Pedro Kantek

O ESTADO DA ARTE EM COMPUTAÇÃO NA ENGENHARIA: O USO DE PYTHON E SEUS 170.000 PACOTES

Curitiba

2019

UFPR


https://www.algoritmovivo.com
INÍCIO Taxonomia Atividades Textos Pesquisa Sobre
Uma aula = um problema individual, prático e único.
Metodologia para aulas:
práticas: Sempre o aluno precisa trabalhar os conceitos vistos na aula.
exigentes: O exercício de cada aula vale cerca de 5% da nota bimestral.
individuais: O exercício que o aluno tem que fazer é único. Ninguém mais na sala (nem no universo) tem aquele exercício.
divertidas: Não sei se exagero ao chamar uma aula de divertida, mas a falação do professor está limitada a 30 minutos, mais ou menos. Depois o aluno tem que por mãos à obra.
respeitosas: Cada exercício feito pelo aluno é corrigido e a ele devolvido no prazo de uma semana. A coleção de exercícios corrigidos sobra como memória dos assuntos estudados em sala de aula
flexíveis: A lista de exercícios prontos a serem usados nesta metodologia é de cerca de 600.
Meu assistente, de nome Travas, emitindo parecer sobre um exercício...
© 2023 por Nome do Site. Orgulhosamente criado com Wix.com


https://www.algoritmovivo.com/atividades
INÍCIO Taxonomia Atividades Textos Pesquisa
Varejão
Uma paródia sobre a história : Sonhos de Poder, de Isaac Asimov
Apresentação sobre Turing e a Enigma
Enigma programada em 3 linguagens (C, APL e Python)
Uma ótima ferramenta para treinar programação (estudar)
um resumo de Python (cart2_python)
Um resumo do pacote Numpy (cart2_numpy)
Um resumo do pacote Matplotlib (cart2_matplotlib)
Um resumo do pacote Sympy (cart2_sympy)
transparências SEATEL 2019
Programação de Computadores (CI180)
Turma do curso de Engenharia de bioprocessos
Página da disciplina
uma possível solução da consistência de datas
Primeira prova de 2014. Uma possível solução.
Segundo semestre de 2014. Possível solução
Provas de 2015. Uma solução.
Provas de 2016. Uma solução.
Provas de 2017. Uma solução.
Um texto sobre funções.
Prova 2: 2014s1, 2014s2, 2015 2017.
Métodos Numéricos (CI181), turma 19FCN
Turma do curso de Engenharia Elétrica
excelente material dos professores Ionildo e Diogenes métodos numéricos
uma publicação sobre Programação & Python
exercício 101 (enigma)
exercício 102 (datas)
exercício 103 (condicionais)


PACKSET
© Getty Images


UBER
ADVANCED
TECHNOLOGIES
CENTER

Por que Python é o cara ?

- Nasceu na hora certa
  - 2000
  - abundância de recursos
  - múltiplos modelos (copia do que é bom)
  - Infraestrutura de desenvolvimento global
- muito fácil de aprender
- resiliente

Python

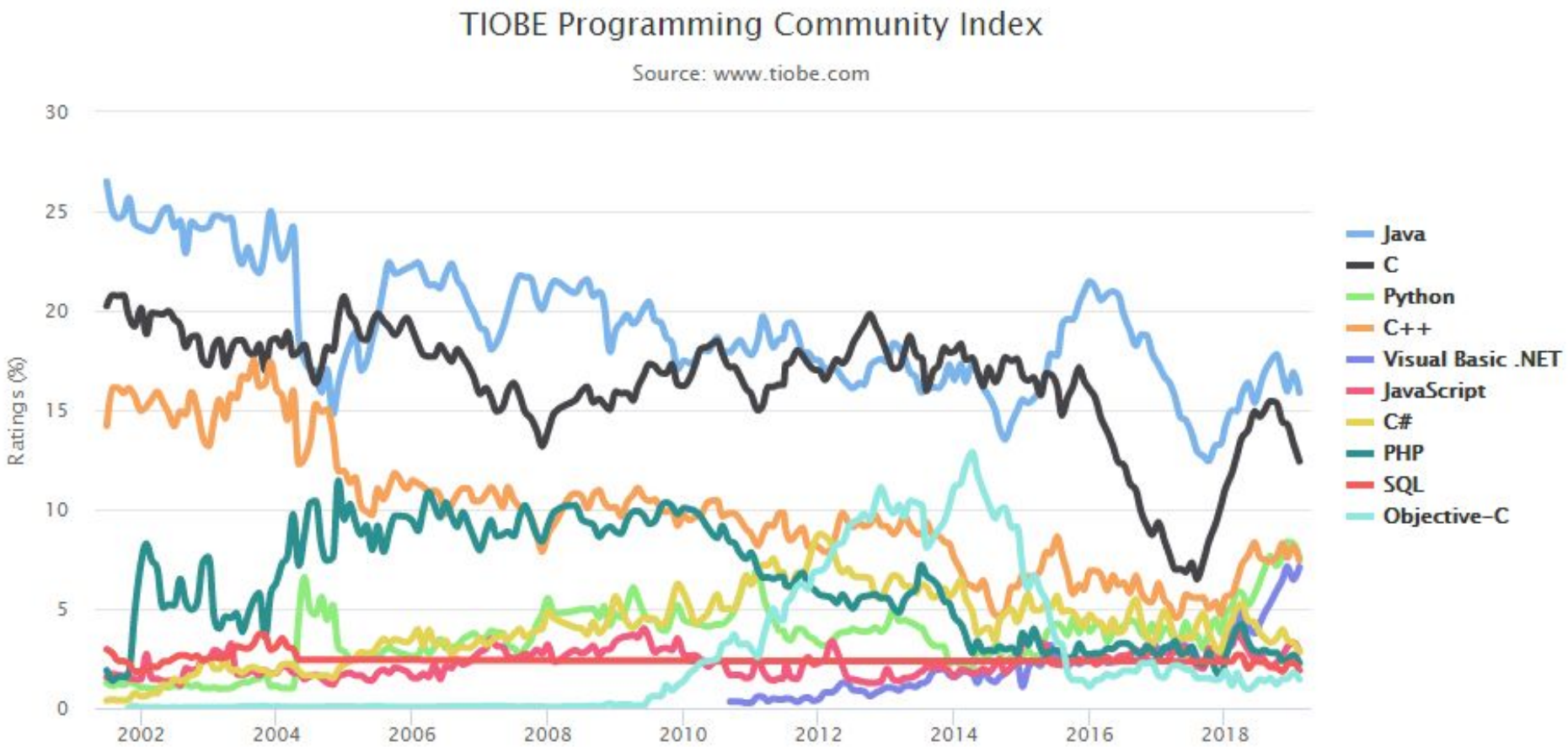
- Guido Van Rossum
- Python vem de Monthy Python

Monty Python

acompanhe este diálogo impagável do filme Monty Python

```
Primeiro aldeão: Achamos uma bruxa. Queimamos ela ?
Todos: Uma bruxa ! Queimem-na !
Bedevere: Por que pensam que é uma bruxa ?
Segundo aldeão: Porque ela me converteu em uma salamandra
Bedevere: Numa salamandra ?
Segundo aldeão (depois de observar-se um tempo): Já estou melhor.
Todos: De todo jeito, há que queimá-la.
Bedevere: Silêncio ! Silêncio !, há maneiras de saber se é bruxa.
Bedevere: Digam-me, O que vocês fazem às bruxas ?
Todos: As queimamos.
Bedevere: E o que mais queimam, além de bruxas ?
Quarto aldeão: ... madeira ?
Bedevere: Bom, então, porque ardem as bruxas ?
Segundo aldeão (em voz baixa):Porque estão feitas de madeira?
Bedevere: Mas, como saberemos se ela está feita de madeira ?
Primeiro aldeão: Construam uma ponte com ela.
Bedevere: Mas, por acaso não se pode fazer pontes de pedra ?
Todos: sim, claro, he, ... hum,...
Bedevere: A madeira afunda na água ?
Todos: não, não, ela flutua. Joguem-na no lago.
Bedevere: Esperem, esperem... O que mais flutua na água ?
Todos: Pão ?, não, não, não.      As maças, a salsa, pedrinhas pequenas.
Bedevere: não, não, não
O rei Arthur: um pato !    (Todos olham para Arthur. Bedevere parece impressionado)
Bedevere: Exatamente, pelo que, ... logicamente...
Primeiro aldeão (começando a pegar a corda): Se ela pesa o mesmo
      que um pato... então está feita de madeira.
Bedevere: E portanto ?
Todos: É uma bruxa !
```

TIOBE


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
Python
C++
Visual Basic .NET
JavaScript
C#
PHP
SQL
Objective-C

Mais do Tiobe

Mais para o passado:

Linguagem	17	12	07	02	97	92
Java	1	1	1	1	12	-
C	2	2	2	2	1	1
C++	3	3	3	3	2	2
C#	4	4	7	17	-	-
Python	5	7	6	11	27	-
VB.NET	6	19	-	-	-	-
PHP	7	6	4	5	-	-
JavaScript	8	9	8	8	19	-
COBOL	25	28	17	9	3	10
Lisp	32	12	14	12	9	5
Prolog	33	32	26	15	20	12
Pascal	106	15	19	97	8	3

Consulta em 07/2017

Porque Python

- Moderna, veja-se por exemplo
  - big nums; / e //; matemática complexa
  - tipagem só a nosso favor
- Chegou na hora exata para desenvolvimento global. (Github c/ 65Mproj)
- Multiplataforma: Unix, Windows, Apple, celular android, raspberry e outros
- Freeware
- Fácil de aprender e usar (Mainstream educativo).
  - escrita limpa: não há ruído
  - 27 das 40 maiores universidades nos EUA usam-no
- Pacotes, pacotes, pacotes
- Inteligência Artificial (DistBelief $\rightarrow$ Tensorflow)

TensorFlow (software library) Machine Learning

Which language should I use for tensorflow?

1 Answer

Nayeem Ahmad, former COO at CTrends (2017-2018)
Answered Mar 7 2018

Though **Python** is the language of choice for **TensorFlow**-client related programming, someone already comfortable with **Java/C/Go** shouldn't switch to Python at the beginning. Because the cost of switching will be pretty high. If you plan to completely focus on **TensorFlow** and not having other dependencies then spending some time to learn Python should be the choice.

Here is how their FAQ answers this question.

"Which client languages are supported in TensorFlow?

TensorFlow is designed to support multiple client languages. Currently, the best-supported client language is Python. Experimental interfaces for executing and constructing graphs are also available for C++, Java and Go.

TensorFlow also has a C-based client API to help build support for more client languages. We invite contributions of new language bindings.

Installing TensorFlow
Installing TensorFlow on Ubuntu
Installing TensorFlow on Mac OS X
Installing TensorFlow on Windows
Installing TensorFlow from Sources

Transitioning to TensorFlow 1.0

Installing TensorFlow for Java
Installing TensorFlow for Go
Installing TensorFlow for C

Installing TensorFlow

The following guides explain how to install a version of TensorFlow that enables you to write applications in Python:

- Installing TensorFlow on Ubuntu
- Installing TensorFlow on Mac OS X
- Installing TensorFlow on Windows
- Installing TensorFlow from Sources

Many aspects of the Python TensorFlow API changed from version 0.n to 1.0. The following guide explains how to migrate older TensorFlow applications to Version 1.0:

- Transitioning to TensorFlow 1.0

The following guides explain how to install TensorFlow libraries for use in other programming languages. These APIs are aimed at deploying TensorFlow models in applications and are not as extensive as the Python APIs.

- Installing TensorFlow for Java
- Installing TensorFlow for C
- Installing TensorFlow for Go

Por exemplo

```
>>> def fat(x):
        if x<2:
            return 1
        else:
            return x*fat(x-1)


>>> fat(10)

3628800
>>> fat(20)

2432902008176640000
>>> fat(30)

265252859812191058636308480000000
>>> fat(100)

933262154439441526816992388562667004907159682643816214685929638952
175999932299156089414639761565182862536979208272237582511852109168
640000000000000000000000000
>>>
```

```
>>> (-2)**0.5
(8.659560562354934e-17+1.4142135623730951j)
>>>
```

Comparando

Em C++

```
#include<iostream>
#include<cmath>
using namespace std;
float pita(float a, float b){
   return sqrt(pow(a,2)+pow(b,2));
   }
int main(){
   float ca,cb,hp;
   cout<<"Informe o cateto A: "<< endl;
   cin>>ca;
   cout<<"Informe o cateto B: "<< endl;
   cin>>cb;
   hp = pita(ca,cb);
   cout<<"A hipotenusa eh: "<<hp;
}
```

Em Python

```
def pita(a,b):
  return (a**2+b**2)**0.5
ca=float(input("Informe o cateto A: "))
cb=float(input("Informe o cateto B: "))
hp=pita(ca,cb)
print("A hipotenusa é: ",hp)
```

Versões

Python 3.7.3	liberado em 25 março 2019
Python 3.7	liberado em 27 junho 2018
Python 3.6.5	liberado em 28 Março 2018
Python 3.6.4	19 Dezembro 2017
Python 3.6.0	23 Dezembro 2016
Python 3.5.1	07 Dezembro 2015
Python 3.3.7	19 Setembro 2017
Python 3.2.6	11 Outubro 2014
Python 3.0	3 Dezembro 2008
Python 2.7.8	1 Julho 2014
Python 2.6.2	14 Abril 2009
Python 2.1.3	8 Abril 2002
Python 1.4	25 Outubro 1996

Começando

```
>>> a = int(input("Informe a quantidade de laranjas: "))
Informe a quantidade de laranjas: 3
>>> pt = 0.76 * a
>>> print("Total: ",pt)
Total: 2.28000000000002

---------------------- OU ----------------------

def custo():
    a = int(input("Informe a quantidade de laranjas: "))
    pt = 0.76 * a
    print("Total: ",pt)
custo()
```

For é o grande interador

O while e o for podem ser usados mais ou menos como em C++, mas...

um interador é uma generalização do conceito de índice. Ele é uma generalização já que o indexador só se aplica a conjuntos homogêneos indexados (conhecidos como arrays) e o interador se aplica a qualquer coleção de objetos Python. Pode ser um array, mas pode ser também registros em um arquivo, itens em um conjunto, em um dicionário etc etc. Veja um exemplo radical disso

```
s="Curitiba - Paraná - Brasil"
qtd=0
for c in s:
  if c=="a":
    qtd=qtd+1
print("achei ",qtd," a")
```

Listas

O grande apelo de Python

```
>>> variav=66
>>> a=[1,4,"laranja","batata",variav,3,4]
>>> a
[1, 4, 'laranja', 'batata', 66, 3, 4]
>>> a[2:4]
['laranja', 'batata']
>>> a[4:2:-1]
[66, 'batata']
>>> quadrados = [x**2 for x in range(10)]
>>> quadrados
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> qi = [x for x in quadrados if x%2==0]
>>> qi
[0, 4, 16, 36, 64]
```

Conjuntos

```
>>> a = set('abracadabra') ou
    a = {'a','b','r','a','c','a','d','a','b','r','a'}
>>> b = set('alacazam')
>>> a # letras unicas em a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b # letras em a mas não em b
set(['r', 'd', 'b'])
>>> a | b # letras em a ou em b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b # letras tanto em a como em b
set(['a', 'c'])
>>> a ^ b # letras em a ou b mas não em ambos
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Dicionários

