# Cours_Python_SRI_outils_scientifiques

September 25, 2020

## 1 Programmation scientifique en python

Python devient de plus en plus une alternative à des outils traditionnels de l'ingénieur, comme Matlab, et son usage s'étend dans la communauté scientifique.

Pourquoi ?

- il est gratuit et ouvert
- c'est un "vrai" langage de programmation

- il bénéficie de la communauté open source: en progression constante
- il unifie de nombreuses bibliothèques existant dans différents langages (algèbre, analyse, statistiques, traitement signal, etc)
- librairie python pour "emballer" le tout: scipy, http://www.scipy.org/
- c'est devenu aussi un des premiers langages en analyse de donnée, IA, Machine Learning

On verra ici :

- le calcul matriciel avec numpy (base de nombreux autres outils)
- une introduction à quelques outils pertinents
  - traitement signal
  - traitement d'image
  - apprentissage automatique
  - graphes
  - robotique

### 1.1 ## Programmation vectorielle: numerical python (numpy)

introduction à la programmation scientifique et l'utilisation de vecteurs/matrices

- définitions, créations
- index/références
- quelques opérations / exemples de calcul vectorisé
- règles de "broadcasting" (composition)
- opérations + complexes, convolution (ex: automates cellulaires, fractales)

Exemple de création d'un vecteur

[2]:
```
import numpy as np
a = np.random.random(1000)
print(type(a))
print(a)
```

```
<class 'numpy.ndarray'>
[6.47390865e-01 9.06222126e-01 7.92057320e-01 2.78416388e-01
 9.27800418e-01 9.53241784e-01 1.75840754e-01 5.36280084e-01
 6.28828546e-01 4.10847691e-01 7.77801470e-01 4.35770215e-01
 7.09318506e-01 8.25526982e-01 4.52838895e-01 3.12461403e-01
 7.23382308e-01 9.83203732e-01 4.78504479e-02 3.94692155e-01
 7.66785107e-01 6.81311786e-01 6.84914110e-01 5.43188347e-02
 1.61571396e-01 3.54975395e-01 2.21641607e-01 9.52536951e-01
 3.64426019e-01 4.13799113e-01 2.07067061e-01 1.46088051e-01
 5.66879877e-02 5.05854572e-01 1.22633179e-01 8.69912217e-02
 6.71113995e-01 6.29576781e-01 7.75830060e-01 7.78506304e-02
 2.00390996e-01 7.18670228e-01 8.23787886e-01 4.01826539e-01
 8.84579202e-01 9.21591919e-01 9.72765531e-01 5.14335394e-01
 7.65488261e-01 9.09533079e-01 8.70097760e-01 9.87338447e-01
 4.69324002e-01 7.76511478e-01 1.20814000e-01 5.55130422e-01
 8.37124687e-01 7.18617533e-01 8.10605900e-01 6.02023164e-01
 8.22465419e-01 3.23346737e-01 6.77496359e-01 6.21683441e-01
 5.44730199e-01 8.63053386e-01 7.68603218e-01 2.43913558e-01
 6.41424211e-01 6.65564762e-01 1.76536943e-01 6.41506515e-01
 2.73713252e-01 3.40661031e-01 2.71408975e-01 7.79726878e-01
 3.15675454e-02 1.97481750e-01 8.94779068e-01 3.60991540e-01
 7.77916467e-01 2.35809665e-01 9.46199828e-01 1.74108724e-01
 4.01456641e-01 9.66167913e-01 7.83941452e-01 9.96324745e-01
 9.89621916e-01 5.74726818e-01 2.97553482e-01 8.89981347e-01
 8.12380211e-01 5.64495739e-01 4.83153218e-01 9.44300677e-01
 8.46109830e-01 6.18580861e-01 2.88167812e-01 6.03268580e-01
 8.79542380e-02 1.58995479e-01 5.32208071e-01 4.16273067e-01
 6.88918041e-01 2.03986045e-01 7.98384824e-01 4.45745970e-01
 5.89845627e-01 9.90150784e-01 6.30033061e-02 8.98000828e-01
 2.03329587e-01 4.67963647e-01 4.45546224e-01 9.72016640e-01
 3.17397132e-01 5.01612169e-01 2.85058854e-02 2.28502946e-01
 8.29215751e-01 6.95420747e-01 5.61645489e-01 7.35191490e-01
 8.11355302e-01 7.89833644e-01 6.86542875e-01 5.41401616e-01
 6.19092909e-01 5.97589065e-01 7.55696117e-01 5.53567066e-01
 7.49840707e-01 1.97178089e-01 3.50635721e-01 4.15123081e-01
 2.71195613e-01 6.50425770e-01 4.85137498e-02 9.33996309e-01
 5.30774937e-01 1.93156181e-02 2.49650749e-01 5.70389118e-01
 4.73507536e-01 1.59730120e-02 2.16644357e-01 3.20890432e-01
 8.91768485e-01 6.78326918e-01 9.50090804e-01 3.05890102e-01
 1.11310489e-01 7.70415493e-02 1.78154555e-01 3.32252583e-01
 9.24516747e-01 4.47981472e-01 7.50664272e-01 3.71265409e-01
 8.22075248e-01 2.86939763e-01 8.13404767e-01 4.29206628e-01
```

