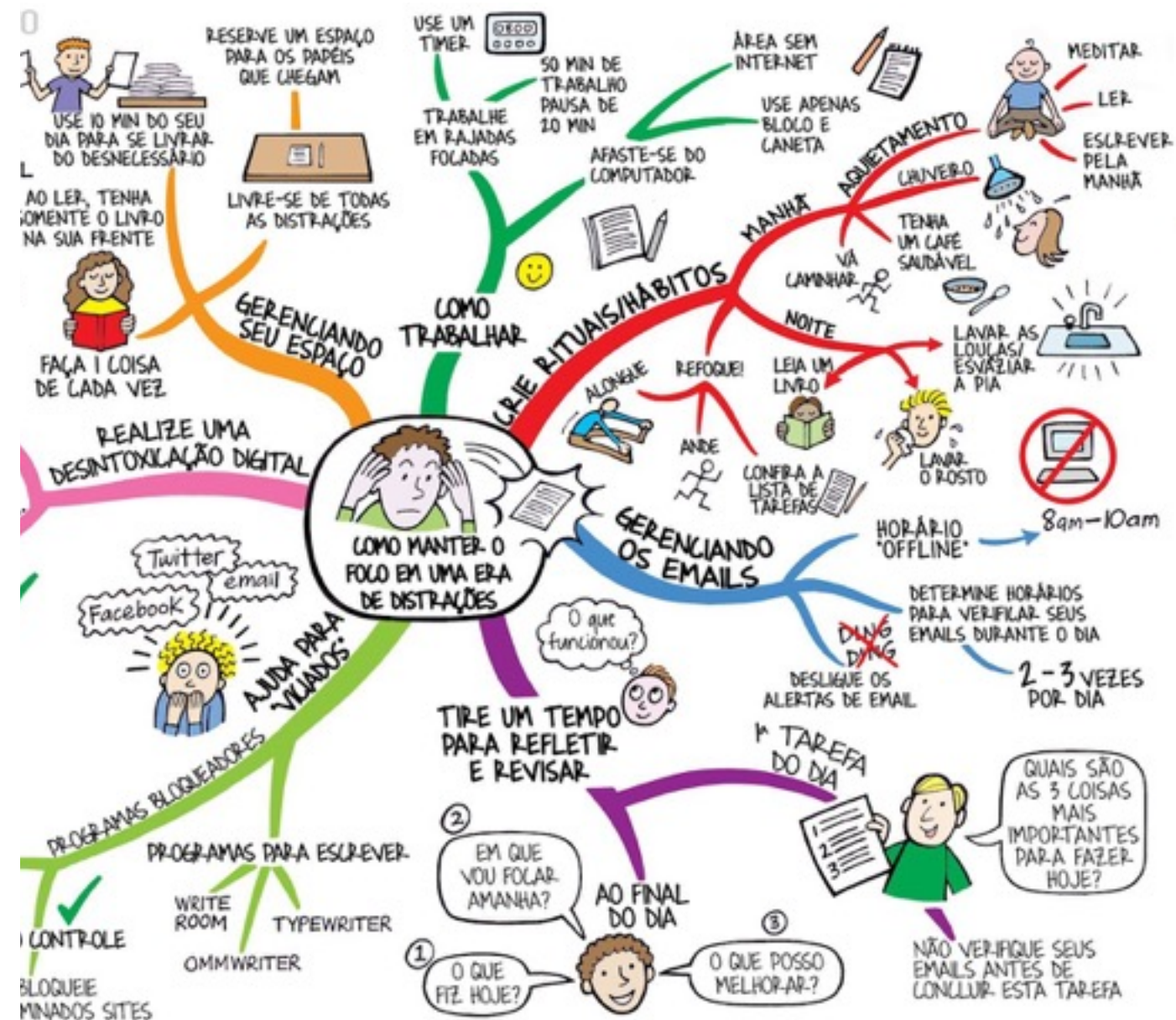
Tipos de Problemas

NP-Hard

- **What is NP-Hard?**
- NP-Hard are problems that are at least as hard as the hardest problems in NP. Note that NP-Complete problems are also NP-hard. However not all NP-hard problems are NP (or even a decision problem), despite having 'NP' as a prefix. That is the NP in NP-hard does not mean 'non-deterministic polynomial time'. Yes this is confusing but its usage is entrenched and unlikely to change.