```
>>> d={}
>>> d[123]='oba' #a chave 123 corresponde a 'oba'
>>> d
{123: 'oba'}
>>> d[124]='fui' #a chave 124 a 'fui'
>>> d
{123: 'oba', 124: 'fui'}
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

Um exemplo original

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
... print 'What is your {0}? It is {1}.'.format(q, a)
...
What is your name? It is lancelot. What is your quest?
It is the holy grail. What is your favorite color?
It is blue.
```

O exemplo acima reproduz um diálogo do filme Monty Python - Em Busca do Cálice Sagrado. Este trecho não pode ser traduzido, exceto por um decreto real

Mais

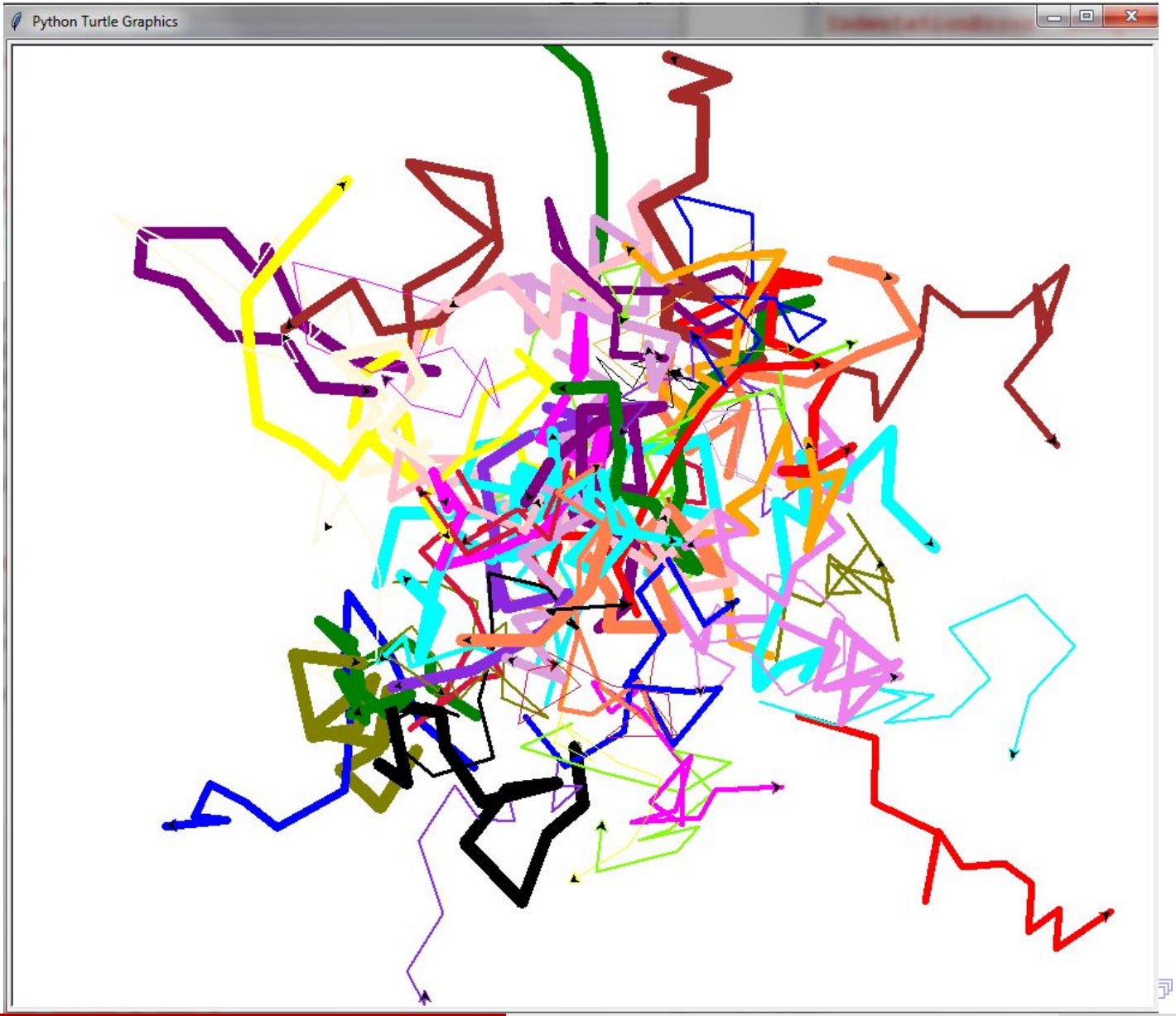
```
>>> cesta = ['uva', 'laranja', 'uva',
    'abacaxi', 'laranja', 'banana']
>>> for fruta in sorted(cesta):
... print fruta
...
abacaxi  banana  laranja  laranja  uva  uva
           (deitado para caber...)
>>> for fruta in sorted(set(cesta)): # sem duplicações
... print fruta
...
abacaxi  banana  laranja  uva
           (deitado para caber...)
```

Objetos

Difícil apreender, mas insuperável quando necessário. Vejamos um exemplo: simular um formigueiro com 1200 formigas

- Uma formiga é simples: deixa rastro, anda x unidades na direção y
- O problema está em compatibilizar as 1200 formigas
- Usando objetos, define-se a classe formiga
- instancia-se 1200 vezes a classe.

Um formigueiro com 60 formigas



Arquivos

```
>>> f = open('/tmp/workfile', 'r')
>>> f.read()
'Texto completo do arquivo.\n'
>>> f.read()
''
>>> f.readline()
'Primeira linha do arquivo.\n'
>>> f.readline()
'Segunda linha do arquivo.\n'
>>> f.readline()
''
>>> f.readlines()
['Primeira linha do arquivo.\n', 'Segunda linha do arquivo.\n']
>>> for line in f:
      print line,
Primeira linha do arquivo.
Segunda linha do arquivo.
```

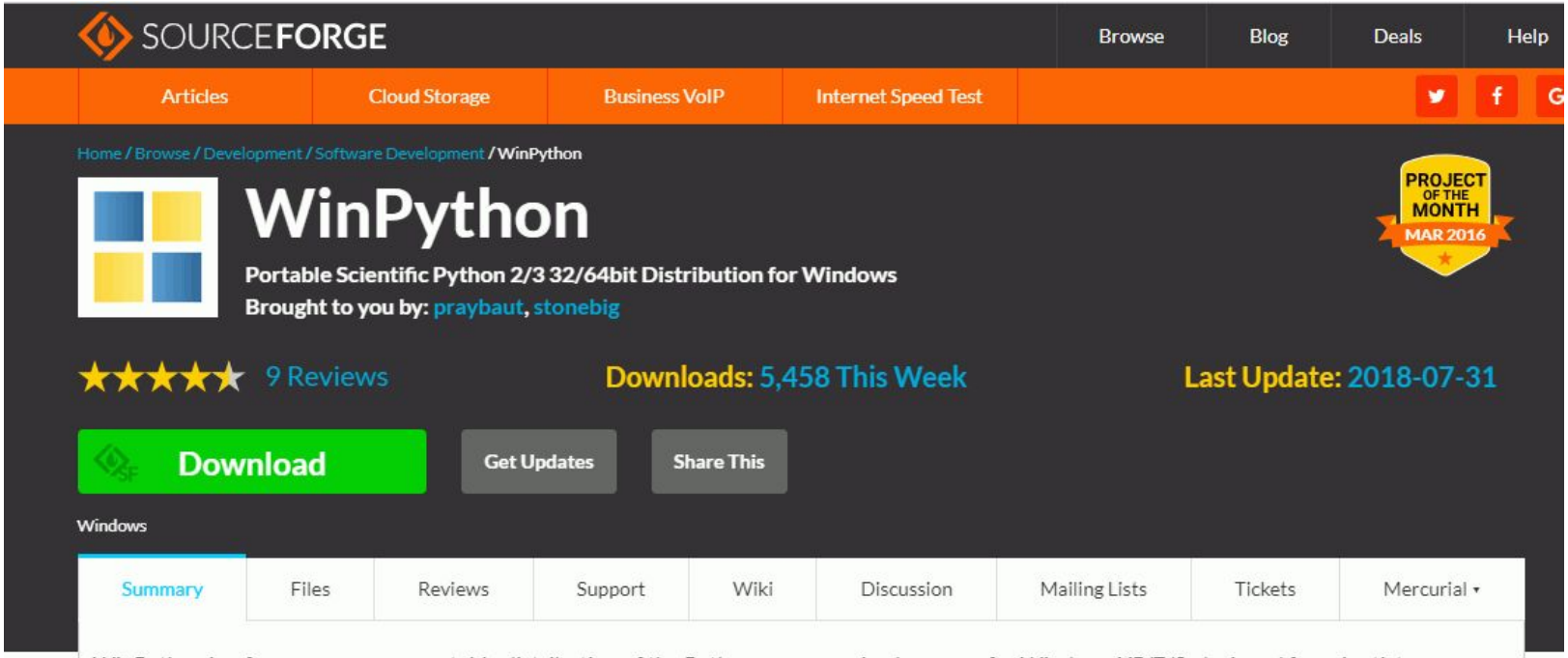
Pacotes

- Numpy
- Sympy
- Matplotlib
- Astropy
- Django
- OpenCV
- Tensorflow
- Pygame
- Itertools
- Pandas
- Qrcode

O pacote Numpy

- Ferramenta para manipulação numérica em Python
- Implementa ndarray e tudo o que está em volta dela [ex.: print(A)]
- Manual tem 1.542 páginas A4
- Pré-requisito em centenas de pacotes Python (Tensor Flow, Matplotlib, Sympy, PyEasyGA, Astropy, OpenCV, Pandas, SciPy...)
- Para instalar: no diretório /scripts execute pip install numpy (precisa acesso à Internet)
- Já instalado no winpython (https://sourceforge.net/projects/winpython/)

Winpython



SOURCEFORGE

Browse | Blog | Deals | Help

Articles | Cloud Storage | Business VoIP | Internet Speed Test

Home / Browse / Development / Software Development / WinPython

PROJECT OF THE MONTH MAR 2016

WinPython

Portable Scientific Python 2/3 32/64bit Distribution for Windows

Brought to you by: praybaut, stonebig

★★★★★ 9 Reviews

Downloads: 5,458 This Week

Last Update: 2018-07-31

Download | Get Updates | Share This

Windows

Summary | Files | Reviews | Support | Wiki | Discussion | Mailing Lists | Tickets | Mercurial

WinPython is a free open-source portable distribution of the Python programming language for Windows XP/7/8, designed for scientists, supporting both 32bit and 64bit versions of Python 2 and Python 3.

Since September 2014, Developpement has moved to https://winpython.github.io/

Features

- Designed for regular scientific users: interactive data processing and visualization using Python with Spyder
- Designed for advanced scientific users and software developers: Python applications development with Spyder, version control with Mercurial and other development tools (like gettext, etc.)
- Portable: preconfigured, it should run out of the box on any machine under Windows (without any requirement) and the folder containing WinPython can be moved to any location (local, network or removable drive) with most of the application settings
- Flexible: one can install (or should I write "use" as it's portable) as many WinPython versions as necessary (like isolated and self-consistent environments), even if those
- Customizable: The integrated package manager (wppm, as WinPython Package Manager) helps installing, uninstalling or upgrading Python packages
- As WPPM may not support some packages, it's also possible to install or upgrade packages using easy_install or pip from the WinPython command prompt
- A configuration file allows to set environment

Uso

```
>>> import numpy as np
>>> a = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> a
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> b = np.zeros((3,4),float)
>>> b
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
>>> c = np.array([[1,2],[3,4]],complex)
>>> c
array([[1.+0.j, 2.+0.j],
       [3.+0.j, 4.+0.j]])
```

Daí

- ravel - retorna um vetor com os elementos
- reshape - muda a forma mantendo os elementos
- resize - recorta o array
- sort - ordena o array
- sum, std, mean, cumsum - soma, acha d.p., max, min, etc
- swapaxes, take - reorganiza os elementos
- tofile, tolist, tostring - transforma o array
- transpose - devolve a transposta
- ... , além de centenas de outros métodos (veja a documentação)

```
Python é gentil... (indexação)
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> a[1][1]
5
```

```
>>> a[1,1]  #so em np
5
>>> a[(1,1)] #idem
5
```

Comparando com C

```
Em C-------------------------
#include<iostream>
using namespace std;
main() {
   int a[2][3]={1,2,3,4,5,6};
   int i,j,soma;
   soma=0;
   for (i=0;i<2;i++){
      for(j=0;j<3;j++){
         soma=soma+a[i][j];
      }
   }
   cout<<soma;
}
```

```
Em Python------------
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
print(a.sum()) #--ou--
print(sum(a.ravel()))
--------- ou ---------
a=[[1,2,3],[4,5,6]]
i=0
s=0
while i<2:
   j=0
   while j<3:
      s=s+a[i][j]
      j=j+1
   i=i+1
print(s)
```

Com e sem numpy

```
>>> import numpy as np
>>> a=[1,2,30]
>>> a+[11]
[1, 2, 30, 11]
>>> a.append(11)
>>> a
[1, 2, 30, 11]
>>> b=np.array([1,2,30])
>>> b+1
array([ 2,  3, 31])
>>> b
array([ 1,  2, 30])
>>> np.append(b,11)
array([ 2,  3, 31, 11])
```

Um exemplo de soma usando axis

```
>>> x = array([[[ 0, 1, 2],[ 3, 4, 5],[ 6, 7, 8]],
[[ 9, 10, 11],[12, 13, 14],[15, 16, 17]],
[[18, 19, 20],[21, 22, 23],[24, 25, 26]]])
>>> x.sum(axis=0)
array([[27, 30, 33],
[36, 39, 42],
[45, 48, 51]])
>>> # for sum, axis is the first keyword, so we may omit it,
>>> # specifying only its value
>>> x.sum(0), x.sum(1), x.sum(2)
(array([[27, 30, 33],
[36, 39, 42],
[45, 48, 51]]),
array([[ 9, 12, 15],
[36, 39, 42],
[63, 66, 69]]),
```

Indexação

```
>>> x = np.array([[[1],[2],[3]], [[4],[5],[6]]])
>>> x.shape
(2, 3, 1)
>>> x[1:2]
array([[[4],
        [5],
        [6]]])

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

Mais exemplos

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

Concatenação

```
>>> a = np.array([[1, 2], [3, 4]], float)
>>> b = np.array([[5, 6], [7,8]], float)
>>> np.concatenate((a,b))
array([[ 1., 2.],
[ 3., 4.],
[ 5., 6.],
[ 7., 8.]])
>>> np.concatenate((a,b), axis=0)
array([[ 1., 2.],
[ 3., 4.],
[ 5., 6.],
[ 7., 8.]])
>>> np.concatenate((a,b), axis=1)
array([[ 1., 2., 5., 6.],
[ 3., 4., 7., 8.]])
```

Matemática entre arrays

```
>>> a = np.array([1,2,3], float)
>>> b = np.array([5,2,6], float)
>>> a + b
array([6., 4., 9.])
>>> a - b
array([-4., 0., -3.])
>>> a * b
array([5., 4., 18.])
>>> b / a
array([5., 1., 2.])
>>> a % b
array([1., 0., 3.])
>>> b**a
array([5., 4., 216.])

>>> a = np.array([1, 4, 9], float)
```