2.03790338e-01 2.92523472e-01 3.88056125e-01 7.63295478e-01
4.43036922e-01 8.48675297e-02 8.97519576e-01 7.26442100e-01
2.31360400e-01 4.76480549e-01 7.57609339e-01 4.49689461e-01
5.48172967e-01 1.53279697e-01 8.26715743e-01 7.36281749e-01
4.80711928e-01 5.00703007e-01 6.21949501e-01 5.63748018e-01
6.01609233e-01 1.17606440e-01 8.17072568e-01 1.41671086e-01
6.20033566e-01 8.76387614e-01 9.52682165e-01 1.26206688e-03
7.18691443e-01 3.52795455e-01 2.92943673e-01 3.48484382e-01
5.83766384e-01 1.59753393e-01 2.74998813e-01 6.06487341e-01
6.00337505e-01 2.65228794e-02 3.43504672e-01 7.93194927e-01
9.54381788e-01 1.88316600e-01 5.76704214e-02 5.54153360e-01
7.10899935e-01 9.94166409e-01 6.70459159e-02 5.95508878e-01
8.85122832e-01 6.07046017e-01 7.21012400e-01 4.08598805e-01
7.23631969e-01 7.88107525e-03 7.39850460e-01 2.23778021e-01
8.86306638e-01 1.78850254e-01 9.21094254e-01 4.16527988e-01
2.46805689e-01 7.16657980e-01 4.88914741e-01 2.99375384e-01
4.28869514e-01 7.69744292e-01 4.87964675e-01 8.74231611e-01
8.57997744e-01 6.73257274e-02 8.86216202e-01 4.54355562e-01
7.57615818e-01 4.66747455e-02 5.19859794e-02 5.40746325e-01
3.87637475e-01 4.21677570e-01 4.01158232e-01 8.40976055e-01
5.99061154e-01 3.04500974e-01 2.54020204e-01 6.59014768e-01
3.89911300e-01 6.15129939e-02 1.81841243e-01 6.59407451e-01
9.88180167e-01 3.86435982e-01 3.35931174e-01 6.99810111e-01
1.43092473e-01 9.09724620e-01 1.84412175e-01 9.35669135e-01
3.98054525e-01 1.78583269e-01 8.35429846e-01 3.42369033e-01
8.91786027e-01 5.29618474e-01 6.08534504e-01 1.46769178e-01
1.83381469e-02 2.55762826e-01 7.35945071e-01 1.60158508e-01
8.74280407e-01 9.17948282e-01 4.10571732e-01 1.96203272e-02
3.29742370e-01 6.68055611e-01 9.32395837e-02 5.98376037e-01
6.76418890e-01 2.84767397e-02 8.54070114e-01 6.50727697e-01
2.75549000e-01 5.68560824e-01 5.97897312e-02 5.04635380e-01
7.20431270e-01 3.55857465e-01 9.96771159e-01 8.81323181e-01
6.36680118e-01 2.36305490e-01 5.43695535e-01 1.52624342e-01
9.38985086e-02 2.26893969e-01 1.76723634e-01 4.73915284e-01
8.25527455e-01 4.80831784e-01 2.53129255e-01 9.99130359e-01
3.27246142e-01 7.24210858e-01 9.71820724e-01 2.52275710e-01
5.51642442e-01 1.24445482e-01 3.61466852e-01 6.60455444e-01
9.34160808e-01 4.49844111e-02 8.77440366e-01 2.33197931e-01
3.24760292e-01 5.69049169e-01 6.96183661e-01 6.48443402e-01
2.72580602e-01 2.77068233e-01 2.26211815e-01 3.64485254e-01
9.49919626e-01 6.81155774e-01 8.46095956e-03 9.37436359e-01
1.13851345e-01 6.97455216e-01 1.74151707e-01 2.31148913e-02
1.86950553e-01 9.49103183e-01 5.71355930e-01 1.56189366e-01
8.60275455e-01 7.38516524e-01 5.90638280e-01 9.44737287e-01
4.90969121e-01 2.95311526e-01 4.80469776e-01 3.04844997e-01
3.27277378e-01 7.40317039e-01 5.33032542e-02 8.28343115e-01
4.96548002e-02 7.07167478e-01 6.90331406e-01 2.73178532e-01
8.36526154e-01 5.77548811e-01 4.72501219e-01 6.77293481e-01