Mapa Mental

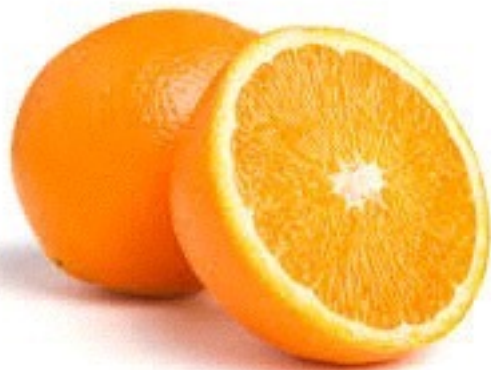
Mapa mental, ou mapa da mente é o nome dado para um tipo de diagrama, sistematizado pelo psicólogo inglês Tony Buzan, voltado para a gestão de informações, de conhecimento e de capital intelectual



Exercício

Exercício 1

Separem-se em equipe de até 5 pessoas e criem um mapa mental descrevendo o que foi feito na aula (10 minutos).



Ce sont les Batraciens, que le voyageur rencontre à chaque pas sur le sol japonais, ils fréquentent les lieux habités et peuplent les fertiles vallées, d'où une culture exubérante à basculer les vivants et les autres animaux sauvages. Fuyant la lumière et les lieux habités, la femelle des Batraciens à queue généralement dégingandée par le nom de Salamandre, se retire dans les profondeurs des montagnes riches en sources et en cascades. L'existence de différentes espèces de Salamandres au Japon, est d'une haute importance pour l'étude de la distribution géographique de ces reptiles, mais ce qui est plus important encore c'est la découverte d'une espèce de Salamandre, qui par sa taille extraordinaire et sa forme spéciale, rappelle une création antédiluvienne, cette Salamandre le Géant des Batraciens est le représentant d'une race appartenant à cette longue période de notre globe qui sépare les formations fossilifères des terrains tertiaires, et qui vit apparues aux milieux de nos reptiles gigantesques et d'organisations bizarres. Je vais parler de l'époque débris, la célèbre Salamandre fœtale des cantons d'Osaka qui depuis Souchou jusqu'à Omi dont les dents ont été si vite livrées sur le monde primitif, a été l'objet des spéculations des naturalistes.

Notre grande Salamandre (Salamandre japonica) vit dans les profondes vallées des hautes montagnes de Nippon entre le 34° et 36° de lat. N., elle séjourne dans les ruisseaux, dans les lacs et dans les lieux humides par les eaux pluviales ou même des cratères des Volcans éteints à une hauteur de 4 à 5000 pieds au dessus du niveau de la mer. Quelquefois elle quitte pendant la nuit les eaux qui lui servent d'habitat, mais son organisation et ses habitudes la rappellent bientôt dans cet élément, car elle meurt plus facilement que sur terre, une salamandre qui consiste en petits poissons, en grenouilles et en vases. C'est à Sakurata le petit village situé aux pentes du mont Sennaga vers 15 Mi environ à l'est de Misaki, que j'ai observé pour la première fois cette Salamandre. En de mes disciples, le Docteur Enryu, avait chargé un herboriste qui habite cette montagne de faire la recherche de ce rare et curieux animal. Malgré les renseignements des montagnards le Souchouman — c'est le nom indigène vulgaire de la grande Salamandre — se trouve le plus souvent dans les montagnes d'Okudayama. Fui en le loup des montagnes rapporteur une vivante en Europe. Elle existe encore au Mont des Pays-Bas, on elle a atteint une longueur d'environ trois pieds, taille extraordinaire que je n'ai jamais observée même au Japon. Cet individu a été depuis plusieurs années l'objet des observations de M. Schlegel, qui en a donné une description complète sous tous les rapports.



Francisco Nauber Bernardo Gois
Email: naubergois@gmail.com