Interadores

```
>>> a = np.array([1, 4, 5], int)
>>> for x in a:
... print x
... <hit return>
1
4
5
```

Mais

```
>>> a = np.array([2, 1, 9], float)
>>> a.mean()
4.0
>>> a.var()
12.666666666666666
>>> a.std()
3.559026084010437 1
>>> a = np.array([2, 1, 9], float)
>>> a.min()
1.0
>>> a.max()
9.0
>>> a = np.array([2, 1, 9], float)
>>> a.argmin()
1
>>> a.argmax()
```

Manipulações booleanas

```
>>> a = np.array([1, 3, 0], float)
>>> np.logical_and(a > 0, a < 3)
array([ True, False, False], dtype=bool)
>>> b = np.array([True, False, True], bool)
>>> np.logical_not(b)
array([False, True, False], dtype=bool)
>>> c = np.array([False, True, False], bool)
>>> np.logical_or(b, c)
array([ True, True, False], dtype=bool)
```

Multiplicação matricial

```
>>> a = np.array([[0, 1], [2, 3]], float)
>>> b = np.array([2, 3], float)
>>> c = np.array([[1, 1], [4, 0]], float)
>>> a
array([[ 0., 1.],
       [ 2., 3.]])
>>> np.dot(b, a)
array([ 6., 11.])
>>> np.dot(a, b)
array([ 3., 13.])
>>> np.dot(a, c)
array([[ 4., 0.],
       [ 14., 2.]])
>>> np.dot(c, a)
array([[ 2., 4.],
       [ 0., 4.]])
```

Subpacotes

Lista de subpacotes que ajudam na programação:

subpacote	Objetivo
core	objetos básicos
lib	utilidades adicionais
linalg	álgebra linear
fft	transformada discreta de Fourier
random	geração de números randômicos
distutils	distribuição de probabilidades
testing	funções de teste
f2py	tradução de código fortran

Determinante

```
>>> a = np.array([[4, 2, 0], [9, 3, 7], [1, 2, 1]], float)
>>> a
array([[ 4., 2., 0.],
       [ 9., 3., 7.],
       [ 1., 2., 1.]])
>>> np.linalg.det(a)
-53.999999999999993
```

A inversa

```
>>> b = np.linalg.inv(a)
>>> b
array([[ 0.14814815, 0.07407407, -0.25925926],
       [ 0.2037037 , -0.14814815, 0.51851852],
       [-0.27777778, 0.11111111, 0.11111111]])
>>> np.dot(a, b)
array([[ 1.00000000e+00, 5.55111512e-17, 2.22044605e-16],
       [ 0.00000000e+00, 1.00000000e+00, 5.55111512e-16],
       [ 1.11022302e-16, 0.00000000e+00, 1.00000000e+00]])
```

Randômicos

```
>>> np.random.rand(2,3)
array([[ 0.50431753, 0.48272463, 0.45811345],
       [ 0.18209476, 0.48631022, 0.49590404]])
>>> np.random.rand(6).reshape((2,3))
array([[ 0.72915152, 0.59423848, 0.25644881],
       [ 0.75965311, 0.52151819, 0.60084796]])

>>> l = range(10)
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> np.random.shuffle(l)
>>> l
[4, 9, 5, 0, 2, 7, 6, 8, 1, 3]
```

Sistemas Lineares: solve e lstsq

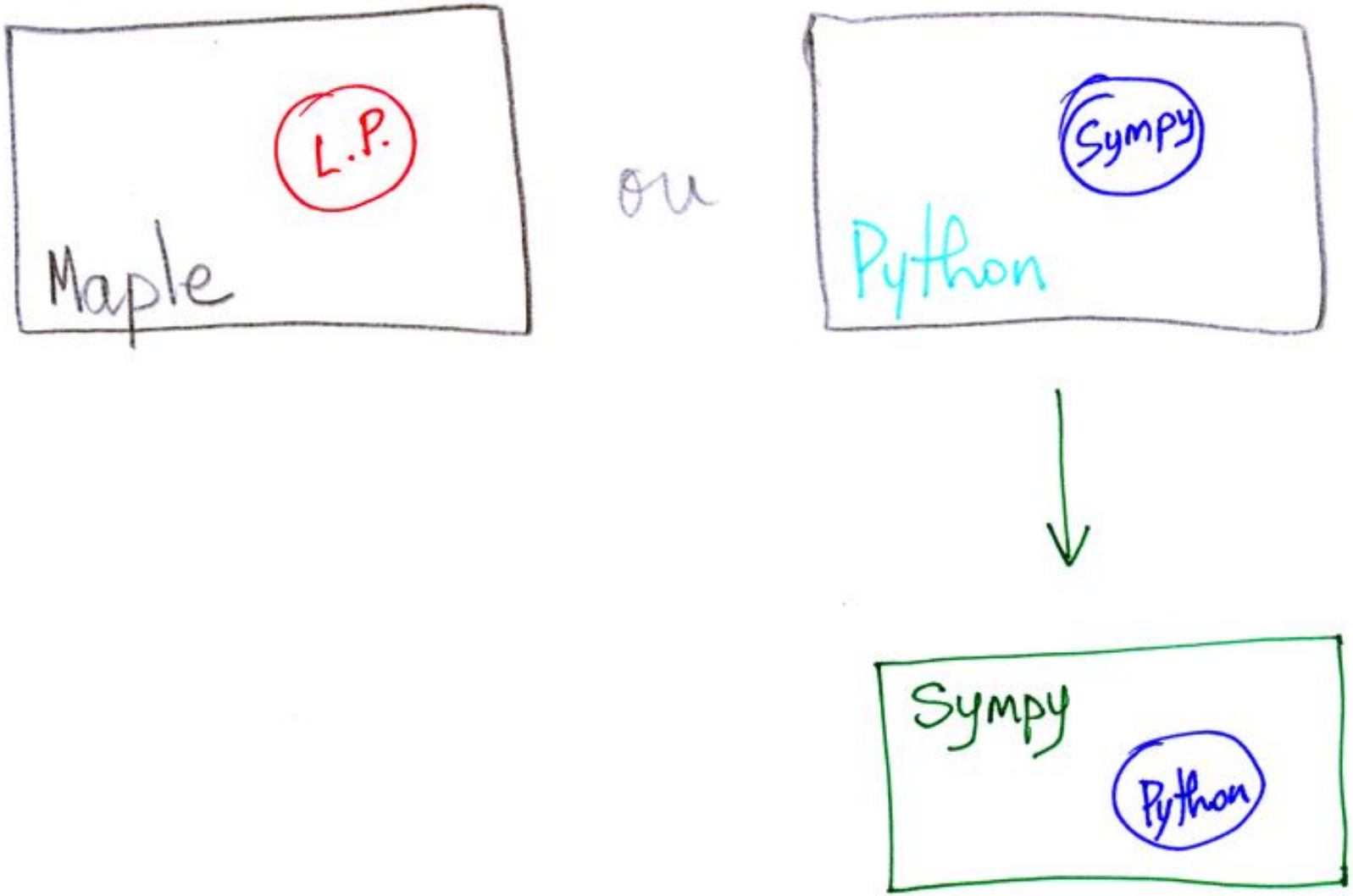
```
>>> a = np.array([[3,1], [1,2]])
>>> b = np.array([9,8])
>>> x = np.linalg.solve(a, b)
>>> x
array([ 2., 3.])

Neste caso: 170 equações a 64 incógnitas:
>>> c=np.linalg.lstsq(a,b)
```

SymPy

Uma ferramenta para manipular a matemática de maneira simbólica.

- Fácil
- Poderosa
- Sob Python
  - junto com tudo o que se conhece de Python
  - grátis
  - sair usando
- concorre com Maple (1000 US$)
- novo: projetado em 2008, lançado em 2013. Maple é dos 80's

SymPy



SymPy

SymPy Documentation

Release 1.1.1

O que é manipulação simbólica

- computação numérica: sqrt(8) é igual a 2.82842712474619
- computação simbólica: sqrt(8) é igual a 2*sqrt(2)

A vantagem desta última é que, nela, sqrt(8).evalf() é 2.82842712474619.
Ou sqrt(8).n(30) com 30 casas ou 2.82842712474619009760337744842.

Instalação e Uso

pip install sympy (no diretório scripts)

inúmeras dependências (verificadas e satisfeitas automaticamente)

import sympy as sp

uso do winpython (já está instalado) em winpython.sourceforge.net

Versão atual: pip show sympy
Atualizar: pip install –upgrade sympy
Dentro do Python

```
>>> import sympy
>>> sympy.__version__
'1.4'
```

Variáveis simbólicas

Uma variável simbolica recebe expressões e resolve essas expressões algebricamente e não computacionalmente. Isto só ao final.

```
>>> x=sp.symbols('x')
>>> x=(x-5)*(x-2)*(x-8)
>>> x
(x - 8)*(x - 5)*(x - 2)
>>> x.expand()
x**3 - 15*x**2 + 66*x - 80
```

Alguns exemplos

```
from sympy import symbols
>>> x,y = symbols('x y')
>>> expr = x+2*y
>>> expr
x + 2*y
>>> expr + 1
x + 2*y + 1
>>> expr - x
2*y
>>> x*expr
x*(x + 2*y)
>>> expand(x*expr)
x**2 + 2*x*y
>>> factor(x**2 + 2*x*y)
x*(x + 2*y)
```

O que dá para fazer

O sympy pode fazer todo o tipo de computação simbólica, como simplificar, calcular derivadas, integrais e limites, resolver equações, trabalhar com matrizes, e muito mais, tudo de maneira simbólica. Ele inclui módulos para plotagem, impressão, geração de código, física, estatística, combinatória, teoria dos números, geometria, lógica, etc.

A documentação do produto em maio/2018 tem 1.878 páginas (SymPy Documentation (Release 1 - SymPy Development Team_6282).

Em abril de 2019, na versão 1.4, a documentação tem 2.127 páginas.

Comandos...

- sympify

```
>>> x=sp.S('1/7')
>>> x
1/7
```

- evalf ou n() ou n(tam) = avalia numericamente

```
x,y,z=symbols('x,y,z')
i,j,k=symbols('i,j,k',integer=True)
f,g,h=symbols('f,g,h',cls=Function)
```

Substituição

```
>>> expr = cos(x)+1
>>> expr
cos(x) + 1
>>> expr.subs(x,pi/2)
1
>>> (x**2+2*x+10).subs(x,11)
153
>>> (x**2+2*x+10).subs(x,y**2)
 4      2
y  + 2.y  + 10
```

Simplificação

```
>>> simplify(sin(x)**2+cos(x)**2)
1
>>> simplify((x**3+x**2-x-1)/(x-1))
x**2 + 2*x + 1
>>> simplify(x**2+2*x+1)
x**2 + 2*x + 1
>>> expand((x+1)**3)
x**3 + 3*x**2 + 3*x + 1
>>> factor(x**3+3*x**2+3*x+1)
(x + 1)**3
>>> expand((cos(x)+sin(x))**2)
sin(x)**2 + 2*sin(x)*cos(x) + cos(x)**2
```

collect(), cancel() e apart()

```
>>> collect(x*y+x-3+2*x**2-z*x**2+x**3,x)
x**3  + x**2 *(2 - z) + x*(y + 1) - 3
>>> e=symbols('e')
>>> e=1/x+(3*x/2-2)/(x-4)
>>> cancel(e)
```

$$\frac{3.x^2 - 2.x - 8}{2.x^2 - 8.x}$$

```
>>> e=(x**3+x**2+1)/(x**4+x**2)
```

$$\frac{x}{x^2+1} + \frac{1}{x^2}$$

Trigonometria

```
>>> trigsimp(sinh(x)/tanh(x))
cosh(x)
>>> expand_trig(sin(x+y))
sin(x)*cos(y) + sin(y)*cos(x)
>>> expand_trig(tan(2*x))
```

$$\frac{2.tan(x)}{1 - tan(x)^2}$$

Potência

1. $x^a x^b = x^{a+b}$
2. $x^a y^a = (xy)^a$ só é verdade quando $x, y \geq 0$ e $a \in \mathbb{R}$
3. $(x^a)^b = x^{ab}$ só é verdade quando $b \in \mathbb{Z}$

Funções: powsimp(), expand_power_exp(), expand_power_base(), powdenest()...

Logaritmos

1. $\log(xy) = \log(x) + \log(y)$
2. $\log(x^n) = n.\log(x)$

Funções: expand_log(), logcombine(), factorial(), binomial(), gamma(), ...

derivadas

```
>>> diff(exp(x**2),x)
```

$$2.x.e^{x^2}$$

```
>>> diff(tan(x),x)
   2
tan (x) + 1
>>> diff(x**4,x,x,x)
24.x
>>> diff(x**4,x,3)
24.x
>>> e=exp(x)+sin(x)
>>> diff(e,x)
```

$$e^x + cos(x)$$

```
>>> Derivative(e)
```

$$\frac{d}{\cdot} e^x + sin(x)$$

Integração

```
>>> integrate(cos(x))
sin(x)
>>> integrate(exp(x),(x,0,oo))
oo
>>> integrate(exp(-x),(x,0,oo))
1
>>> integrate(exp(-x**2-y**2),
    (x,-oo,oo),(y,-oo,oo))
pi
>>> e=log(x)**2
>>> Integral(e,x)
```

$$\int \log(x)^2 dx$$

```
>>> Integral(e,x).doit()
     2
```

Limites e séries

Seja o limite $lim_{x \to x_0} f(x)$

```
>>> limit(sin(x)/x,x,0)
1
```

$$\lim_{x \to 0+} \frac{cos(x) - 1}{x}$$

```
>>> Limit((cos(x)-1)/x,x,0).doit()
0
```

```
>>> exp(sin(x)).series(x,0,4)
```

$$1 + x + \frac{x^2}{2} + O(x^4)$$

Solução de equações

```
>>> solveset(x**2-x,x)
{0, 1}
>>> solveset(sin(x)-1,x,domain=Reals)
```

$2.n.\pi + \frac{\pi}{2} / \in \mathbb{Z}$

Seja o sistema: $2x + 3y - z = 4$, $x + y + z = 7$ e $x - 2y + 3z = 9$

```
>>> linsolve([2*x+3*y-z-4,x+y+z-7,x-2*y+3*z-9],(x,y,z))
{(1, 2, 4)}
>>> linsolve(Matrix([[2,3,-1,4],
    [1,1,1,7],[1,-2,3,9]]),(x,y,z))
{(1, 2, 4)}
>>> nonlinsolve([x**2+y-z+1,2*x+y+z-8,x-2*y+3*z-9],(x,y,z))
{(-1, 4, 6), (1, 2, 4)}
```

Equações diferenciais

```
>>> f,g=symbols('f,g',cls=Function)
>>> f(x).diff(x)
d
--(f(x))
dx
```

Supondo a equação diferencial $f''(x) - 2f'(x) + f(x) = sin(x)$ deve-se fazer

```
>>> dx=(f(x).diff(x,x)-
    2*f(x).diff(x)+f(x)-sin(x))
>>> dx
```

$$f(x) - sin(x) - 2.\frac{d}{dx}(f(x)) + \frac{d^2}{dx^2}(f(x))$$

```
>>> dsolve(dx,f(x))
```

$$f(x) = (C_1 + C_2.x).e^x + \frac{cos(x)}{2}$$

Matrizes

```
>>> M=Matrix([[1,2,3],[1,2,1],[1,0,6]])
>>> M
[1  2  3]
[1  2  1]
[1  0  6]
>>> M**-1
[-3    3     1  ]
[5/4  -3/4  -1/2]
[1/2  -1/2   0  ]
>>> M.det()
-4
>>> N=Matrix([[2,3,-1,4],[1,1,1,7],
    [1,-2,3,9]])
>>> N
[2  3   -1  4]
[1  1   1   7]
```

... Don't worry, we'll fix the math-fobia thing right up for you; when you've got SymPy skills, math fears you!.