8.76929399e-01 6.53526560e-01 8.52906132e-01 4.55128017e-01
8.02338826e-01 7.20080601e-01 9.50087224e-01 5.76006931e-02
5.18881692e-01 4.25215080e-02 8.47474253e-01 9.01347566e-01
9.37713510e-01 9.71614079e-02 4.08192939e-01 4.58513866e-01
1.72581139e-01 4.05039663e-01 7.90427626e-01 7.83359624e-01
5.00211985e-01 5.08728994e-01 2.47433399e-01 3.51112387e-01
7.58535650e-02 3.17927807e-02 7.59050187e-01 7.92843674e-01
9.94237544e-01 3.46239694e-01 1.45521600e-01 6.97912351e-01
5.30577157e-01 2.11180145e-01 7.64010487e-01 7.56846863e-01
6.45525007e-01 1.90855074e-01 5.17699308e-01 5.70593091e-01
8.59839307e-02 8.05483607e-01 2.45176385e-02 5.78691017e-01
9.80448423e-01 9.23721224e-01 6.80558414e-01 6.75557904e-01
9.87970092e-01 9.84080834e-02 2.00053880e-02 2.12909637e-01
6.83038509e-01 3.56452683e-01 7.97577344e-01 3.53054061e-01
4.05491260e-01 7.72547329e-01 6.84801857e-01 9.35434043e-01
4.52230972e-01 8.57969766e-02 7.22557768e-01 9.58596741e-01
3.88952110e-01 7.23258354e-01 4.60560981e-01 3.53938171e-01
7.42019137e-02 7.22521478e-01 4.61929294e-01 7.94380963e-02
3.69928898e-02 3.15215264e-01 2.06880990e-01 5.64316846e-01
7.28937720e-01 2.92571566e-01 9.11514404e-02 1.92566723e-01
2.49665021e-01 4.36969278e-01 9.48790026e-01 9.32969640e-01
2.92741050e-01 2.85768363e-01 6.29898230e-01 8.76232543e-01
8.10164112e-01 5.61138143e-01 6.14204818e-01 2.69142409e-01
1.90413979e-02 7.48110526e-01 5.84402855e-01 7.68684427e-01
2.73722636e-01 9.49749230e-01 2.48056779e-01 3.53288362e-02
4.88276024e-01 3.69011335e-01 3.49346316e-01 6.22425872e-01
7.55896652e-01 6.46282431e-01 4.54229995e-03 3.28215926e-01
7.83273964e-01 6.13089863e-01 1.81793852e-01 7.04813567e-02
3.89957654e-01 4.58541410e-02 1.10138357e-01 4.01003279e-01
4.45730741e-01 3.76550583e-01 7.65352982e-01 3.20532937e-01
2.84989138e-01 3.01564149e-01 4.20569577e-01 5.84335546e-01
8.53325518e-02 5.03129074e-01 5.56348116e-01 7.24220176e-01
8.29539880e-01 1.53045899e-01 8.91553310e-01 6.46847783e-02
2.09943305e-01 3.71144741e-01 6.07535452e-01 3.57691935e-01
4.32857193e-01 8.66065740e-01 7.42252296e-01 3.48404892e-01
4.13034678e-01 2.82243425e-01 4.18917486e-01 4.59237959e-01
2.47603320e-01 4.18683277e-01 4.25093810e-01 3.81437670e-02
5.27394289e-01 9.21781309e-01 6.67030292e-01 6.75906903e-01
8.46773047e-01 1.37154538e-01 7.00873771e-01 7.43476961e-01
5.57665924e-01 5.76678309e-01 2.14654621e-01 5.70581603e-01
5.46550807e-01 8.12095107e-01 7.30175610e-02 9.67598010e-01
7.64551128e-01 5.76542856e-01 5.22735240e-01 2.29524712e-01
9.83726313e-01 1.32977567e-01 3.74911481e-01 2.21984929e-01
3.73352555e-01 1.85161998e-01 2.42396810e-01 4.17467617e-01
8.52746526e-03 8.01510473e-01 8.31844983e-02 9.64494701e-01
7.69402720e-01 3.34043514e-01 9.90472481e-01 1.35264756e-01
3.46079273e-01 6.49565265e-01 9.88124757e-02 3.38519848e-01
6.40920595e-01 1.58471371e-01 3.49572287e-01 8.77664388e-01