Não se preocupe, vamos consertar a coisa da matemática para você; Quando você adquirir as habilidades "SymPy", a matemática é que vai ter medo de você!

MatPlotLib

Uma ferramenta para apresentar dados

- Fácil
- Poderosa
- Sob Python
  - junto com tudo o que se conhece de Python
  - grátis
  - sair usando

Instalação e Uso

pip install matplotlib (no diretório scripts)
inúmeras dependências (verificadas e satisfeitas automaticamente)
import matplotlib.pyplot as plt
uso do winpython (já está instalado) em winpython.sourceforge.net

2 modos de uso

- interface procedural: cada tarefa um comando (método)
  - Neste jeito se chamar o método sem parâmetros $\Rightarrow$ pergunta
  - Se chamar com parâmetros $\rightarrow$ determinar ordens
- via API, via objetos
- Cada tarefa tem 2 métodos:
  - xxx_get: pergunta
  - xxx_set: ordem

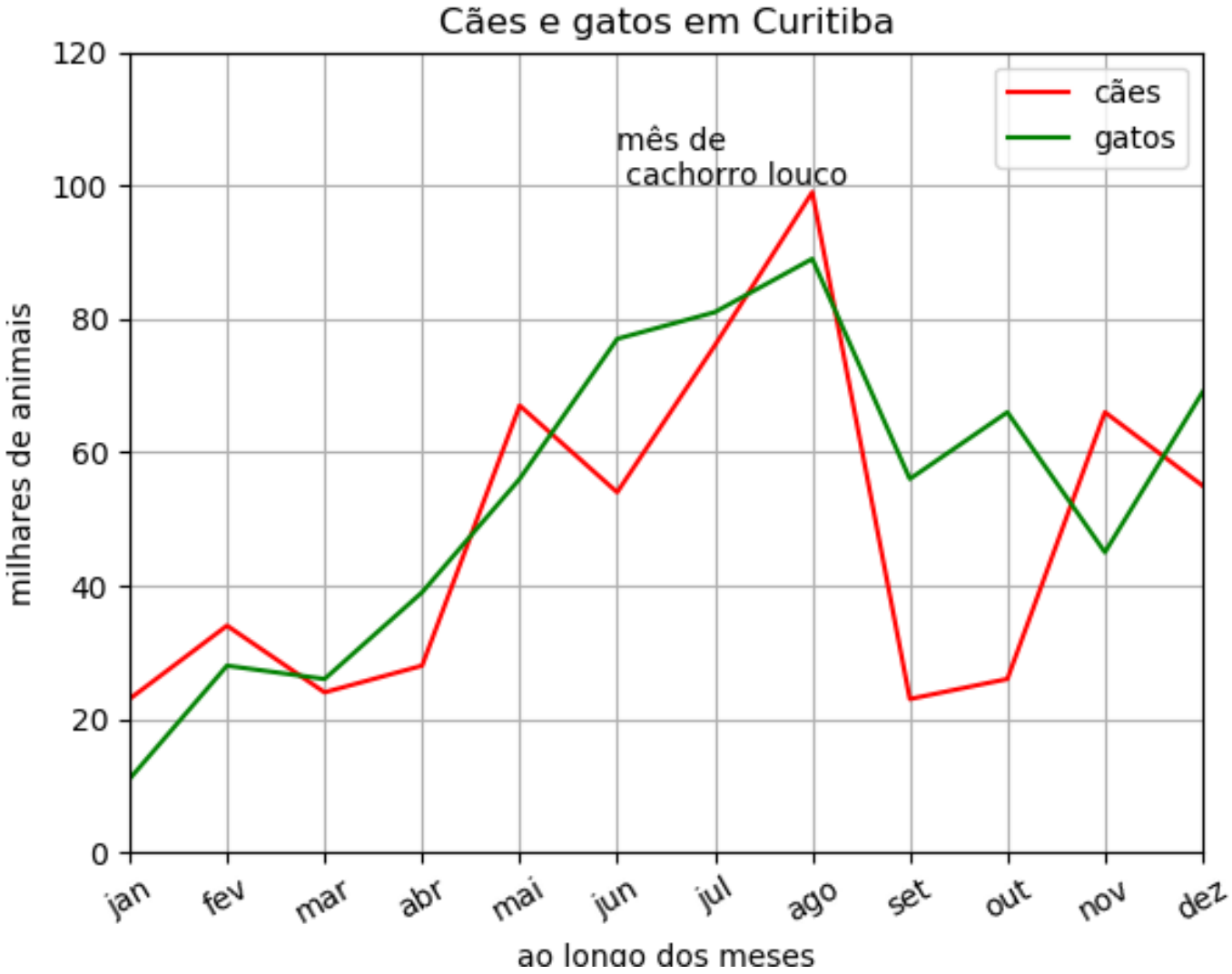
Um exemplo

```
import matplotlib.pyplot as plt ; import numpy as np
def caogato():
 ca=[23,34,24,28,67,54,76,99,23,26,66,55]
 ga=[11,28,26,39,56,77,81,89,56,66,45,69]
 plt.plot(ca,'r-',label='cães')
 plt.plot(ga,'g-',label='gatos')
 plt.legend()
 plt.title('Cães e gatos em Curitiba')
 plt.xlabel('ao longo dos meses')
 plt.ylabel('milhares de animais')
 plt.axis([0,11,0,120])
 plt.xticks(np.arange(12),('jan','fev','mar',
 'abr','mai','jun','jul','ago','set','out',
 'nov','dez'),rotation=30)
 plt.text(5,100,'mês de\n cachorro louco')
 plt.grid()
 plt.savefig('c:/p/python/caogato.png')
 plt.show()
caogato()
```

O mesmo, via API

```
import matplotlib.pyplot as plt; import numpy as np
def caogato2():
  ca=[23,34,24,28,67,54,76,99,23,26,66,55]
  ga=[11,28,26,39,56,77,81,89,56,66,45,69]
  fig,axe=plt.subplots()
  axe.plot(ca,'r-',label='cães')
  axe.plot(ga,'g-',label='gatos')
  axe.legend()
  axe.set_title('Cães e gatos em Curitiba')
  axe.set_xlabel('ao longo dos meses'); axe.set_ylabel('milhares de animais
  axe.axis([0,11,0,120])
  axe.set_xticks(range(12))
  axe.set_xticklabels(['jan','fev','mar',
  'abr','mai','jun','jul','ago','set','out','nov','dez'])
  axe.grid()
  axe.text(5,100,'mes de \ncachorro louco',rotation=45)
  plt.savefig('c:/p/python/caogato2.png')
  plt.show()
caogato2()
```

Resultou em

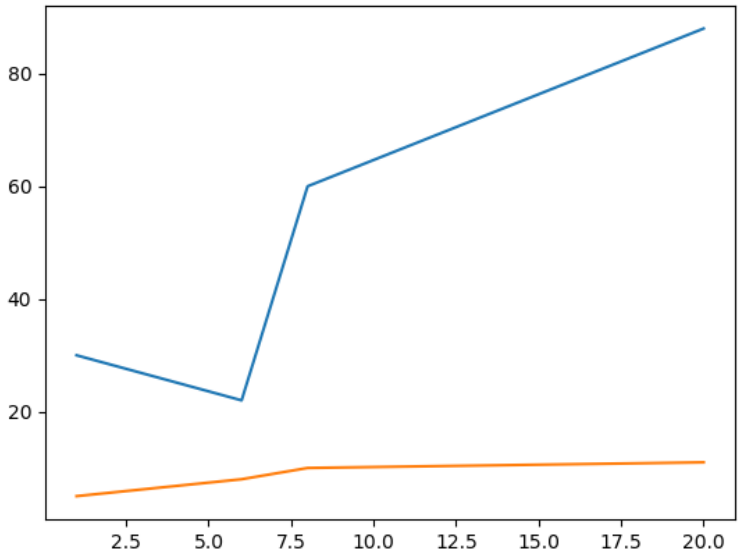

Cães e gatos em Curitiba
cães
gatos
mês de
cachorro louco
milhares de animais
120
100
80
60
40
20
0
jan
fev
mar
abr
mai
jun
jul
ago
set
out
nov
dez
ao longo dos meses

Plot

O comando mais simples:

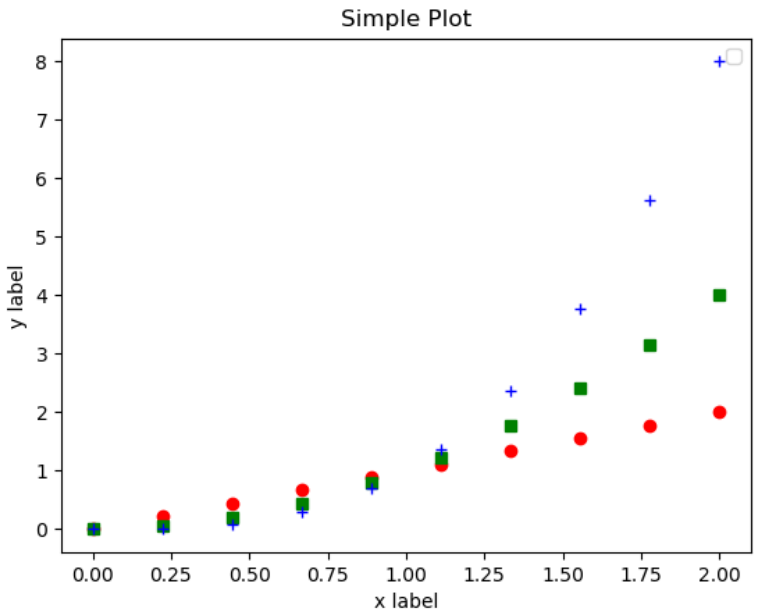
- valores de y (com x=índice de y)
- valores de (x,y)
- pode juntar quantos plots quiser

```
a=[1,6,8,20];b=[30,22,60,88];c=[5,8,10,11]
plt.plot(a,b);plt.plot(a,c);plt.show()
```



Outro exemplo

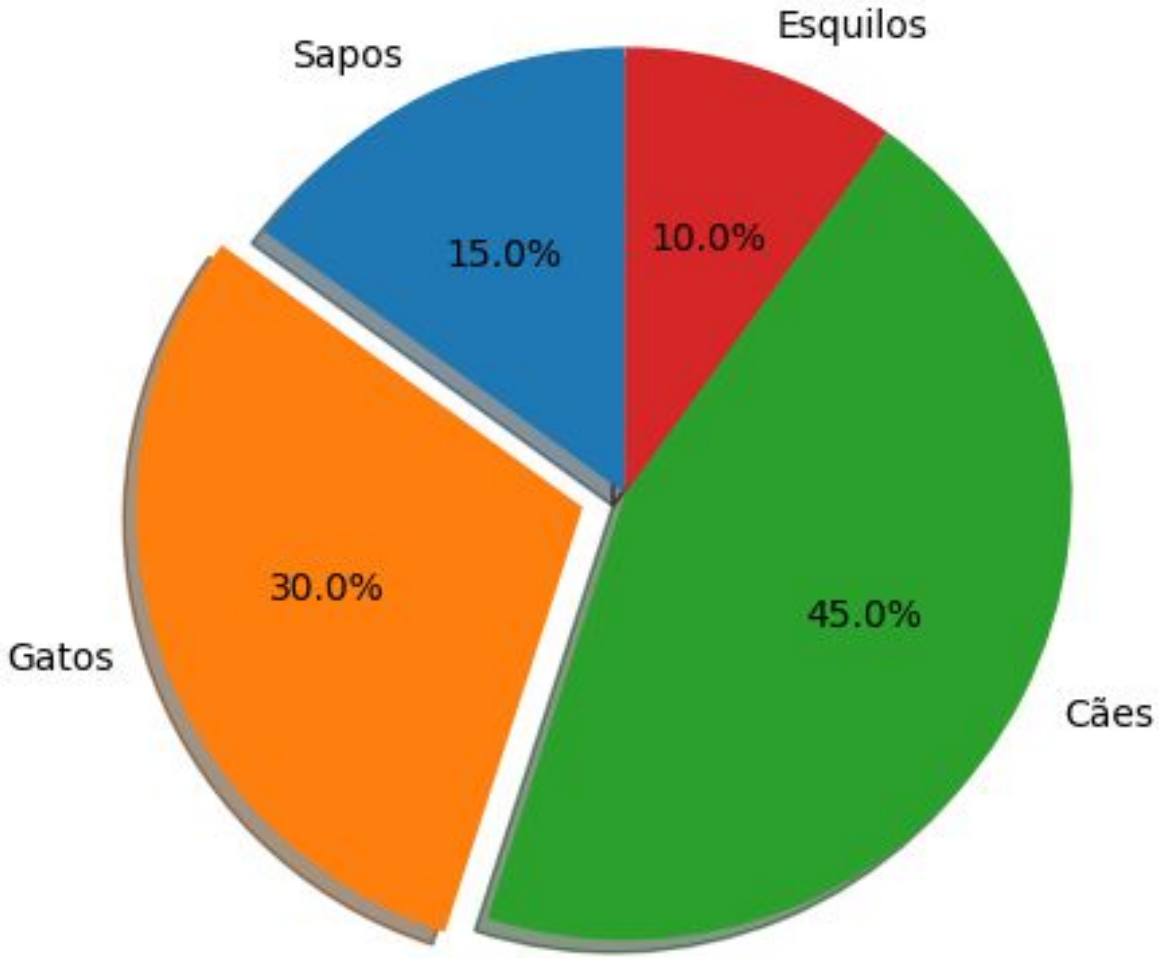
```
x = np.linspace(0, 2, 10) # pontos espaçados
plt.plot(x, x,'ro'); plt.plot(x, x**2,'gs')
plt.plot(x, x**3,'b+'); plt.xlabel('x label')
plt.ylabel('y label'); plt.title("Simple Plot")
plt.legend(); plt.show()
```



Pizzas

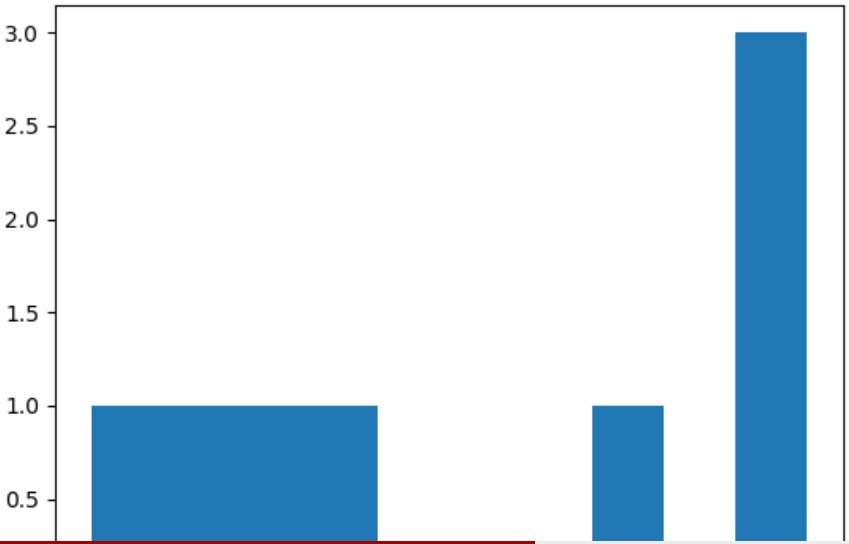
```
import matplotlib.pyplot as plt
# Pie chart, as fatias serão
ordenados e desenhados em anti-horário
labels = 'Sapos', 'Gatos', 'Cães', 'Esquilos'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)
    # explode a segunda fatia (i.e. 'Gatos')
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels,
    autopct='%1.1f%%',shadow=True, startangle=90)
ax1.axis('equal')  # desenha um círculo
plt.show()
```

Deu como resultado

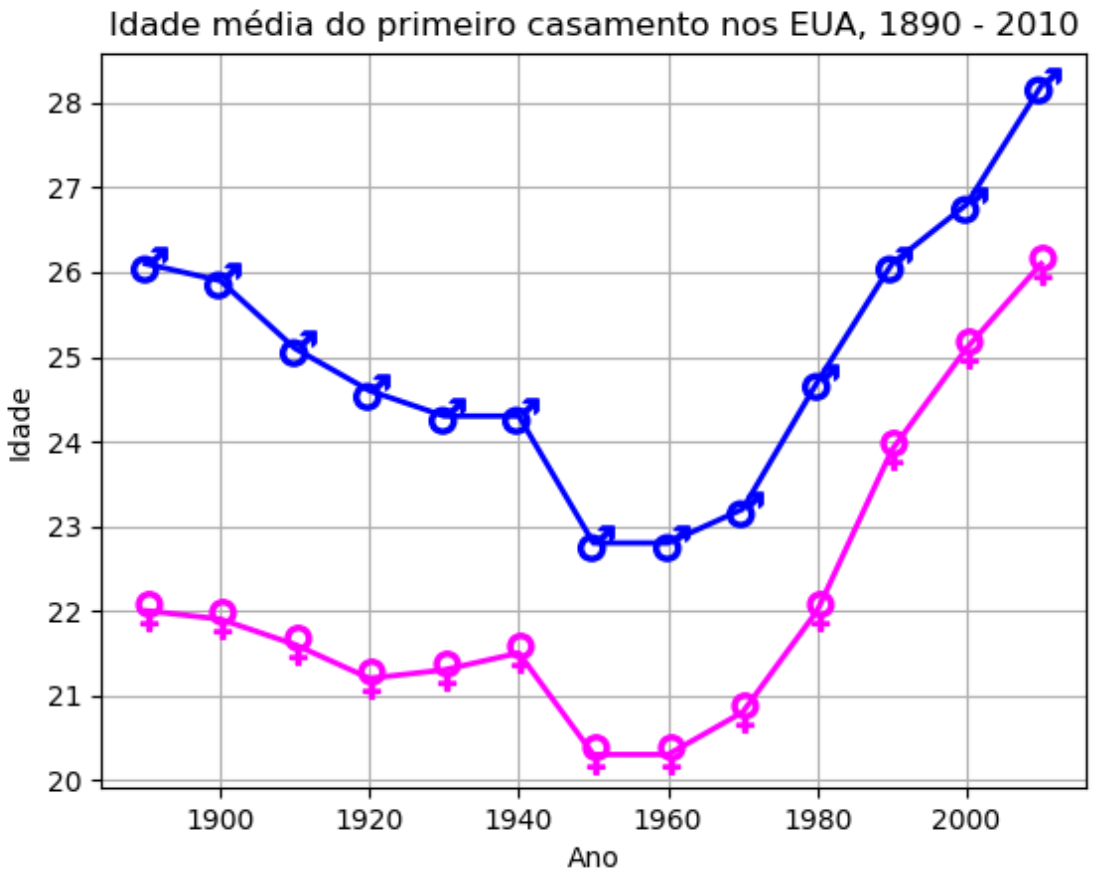


Histograma

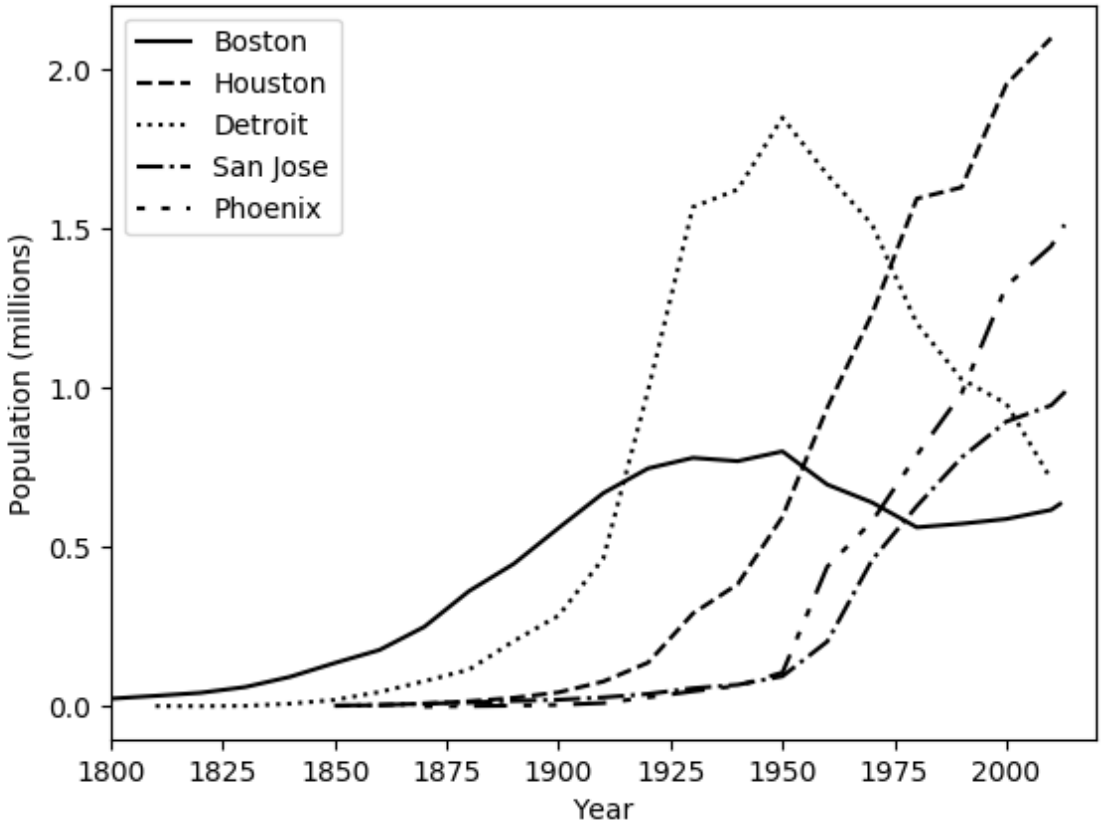
```
import matplotlib.pyplot as plt
def hist1():
  x=[1,4,7,8,16,22,22,22]
  plt.hist(x)
  plt.show()
  plt.savefig('c:/p/python/hist2.png')
hist1()
```