9.94216872e-01 7.09305785e-01 7.21258132e-01 5.88634848e-01
7.79592111e-01 6.89471164e-01 8.13847482e-01 2.80795850e-01
9.23764384e-01 1.69480933e-01 7.86347055e-02 7.03203925e-01
1.32574539e-01 1.83894819e-01 8.81745288e-01 8.19298432e-01
3.61456697e-01 5.44082316e-01 1.90989900e-04 5.50728364e-01
3.51311224e-01 2.35772979e-01 9.46136793e-01 9.87653353e-01
8.96688898e-01 3.22711996e-01 3.54001141e-01 5.74220667e-01
2.48719656e-01 4.99448620e-01 5.35727487e-01 4.83301323e-01
3.07970715e-01 7.33926931e-01 2.90173384e-01 8.10877775e-01
5.75699752e-01 8.83816230e-01 4.38162473e-01 9.97647830e-01
6.03288314e-01 5.31960653e-01 8.84734756e-01 7.92904077e-01
3.58089720e-01 2.22434846e-01 8.33401263e-01 2.00699863e-01
7.64442979e-01 9.66568016e-01 1.56126869e-01 7.04282848e-01
4.43082803e-02 5.17952695e-01 1.27071031e-01 1.54788979e-01
1.99763615e-01 7.45042832e-02 4.89883497e-01 2.65546437e-04
1.12889577e-01 1.60206599e-02 8.25205218e-01 5.47203773e-01
6.40532464e-01 7.40431401e-02 9.15095428e-01 4.27516310e-01
6.33897648e-01 2.66424575e-01 7.67222982e-01 2.02081385e-01
2.40371775e-01 2.09406702e-01 7.96497773e-01 8.84211072e-01
7.04051293e-01 4.12892221e-01 3.75928953e-02 8.52687337e-01
2.65936862e-01 1.59088189e-02 8.91727124e-01 1.26469624e-01
7.35064617e-01 8.19726687e-01 6.17297496e-01 2.18681659e-01
8.84630244e-01 2.75125605e-01 7.94504981e-02 2.84892594e-01
8.16859560e-01 1.30762264e-01 4.35565378e-02 6.92956635e-01
1.14521521e-01 9.48152864e-01 1.00071044e-01 5.90405782e-01
1.29707199e-01 5.93654987e-01 3.11606511e-01 3.48265771e-01
4.78613517e-01 2.20377865e-01 4.62433047e-01 8.35109396e-01
7.49083595e-01 9.92584834e-02 5.89840218e-01 9.81043645e-01
6.81925711e-01 3.50143699e-01 9.09110653e-01 6.73314010e-01
4.84402828e-02 2.84976534e-01 4.69461898e-01 2.56117714e-01
3.10956473e-01 5.80530010e-01 7.34903235e-01 8.35700720e-01
9.88249892e-01 1.91460863e-01 7.11069504e-02 9.54211540e-01
5.25652653e-01 4.21557872e-02 4.32566517e-01 9.21512286e-01
6.24241372e-01 3.06172695e-01 2.17632784e-01 5.16685939e-02
1.70703169e-01 6.26086205e-01 1.19033880e-01 3.73494078e-01
7.97062506e-01 8.47093555e-01 3.41287883e-01 7.55954377e-01
8.07276670e-01 2.61400765e-01 9.08864407e-01 6.99649378e-01
3.01570182e-01 2.51609828e-01 2.25878625e-01 2.30202237e-01
7.26398291e-02 9.37136652e-01 4.90180979e-01 6.14682115e-01
1.45406025e-01 5.62729903e-01 3.81912192e-01 1.70324968e-01
5.30562364e-01 3.96937651e-01 5.58890817e-01 4.07488596e-01
9.13097260e-02 4.50010938e-01 3.94852092e-01 3.76904023e-01
4.65122165e-01 8.11913775e-01 7.71464263e-01 4.43181490e-01
9.78366595e-01 3.40276303e-01 3.57901754e-01 3.29164550e-01
9.56669634e-01 8.46941739e-01 6.93567230e-01 8.76986736e-01
9.49377196e-01 3.98045830e-01 1.65336494e-03 3.30695641e-01
5.98779712e-01 8.79215524e-01 6.24465103e-01 7.28932014e-01
6.39808386e-02 3.38785095e-01 7.96075175e-01 1.00665748e-01