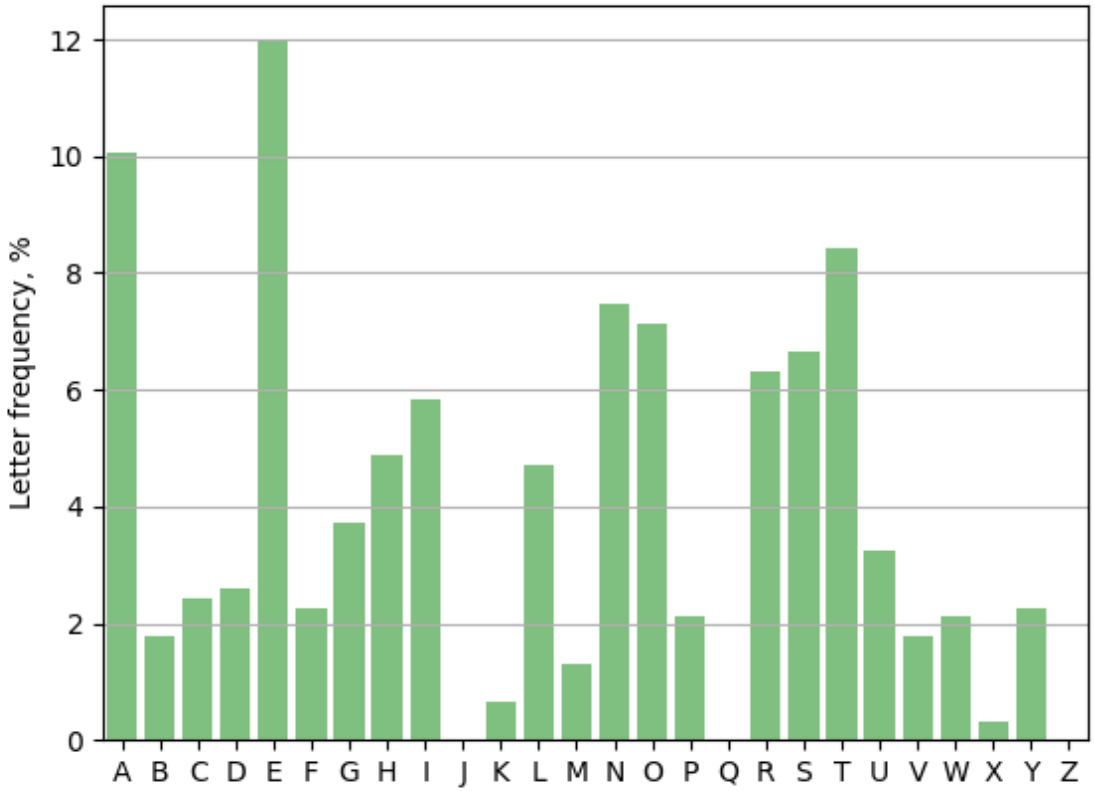
mais exemplos



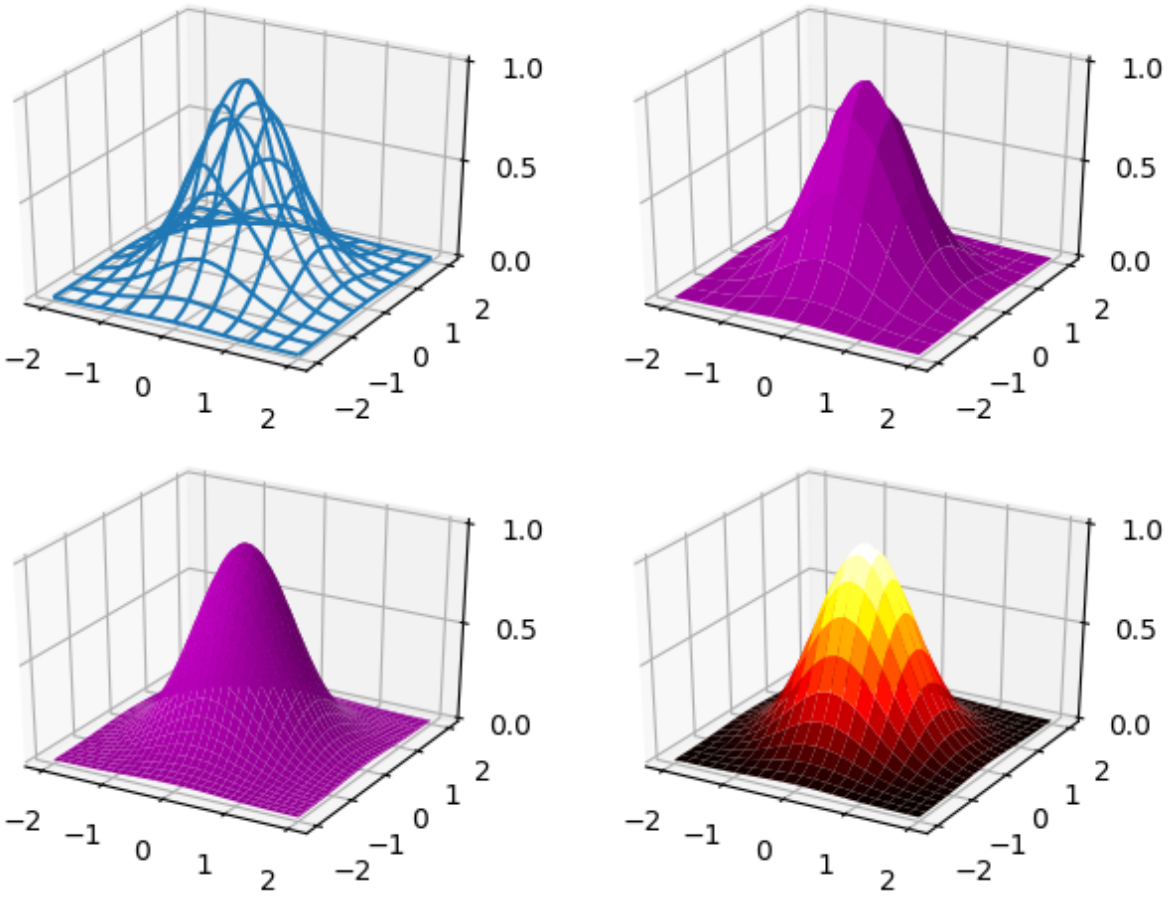
mais exemplos



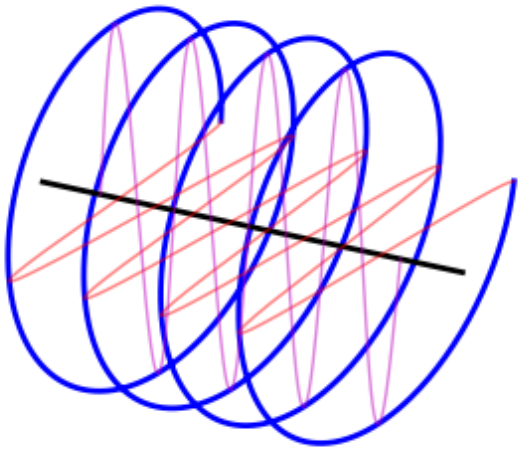
mais exemplos



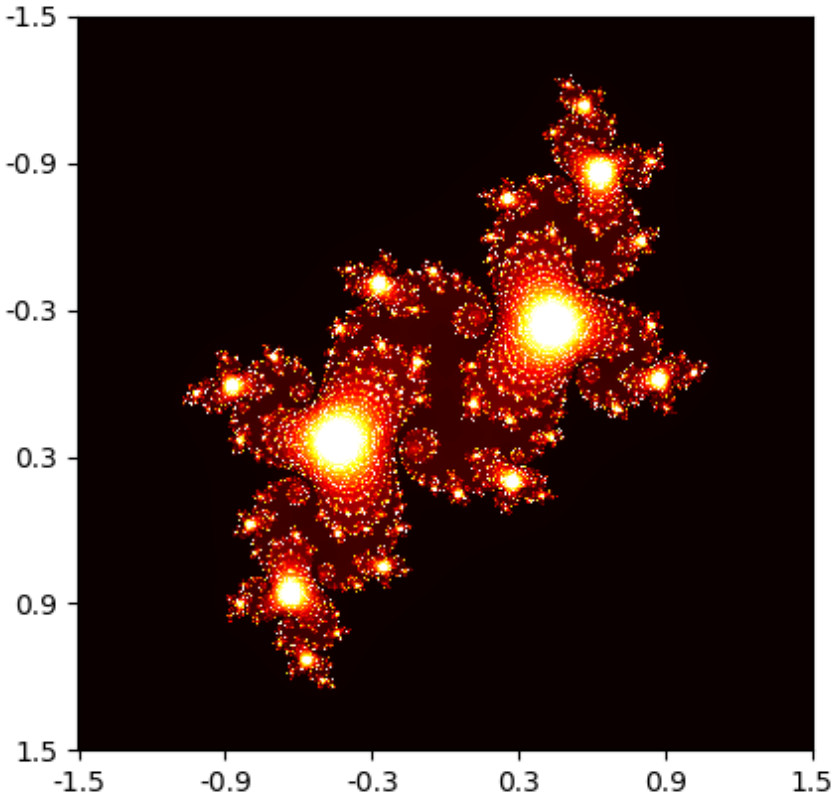
mais exemplos



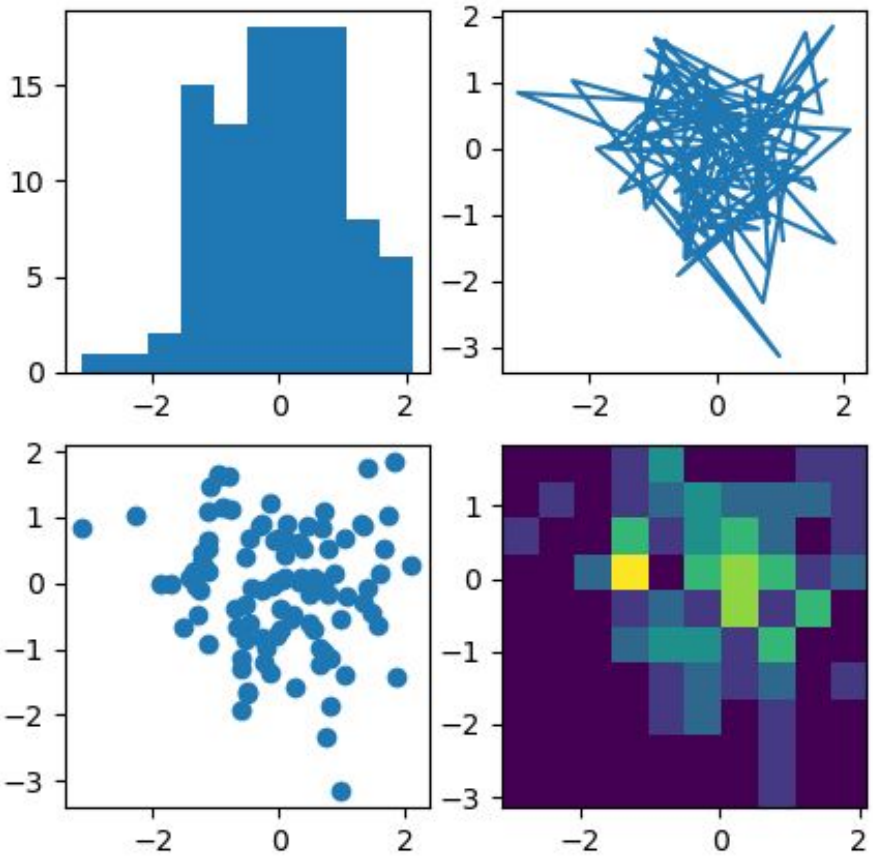
mais exemplos



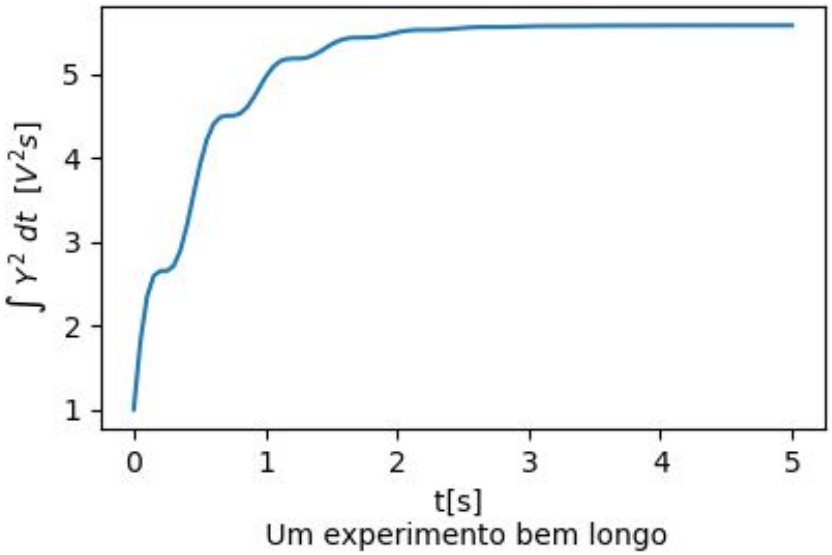
mais exemplos



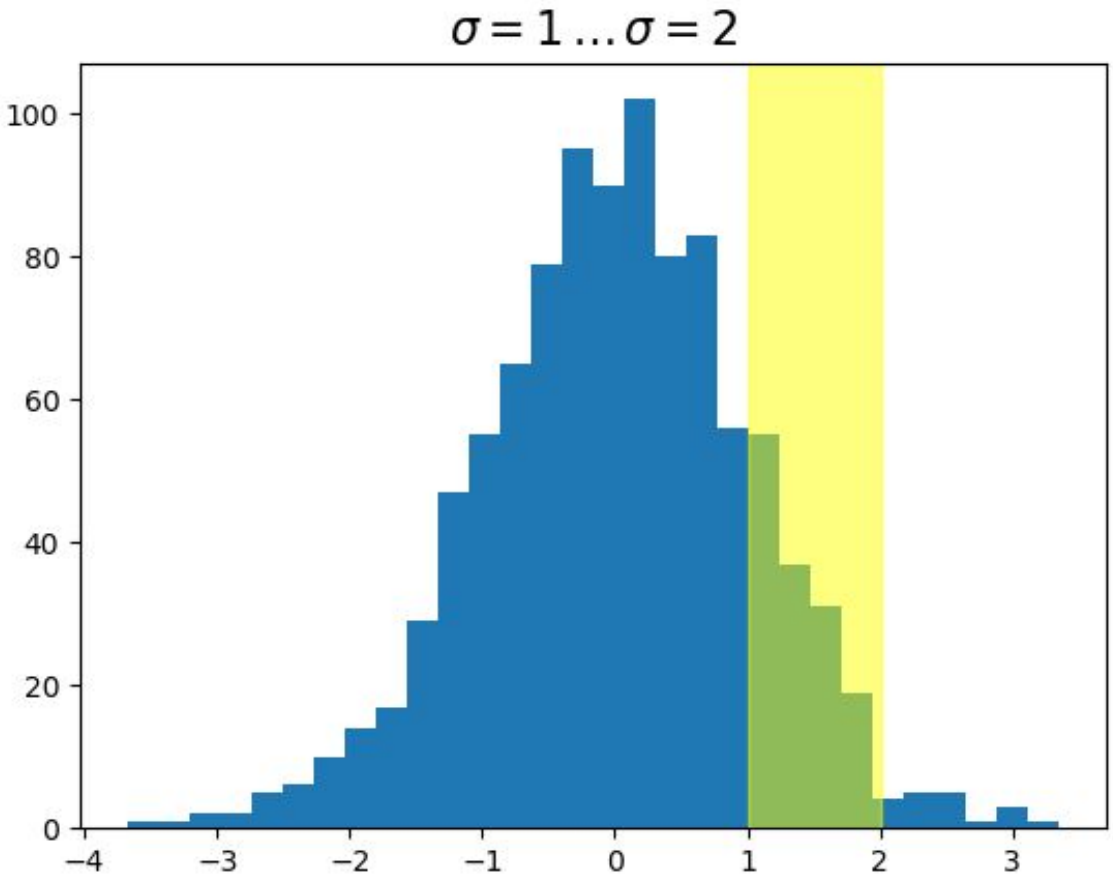
mais exemplos



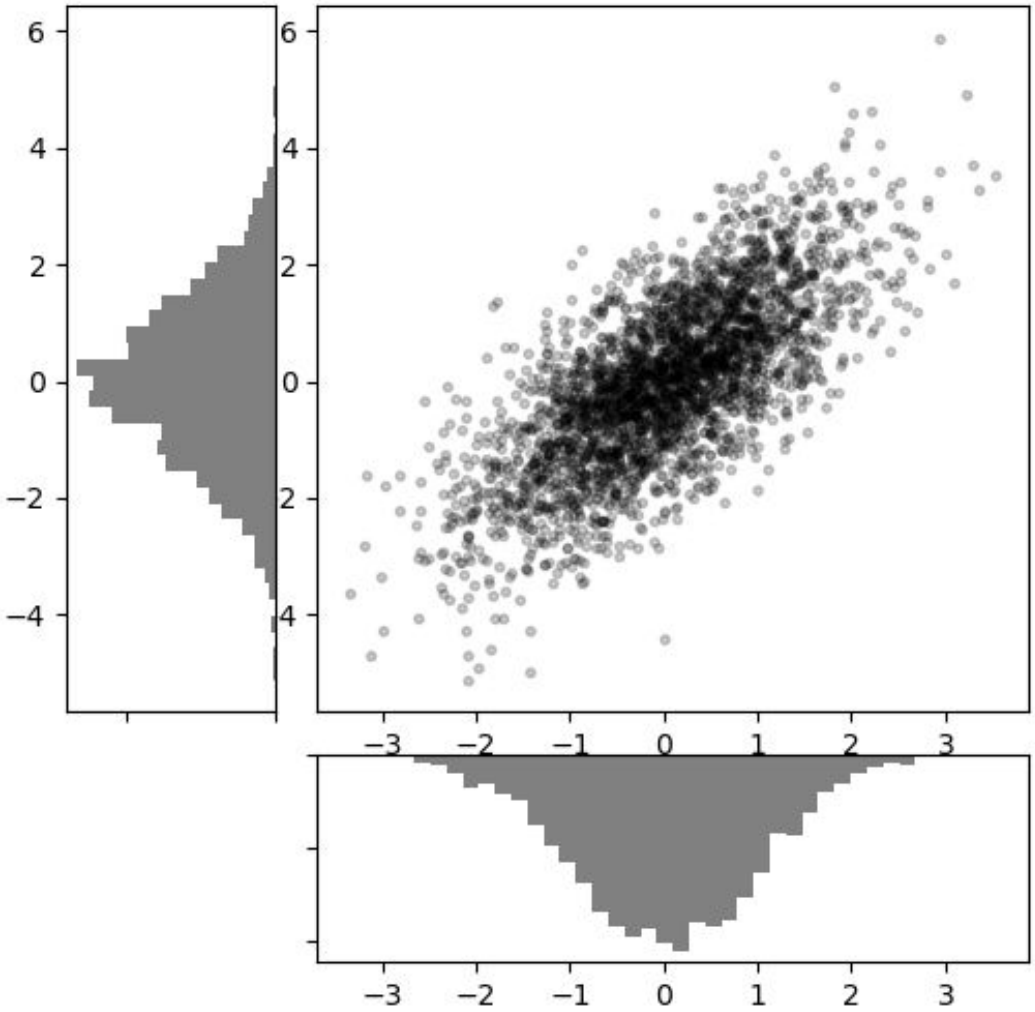
mais exemplos



mais exemplos

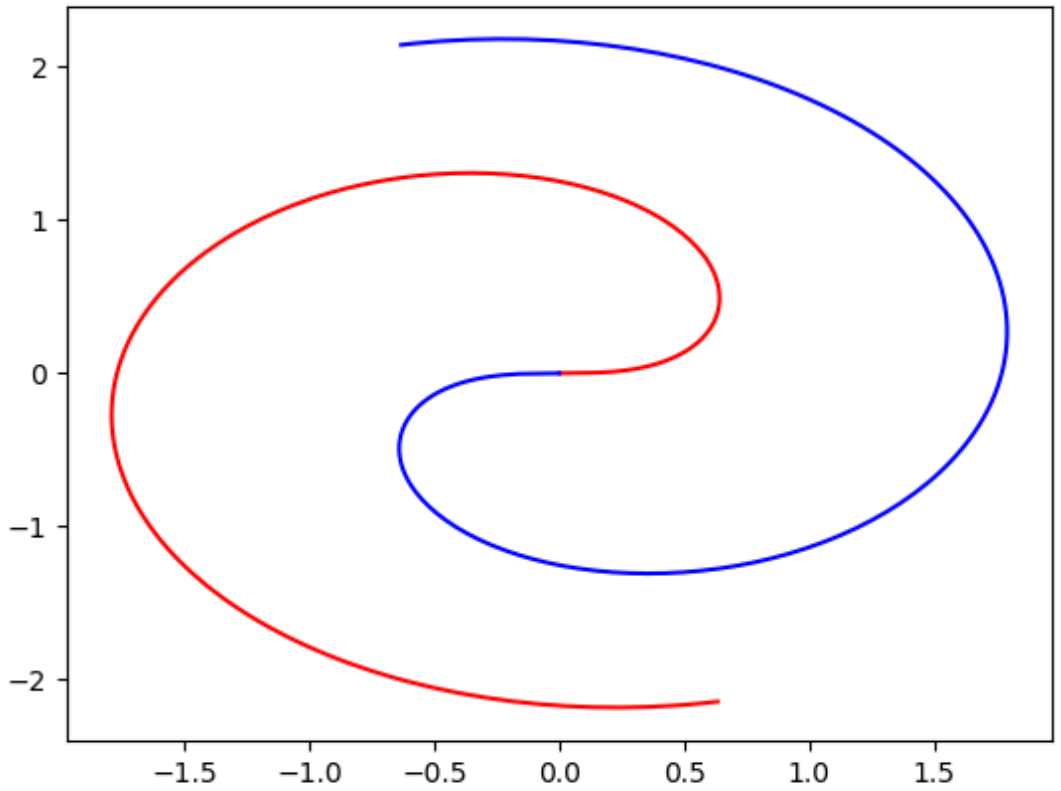


mais exemplos



Espiral de Fermat

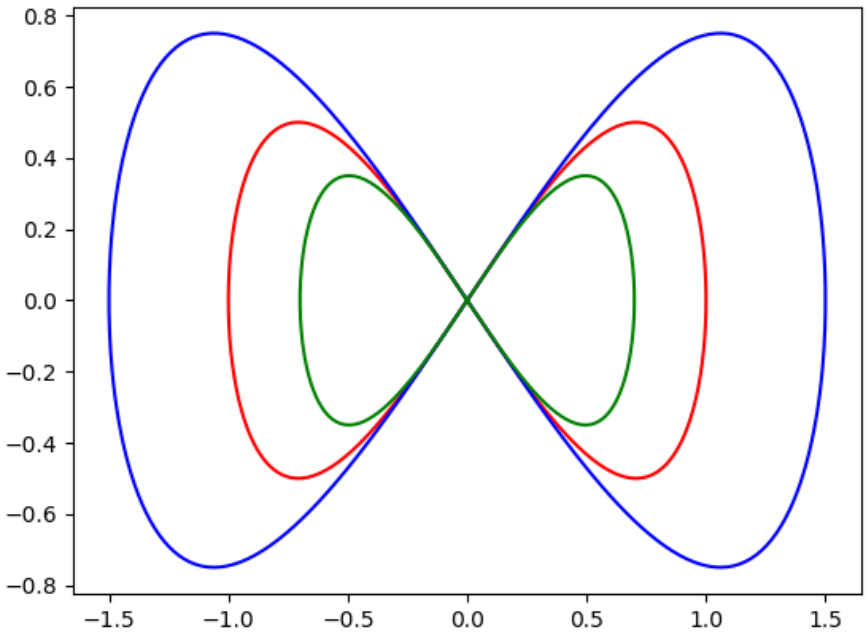
Em coordenadas polares:
$r = \theta^{1/2}$ ou ainda $r^2 = a^2 theta$