1.99893491e-01 9.28290574e-01 6.82897837e-01 3.95384210e-01
1.67924494e-01 6.14007209e-01 2.46064776e-01 2.67922984e-01
5.24283982e-01 4.32125288e-01 1.25262314e-01 6.66973968e-01
3.86294735e-01 2.00338572e-01 5.34507947e-01 2.48731769e-01
1.67282233e-01 4.99731418e-01 4.73862187e-01 9.93650828e-01
9.41598850e-01 6.96836719e-01 3.91605934e-02 4.62811449e-01
4.30661685e-01 1.43648416e-01 1.22458153e-01 7.91559421e-01
7.84433029e-01 1.61938169e-01 2.26964632e-01 2.07127608e-01
2.59108287e-01 6.00161245e-01 2.14868070e-01 9.68567528e-01
4.13750452e-01 9.16418366e-01 4.45648376e-01 8.23445235e-01
8.51315703e-01 4.00353972e-01 2.10520794e-01 2.95400015e-01
1.11938277e-01 1.59568499e-01 5.95336036e-01 1.25576201e-01
3.26313051e-01 7.33755960e-01 3.09787363e-01 9.85258895e-01
9.53533520e-01 5.27134952e-01 7.62008708e-01 5.49208063e-01
1.15348613e-02 5.79458407e-01 4.54919727e-01 8.48104929e-01
3.01724019e-01 1.79979844e-01 5.42107690e-01 1.47590005e-01
1.77079215e-01 2.68879531e-01 3.53764008e-01 5.20074264e-01
6.92755897e-01 9.02677218e-01 8.14237407e-01 1.32347389e-01
3.91087793e-01 7.81413362e-01 6.33757384e-01 8.66960835e-02
5.78990687e-01 9.07381432e-01 8.66155411e-01 6.36067613e-01
5.44634798e-01 6.79894193e-01 9.38520125e-01 8.87587077e-01
7.01990021e-01 5.24316894e-01 9.09339621e-02 7.37001690e-01
6.42150127e-01 9.61915137e-01 8.38478758e-01 2.24317369e-01
8.86782110e-01 9.57795670e-01 4.84453250e-01 5.21513905e-02
3.45309393e-01 4.86934672e-01 6.87774298e-01 6.34435359e-01
8.28019210e-01 7.31774700e-01 6.19872203e-01 2.36452061e-01
6.99306979e-01 1.67667023e-01 7.51499554e-01 2.45474873e-01
2.22630539e-01 7.97972221e-02 4.35893692e-02 1.10314789e-01
6.51409292e-01 2.03036025e-01 1.89711702e-01 5.58172253e-01
3.64395633e-01 6.16439381e-01 2.80643404e-01 2.83921398e-01
5.69316035e-01 3.12040907e-01 8.43116222e-01 1.58555597e-01
5.33237646e-01 1.15009614e-01 3.69742631e-01 1.10943848e-01
6.18773817e-01 1.35682453e-01 9.06158745e-01 9.63340283e-01
5.72611492e-01 3.64353972e-01 7.50452677e-01 8.22356623e-02
2.77049104e-01 9.69377800e-01 7.19698665e-01 9.24677813e-01
4.37649936e-01 6.23601144e-01 4.50113247e-01 2.84483380e-01
3.77600543e-01 2.06349625e-01 9.28430822e-01 7.63507806e-01
3.16746074e-01 9.00116963e-01 9.09778230e-01 6.86658299e-01
4.89056840e-01 9.53741103e-01 7.17336199e-01 3.74907068e-01
2.52678749e-01 3.00226893e-01 5.68771625e-01 5.51045394e-01
9.27277157e-01 2.96905496e-01 6.61887571e-01 4.42060973e-01
1.20684551e-01 5.76269024e-01 3.88870256e-01 5.86682690e-01
8.60387827e-01 8.31953607e-01 8.14997714e-01 5.28195632e-01
3.82364265e-01 9.65192880e-01 3.77286222e-01 2.08239482e-01
3.47877858e-01 7.20855574e-01 8.15252999e-01 8.23188080e-04
8.65378236e-01 1.58994079e-01 9.00525113e-01 5.94368273e-02
7.01604834e-01 3.18431215e-01 6.38693688e-01 4.89914809e-01
4.05310945e-01 3.69221815e-01 6.05435886e-01 9.36195178e-01

```
 4.66485407e-01 2.80284212e-01 5.83830218e-01 7.58017958e-01
 9.40347285e-01 1.01222936e-01 1.78887491e-01 6.57599079e-01
 6.85745320e-01 9.80296374e-01 7.71563185e-01 6.36629288e-01
 4.63582663e-01 8.39022770e-01 4.23388998e-01 7.01463657e-01
 9.42595914e-01 5.99942260e-01 2.34407177e-01 4.60516928e-01
 1.98145416e-02 4.30708627e-01 4.92621201e-01 5.38220829e-01
 6.02275685e-01 3.48913128e-01 2.13712719e-01 3.43601787e-01
 6.08361142e-02 9.78762262e-01 1.19864532e-01 8.73256521e-01
 1.08992683e-01 1.60472373e-01 9.51974083e-01 9.73438645e-01
 4.30126420e-01 3.36960247e-01 4.29831906e-01 7.29456781e-01
 7.33345102e-02 9.77628069e-01 5.71750553e-01 7.34886756e-01
 3.40005707e-01 1.21456319e-01 7.02004056e-01 1.73418943e-01
 4.81021883e-01 4.73391230e-01 7.84971066e-01 6.33964505e-01
 7.59019316e-01 3.41910743e-01 4.22225397e-01 6.86634730e-03
 9.40267515e-02 5.41691163e-01 8.83359793e-01 6.55082712e-01
 8.14998411e-01 2.43243424e-01 2.92447409e-01 6.11073660e-01
 5.86422228e-01 5.23138744e-01 2.06136433e-01 5.23231274e-01]
```

[3]:
```
print(a[:10])

print(a.min(),a.max(),a.mean())
```

```
[0.64739086 0.90622213 0.79205732 0.27841639 0.92780042 0.95324178
 0.17584075 0.53628008 0.62882855 0.41084769]
0.00019098989981880532 0.9991303591608571 0.5021080008864625
```

### 1.2 Calcul vectorisé

[4]:
```
# on peut pas ...
from math import cos
print(cos(a))
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-4-bb9946c49d67> in <module>
  1 # on peut pas ...
  2 from math import cos
----> 3 print(cos(a))


TypeError: only size-1 arrays can be converted to Python scalars
```

[5]:
```
# numpy a des fonctions "universelles"
from numpy import cos
c = cos(a)
print(c[:10])
```

```
[0.7976601  0.61672401 0.7023824  0.96149187 0.59959576 0.57904312
 0.98457981 0.85961529 0.80871711 0.9167826 ]
```

[6]:
```
# et des opérations vectorielles / matricielles
(a+a)[:10]
```

[6]:
```
array([1.29478173, 1.81244425, 1.58411464, 0.55683278, 1.85560084,
       1.90648357, 0.35168151, 1.07256017, 1.25765709, 0.82169538])
```

### 1.2.1 Comparaison de performances

[7]:
```
b = list(a)
print(b[:10])
```

```
[0.6473908646028699, 0.9062221261242952, 0.7920573201923637, 0.2784163875383706,
0.9278004180370281, 0.9532417841692604, 0.17584075357503948, 0.5362800840824067,
0.6288285463897716, 0.4108476906421247]
```

[8]:
```
%timeit a**2
```

```
2.44 µs ± 147 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

[9]:
```
 %timeit [x**2 for x in b]
```

```
533 µs ± 17.5 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

[10]:
```
%%timeit
s = 0
for x in b:
    z =  x**2
```

```
579 µs ± 39.9 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

## 1.3 Indexation et dimensions

[58]:
```
c = np.random.random(1000)
print(type(c))
print(c[9])
c.shape
```

```
<class 'numpy.ndarray'>
0.09688718654390482
```

[58]:
```
(1000,)
```

[59]:
```
nc = 100
c = c.reshape(c.shape[0]//nc,nc)
print(c[0,9])
print(c.shape)
print(c[:,2:4])
```

```
0.09688718654390482
(10, 100)
[[0.19082891 0.55505454]
 [0.78906142 0.05144676]
 [0.54241441 0.80034217]
 [0.11303389 0.49071946]
 [0.91390556 0.02892353]
 [0.93979772 0.53768931]
 [0.45594293 0.04930522]
 [0.16158351 0.07651733]
 [0.87074684 0.39642192]
 [0.35735449 0.34872099]]
```

[13]:
```
print(c[:,1])
print(c[1::2,1])
```

```
[0.90622213 0.15899548 0.02652288 0.48083178 0.92372122 0.41868328
 0.5179527  0.93713665 0.17997984 0.2969055 ]
[0.15899548 0.48083178 0.41868328 0.93713665 0.2969055 ]
```

### 1.4 Création

[14]:
```
import numpy as np
c = np.zeros(shape=(3,4))
print(c.shape)
print(c)
c.ravel()
```

```
(3, 4)
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

[14]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[15]:
```
c = np.arange(10)
print (c)
```

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

[63]:
```
c  = np.arange(0.,3.2,0.05)
cos(c)
```

[63]:
```
array([ 1.        ,  0.99875026,  0.99500417,  0.98877108,  0.98006658,
        0.96891242,  0.95533649,  0.93937271,  0.92106099,  0.9004471 ,
        0.87758256,  0.85252452,  0.82533561,  0.7960838 ,  0.76484219,
        0.73168887,  0.69670671,  0.65998315,  0.62160997,  0.58168309,
        0.54030231,  0.49757105,  0.45359612,  0.40848744,  0.36235775,
        0.31532236,  0.26749883,  0.21900669,  0.16996714,  0.12050277,
        0.0707372 ,  0.02079483, -0.02919952, -0.07912089, -0.12884449,
       -0.17824606, -0.22720209, -0.27559025, -0.32328957, -0.37018083,
       -0.41614684, -0.46107269, -0.5048461 , -0.54735767, -0.58850112,
       -0.62817362, -0.66627602, -0.70271308, -0.73739372, -0.77023125,
       -0.80114362, -0.83005354, -0.85688875, -0.8815822 , -0.90407214,
       -0.92430238, -0.94222234, -0.95778724, -0.97095817, -0.9817022 ,
       -0.9899925 , -0.99580832, -0.99913515, -0.99996466])
```

[67]:
```
c = np.linspace(0, 1.0,101)
c
```

[67]:
```
array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99, 1.  ])
```