lemniscata

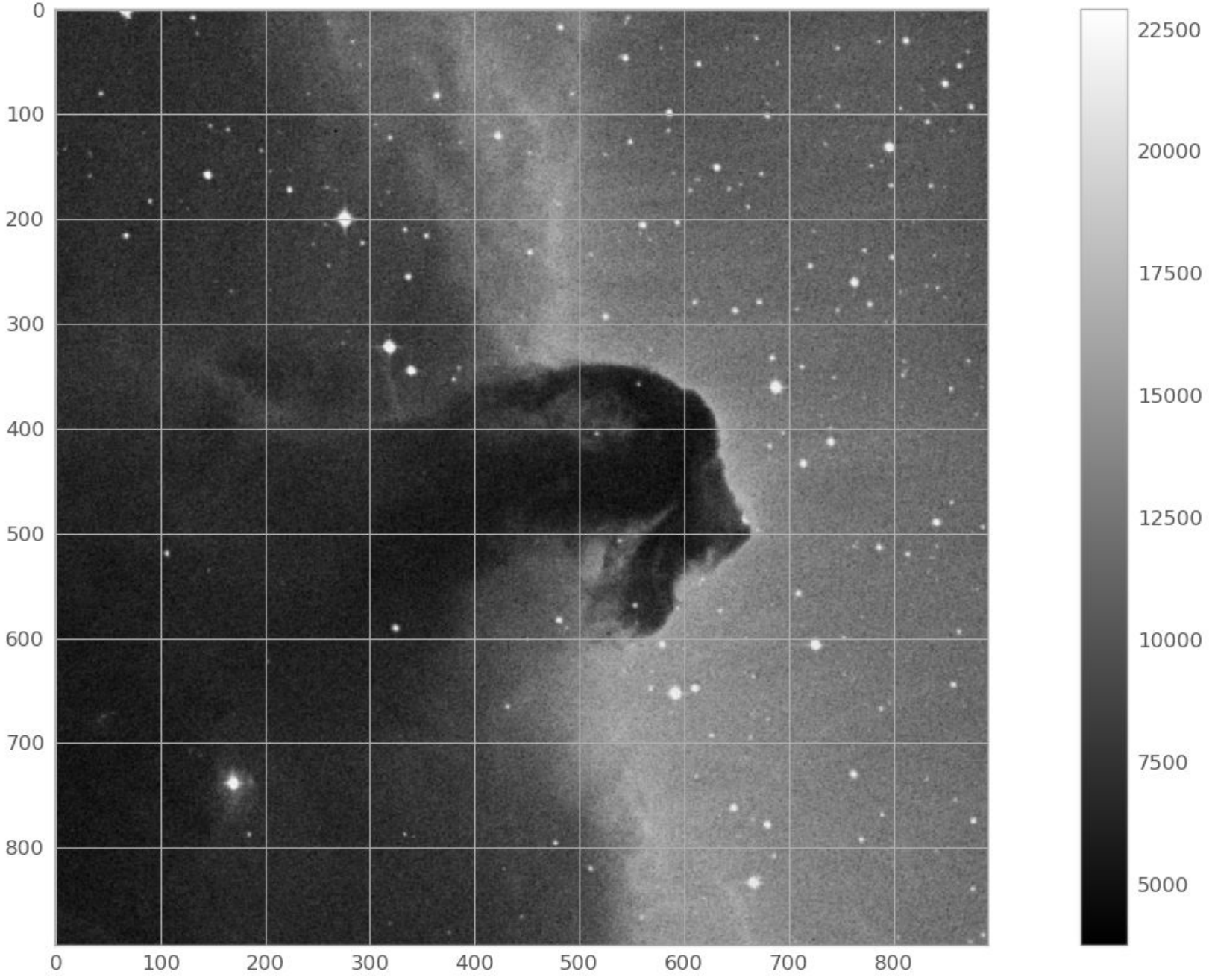
Em coordenadas cartesianas:
$(x^2 + y^2)^2 = 2a^2(x^2 - y^2)$

Em coordenadas polares:
$r^2 = 2a^2 cos2\theta$



Astropy

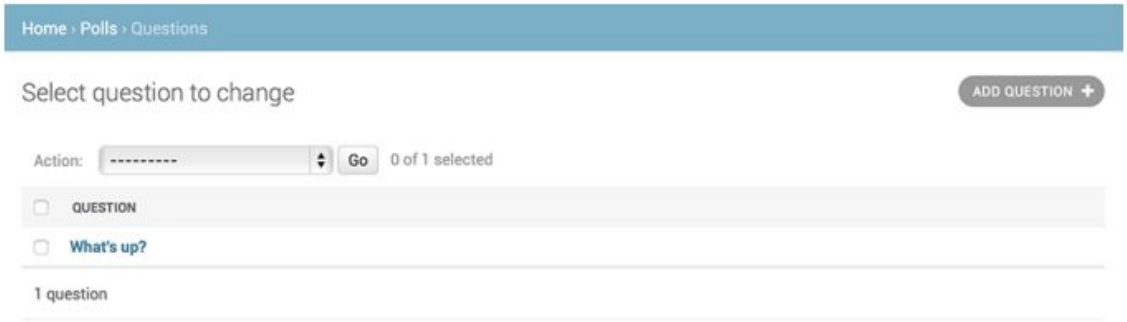
Padrão FITS=Flexible Image Transport System: padrão aberto de imagem digital nascido na astronomia para intercambiar imagens de (rádio) telescópios. Mote: ser legível mesmo depois que o software e o hardware usado não existam.
Escolhido pela Biblioteca do Vaticano para captura.
Livre, auto contido, nada é comprimido, bem documentado.
Mais detalhes em www.astropy.org


0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800
22500
20000
17500
15000
12500
10000
7500
5000

Django

- Framework de sistema web.
- A idéia de framework: uma arquitetura padrão, parte comum semi-pronta
  - Parte comum: usuários, login-senha, direitos, ...
  - geração e manutenção dos bancos de dados...
  - views, querys, funções, ...
- Pré-requisitos:
  - Banco de dados (mysql)
  - HTML, CSS
  - Python
  - Arquitetura de aplicações

Clique em "Questions". Agora cocê está na página "change list" para "questions". Essa página mostra todas as "questions" em seu banco de dados e lhe deixa escolher uma para alterar. Lá está a pergunta "What's up?" que criamos anteriormente.



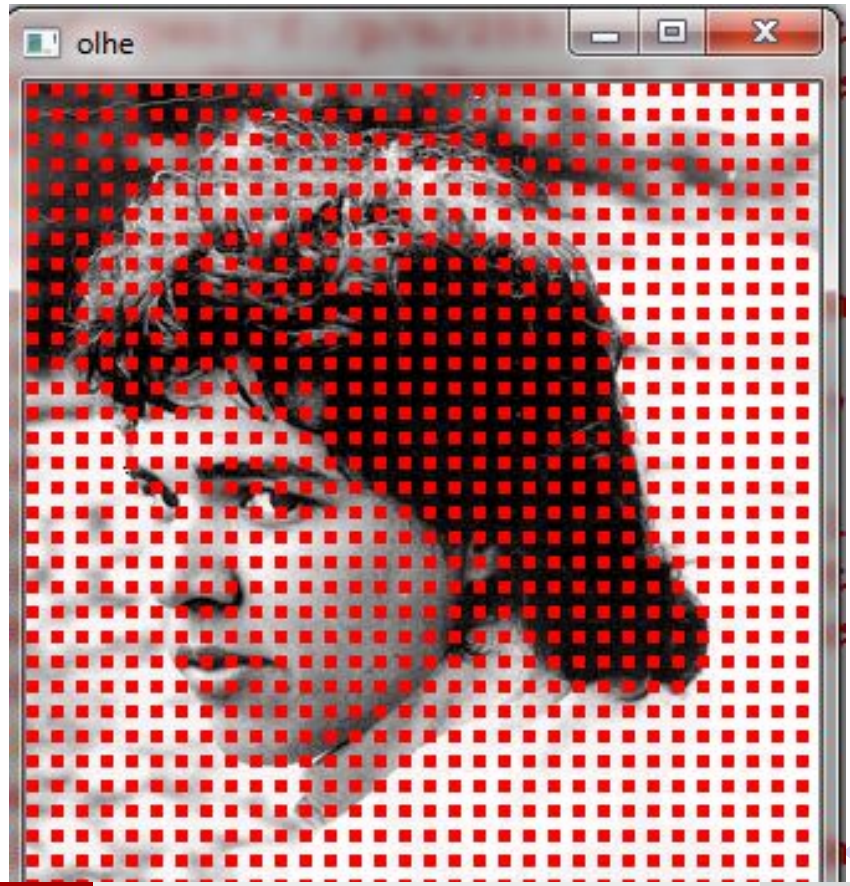
django

- Django is used by many popular websites/services:

OpenCV

Pacote de manuseio de imagens digitais. Originalmente “Open Computer Vision”.
OpenCV (Open Source Computer Vision Library). Originalmente, desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão computacional. Possui mais de 350 algoritmos de Visão computacional como: Filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e outros. O seu processamento é em tempo real de imagens.

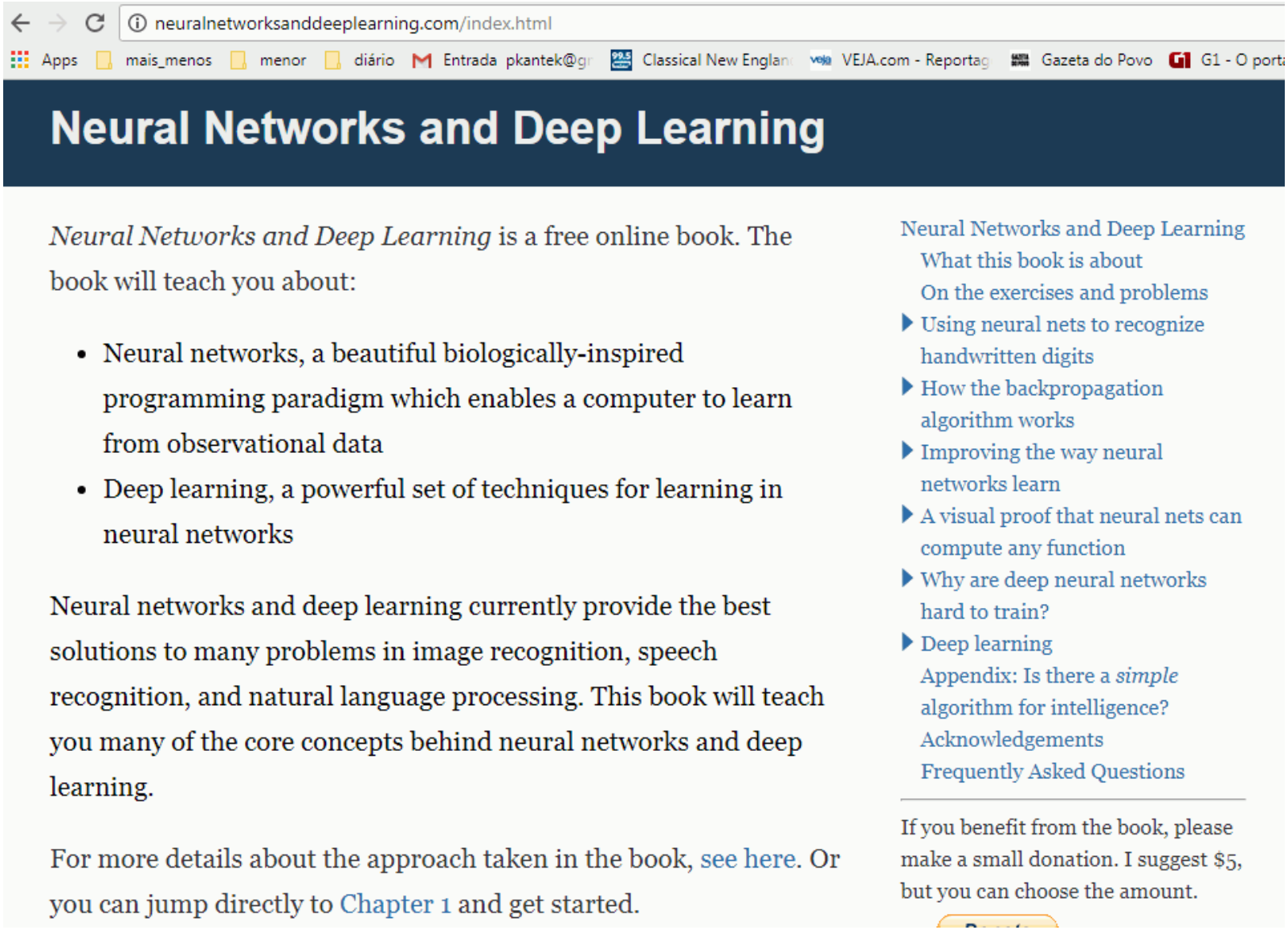
```
def ima1():
 import cv2
 imagem=cv2.imread('c:/p/pessoal/pedro_jovem.jpg')
 for i in range(0,imagem.shape[0],10):
  for j in range(0,imagem.shape[1],10):
   imagem[i:i+5,j:j+5]=(0,0,255)
 cv2.imshow('olhe',imagem)
ima1()
```



TensorFlow

- estado da arte em Redes Neurais (RN) configuráveis
- produto de produção da Google (tradutor, classificador, reconhecedor de fala,...)
- sucessor do DistBelief
- liberado em novembro/2015 e em desenvolvimento desde 2011
- www.tensorflow.org (licença Apache)

Um ótimo livro



Uma boa explicação de perceptron

Suponha que o fim de semana esteja chegando, e você já ouviu falar que haverá um festival de queijos em sua cidade. Você gosta de queijo e está tentando decidir se deve ou não ir ao festival. Você pode tomar sua decisão pesando três fatores:

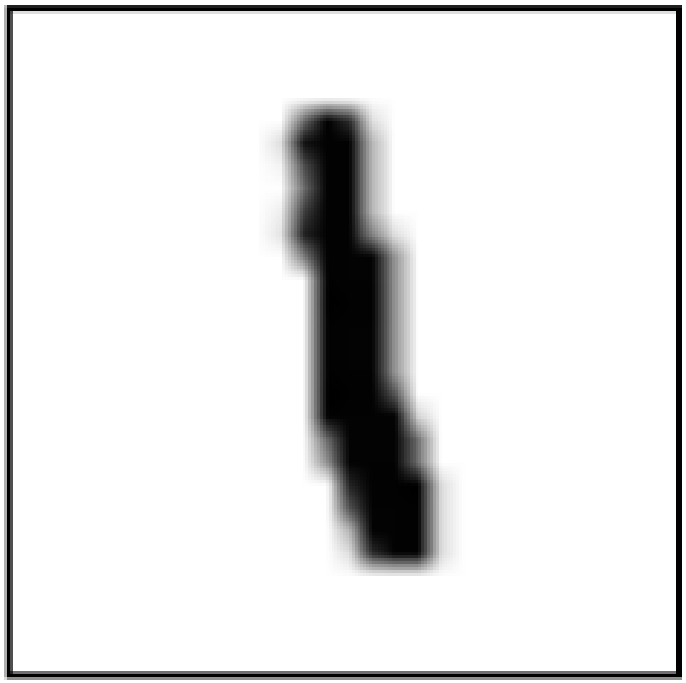
- O clima estará bom?
- Seu namorado ou namorada quer acompanhá-lo?
- O festival está perto do transporte público? (Você não tem carro).

Podemos representar estes três fatores pelas variáveis binárias correspondentes: Por exemplo, teríamos $x_1 = 1$ se o tempo estiver bom e $x_1 = 0$ se o tempo estiver ruim. Da mesma forma, $x_2 = 1$ se o seu namorado ou namorada quiser ir, e $x_2 = 0$ se não. E de forma similar novamente para x_3 é o transporte público

Dados do NIST

NIST é National Institute of Standards and Technology. Os dados de Machine Learning do NIST são famosos:
O MNIST vem em duas partes. A primeira parte contém 60.000 imagens para serem usadas como dados de treinamento. Essas imagens são amostras de manuscritas escaneadas de 250 pessoas, metade dos quais funcionários do Departamento de Censo dos EUA e metade dos estudantes do ensino médio. As imagens são de escala de cinza e 28 por 28 pixels de tamanho. A segunda parte do conjunto de dados MNIST é de 10.000 imagens a serem utilizadas como dados de teste. Novamente, estas são 28 por 28 imagens em escala de cinza. Usaremos os dados do teste para avaliar o quão bem a nossa rede neural aprendeu a reconhecer os dígitos. Para fazer deste um bom teste de desempenho, os dados do teste foram retirados de um conjunto diferente de 250 pessoas do que os dados de treinamento originais (embora ainda seja um grupo dividido entre os funcionários do Census Bureau e estudantes do ensino médio).

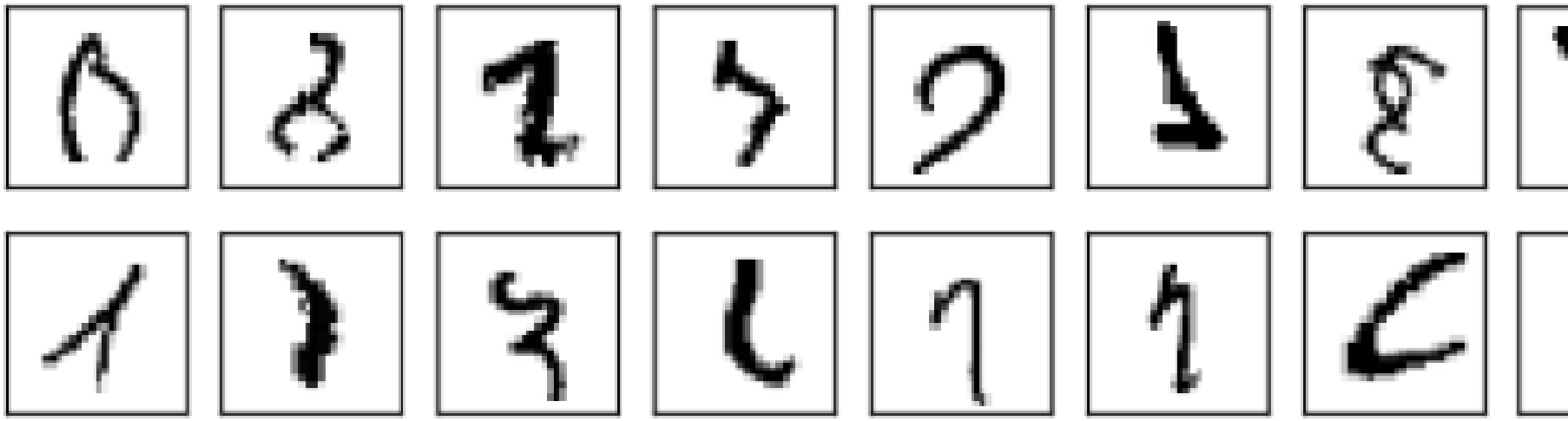
Os dados NIST



$\approx$

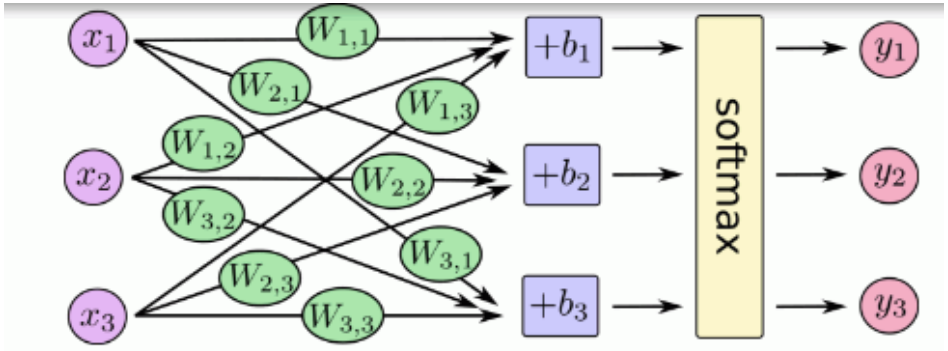
$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Os dados NIST: nem sempre é fácil



as redes neurais podem classificar com precisão todas menos 21 das 10.000 imagens de teste

Uma rede bem simples (sem camada escondida)

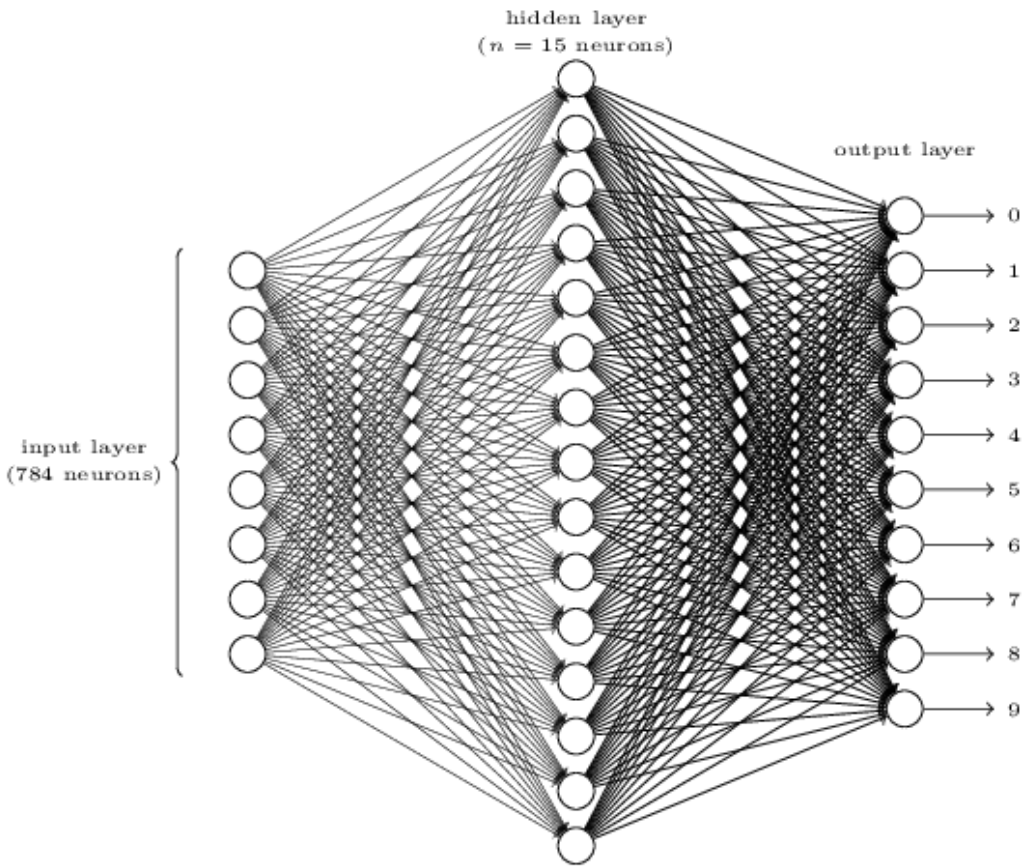


$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{bmatrix} \right)$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

O desempenho desta rede chega a 92% em média

Uma rede com camada escondida



O desempenho desta rede chega a 98% em média

O estado da arte chega a 99.8% das imagens (9979 em 10000)

Tensor Flow

Este produto só roda em windows 64 (lembrem-se, ele não economiza recursos) e estando nesse ambiente, a instalação é simples, basta fazer `pip install tensorflow`.

Depois de instalar, para verificar se deu tudo certo faça:

```
>>> import tensorflow as tf
>>> hello = tf.constant('Oi, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
```

Deve aparacer a mensagem `Oi, TensorFlow!` indicando que a instalação foi bem sucedida.

Tensor

O ponto central em TF é o tensor. Trata-se de um array de valores primitivos configurados em uma matriz (Python). O ranking de um tensor é o número de dimensões da matriz que o contém. Eis alguns exemplos:

```
3 # tensor de rank 0; escalar com forma []
[1. ,2., 3.] # tensor de rank 1;vetor forma [3]
[[1., 2., 3.], [4., 5., 6.]] # tensor rank 2;
#                       matriz com forma [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]]
# tensor rank 3; tensor com forma [2, 1, 3]
```

Uma RN: simples 1

O módulo abaixo é mnist_softmax.py...

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import sys

from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf #importa o tensorflow

FLAGS = None
```

RN simples: 2

```
def main(_):
  # Import data
  mnist = input_data.read_data_sets(FLAGS.data_dir,
     one_hot=True)

  # Create the model
  x = tf.placeholder(tf.float32, [None, 784])  # dados de entrada
  W = tf.Variable(tf.zeros([784, 10]))         # a RN
  b = tf.Variable(tf.zeros([10]))              # os bias
  y = tf.matmul(x, W) + b                      # o resultado da RN

  # Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])
       # lugar onde vem a resposta certa
```

RN simples: 3

```
cross_entropy = tf.reduce_mean(
  tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
# quero minimizar a diferença entre y (calculado) e y_ (certo)
train_step =
tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

sess = tf.InteractiveSession()   # cria a sessao
tf.global_variables_initializer().run()
   # inicializa variáveis
# Train
for _ in range(1000):
 # treinamento 1000 sessoes de 100 aleatorios
 batch_xs, batch_ys = mnist.train.next_batch(100)
 # coloca aqui os dados
 sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
 #roda a RN
```

RN simples: 4

```
# Test trained model
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
                  #y é igual a y_?
accuracy =
 tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
   #acurácia
print(sess.run(accuracy, feed_dict={x: mnist.test.images,
                                    y_: mnist.test.labels}))
   #imprime acurácia a partir das imagens de teste...
```

Pygame


```
import pygame
from pygame.sprite import Group
from settings import Settings # importa a classe de bloco de controle do jogo
from ship import Ship # importa a classe da nave que atira
from game_stats import GameStats # importa a classe de estatisticas
import game_functions as gf # importa o conjunto de funções do jogo
def run_game(): # define o jogo
    pygame.init() # inicializa o pigame
    ai_settings = Settings() # cria um bloco de constantes de nome ai_settings
    screen = pygame.display.set_mode((ai_settings.screen_width,ai_settings.screen_height))
        # acima: define uma tela com o tamanho especificado (largura x altura)
    pygame.display.set_caption("Invasão de aliens") # dá título à janela
    stats = GameStats(ai_settings) # cria uma estatística para este jogo
```

Itertools

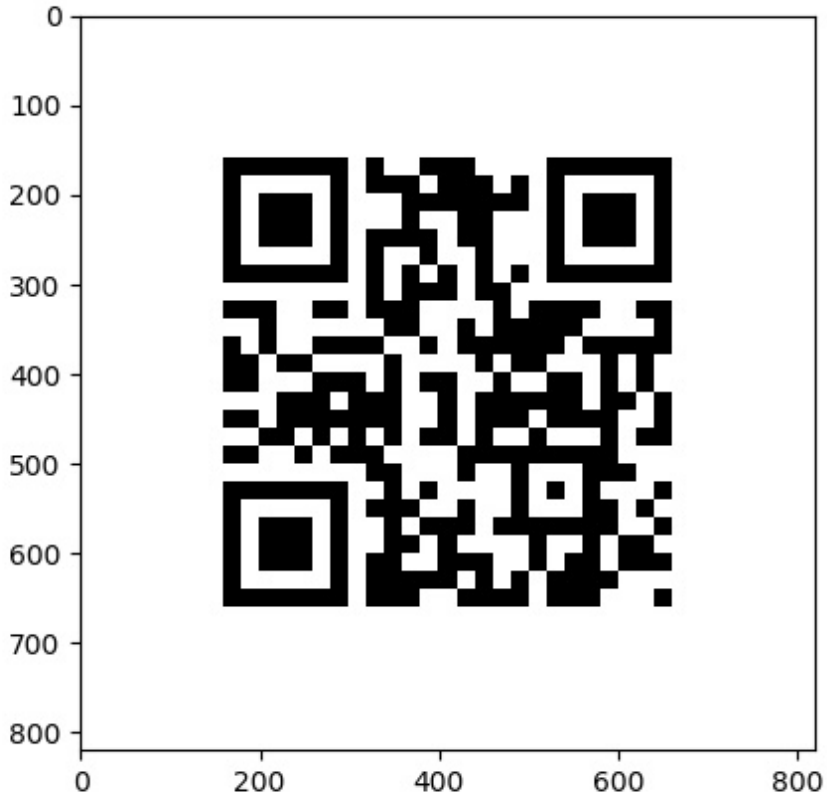
Pandas

Qrcode

Código ótico bidimensional
Nascido em indústria japonesa de ar condicionado
Padrão aberto mundial

- grande capacidade
- até 30% de recuperação em caso de erro
- padrão de fato no mercado

```
import qrcode as qq
import matplotlib.pyplot as plt
qr = qq.QRCode(
    version=1,   # tamanho do QRCODE 10=menor, 40=maior
    error_correction=qq.constants.ERROR_CORRECT_L,  #correçao de erro
    box_size=20,  # tamanho do pixel                #   l=7%, M=15% Q=25% e H=30%
    border=8, #tamanho da borda)
qr.add_data('Oi nois aqui, traveis')  #conteúdo do QRCODE
qr.make(fit=True)
img = qr.make_image()
plt.imshow(img)
plt.show()
```


0
100
200
300
400
500
600
700
800
0
200
400
600
800