[ ]:
```

```

### 1.5 Création (suite)

[18]:
```
np.zeros((2,2))
```

[18]:
```
array([[0., 0.],
       [0., 0.]])
```

[19]:
```
np.ones((2,2))
```

[19]:
```
array([[1., 1.],
       [1., 1.]])
```

[20]:
```
np.eye(3)
```

[20]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

### 1.6 Références

attention aux références -> différent des listes

les “tranches” de vecteurs/matrices sont justes des “vues” sur le vecteur

[73]:
```
c[1] = 0
print(c)
a = c[1:-1]
c[1] = -1
print(a[:10])
```

```
[0.   0.   0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1  0.11 0.12 0.13
 0.14 0.15 0.16 0.17 0.18 0.19 0.2  0.21 0.22 0.23 0.24 0.25 0.26 0.27
 0.28 0.29 0.3  0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4  0.41
 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5  0.51 0.52 0.53 0.54 0.55
 0.56 0.57 0.58 0.59 0.6  0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69
 0.7  0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8  0.81 0.82 0.83
 0.84 0.85 0.86 0.87 0.88 0.89 0.9  0.91 0.92 0.93 0.94 0.95 0.96 0.97
 0.98 0.99 1.  ]
[-1.    0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1 ]
```

[74]:
```
# Mais on peut faire des copies:
c[0] = 0
a = np.zeros(c.shape)
# ou
#a[:] = c
a[...] = c
c[0] = 2
print(a[:10])
```

```
[ 0.   -1.    0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09]
```

### 1.7 Index Conditionnels

[75]:
```
print(a,a.shape)
ia1 = np.argwhere(a>0.55)
ia2 = np.argwhere((a>.55) | (a<0.2))
print(ia2.T)
print(ia2.shape)
```

```
[ 0.   -1.    0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1   0.11
  0.12  0.13  0.14  0.15  0.16  0.17  0.18  0.19  0.2   0.21  0.22  0.23
  0.24  0.25  0.26  0.27  0.28  0.29  0.3   0.31  0.32  0.33  0.34  0.35
  0.36  0.37  0.38  0.39  0.4   0.41  0.42  0.43  0.44  0.45  0.46  0.47
  0.48  0.49  0.5   0.51  0.52  0.53  0.54  0.55  0.56  0.57  0.58  0.59
  0.6   0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69  0.7   0.71
  0.72  0.73  0.74  0.75  0.76  0.77  0.78  0.79  0.8   0.81  0.82  0.83
  0.84  0.85  0.86  0.87  0.88  0.89  0.9   0.91  0.92  0.93  0.94  0.95
  0.96  0.97  0.98  0.99  1.  ] (101,)
[[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
   18  19  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
   72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
   90  91  92  93  94  95  96  97  98  99 100]]
(65, 1)
```

[76]:
```
print(a[ia2].shape)
#print(a[ia2])
print(a[ia1].ravel())
print(a[ia2].ravel())
```

```
(65, 1)
[0.56 0.57 0.58 0.59 0.6  0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69
 0.7  0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8  0.81 0.82 0.83
 0.84 0.85 0.86 0.87 0.88 0.89 0.9  0.91 0.92 0.93 0.94 0.95 0.96 0.97
 0.98 0.99 1.  ]
[ 0.   -1.    0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1   0.11
  0.12  0.13  0.14  0.15  0.16  0.17  0.18  0.19  0.56  0.57  0.58  0.59
  0.6   0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69  0.7   0.71
  0.72  0.73  0.74  0.75  0.76  0.77  0.78  0.79  0.8   0.81  0.82  0.83
  0.84  0.85  0.86  0.87  0.88  0.89  0.9   0.91  0.92  0.93  0.94  0.95
  0.96  0.97  0.98  0.99  1.  ]
```

[77]:
```
print(a[(a<0.2) | (a>0.55)])
```

```
[ 0.   -1.    0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1   0.11
  0.12  0.13  0.14  0.15  0.16  0.17  0.18  0.19  0.56  0.57  0.58  0.59
  0.6   0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69  0.7   0.71
  0.72  0.73  0.74  0.75  0.76  0.77  0.78  0.79  0.8   0.81  0.82  0.83
  0.84  0.85  0.86  0.87  0.88  0.89  0.9   0.91  0.92  0.93  0.94  0.95
  0.96  0.97  0.98  0.99  1.  ]
```

[26]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

### 1.8 Exemples d'utilisation : calcul vectoriel

[78]:
```
a = np.arange(50)
print(a)
a = 2*a
a = a*a
print(a)
a.sum()
```

```
[ 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 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49]
[   0    4   16   36   64  100  144  196  256  324  400  484  576  676
  784  900 1024 1156 1296 1444 1600 1764 1936 2116 2304 2500 2704 2916
 3136 3364 3600 3844 4096 4356 4624 4900 5184 5476 5776 6084 6400 6724
 7056 7396 7744 8100 8464 8836 9216 9604]
```

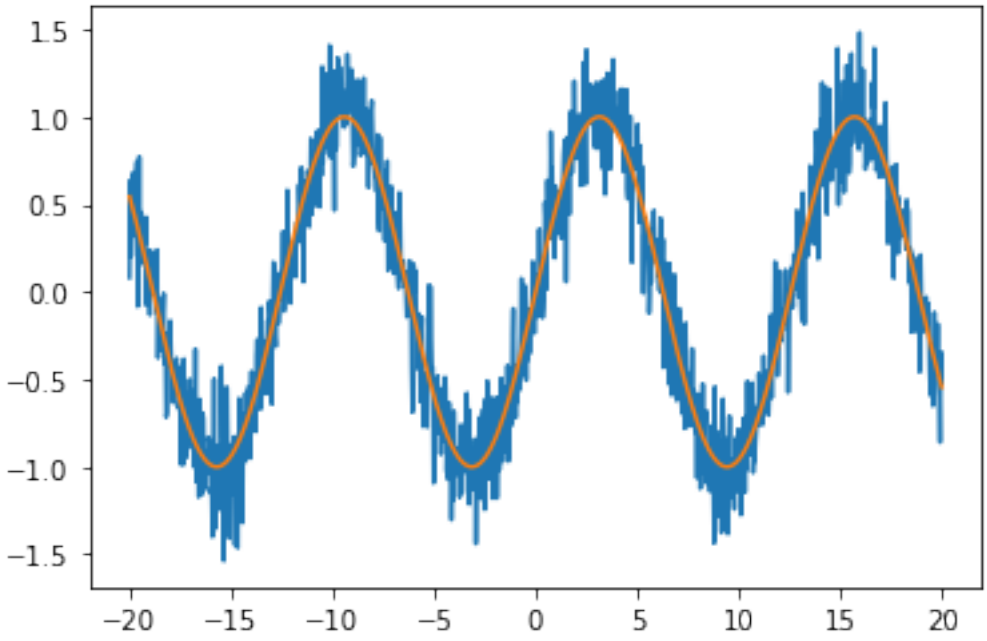
[78]:
```
161700
```

Pratique aussi pour calculer/tracer des fonctions numériques :

[81]:
```
%pylab inline
import pylab
from numpy import sin

x = np.linspace(-20,20,1000)
a0 = sin(x/2.)
a = a0 + np.random.normal(scale=0.2,size=1000)
f = pylab.plot(x,a)
g = pylab.plot(x,a0)
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
/anaconda3/envs/visual/lib/python3.8/site-
packages/IPython/core/magics/pylab.py:159: UserWarning: pylab import has
clobbered these variables: ['f', 'pylab']
`%matplotlib` prevents importing * from pylab and numpy
  warn("pylab import has clobbered these variables: %s"  % clobbered +
```




Faisons une moyenne glissante: on remplace chaque point de a par la moyenne des valeurs voisines:

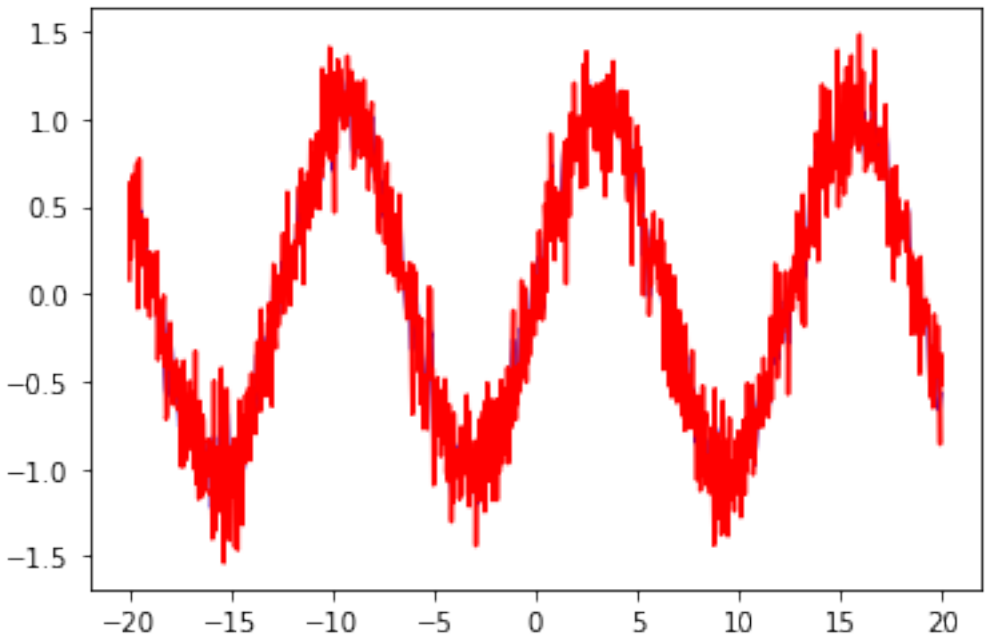
$$\frac{(a_{i-1} + a_i + a_{i+1})}{3}$$

$a_0 a_1 a_2 a_3 ... a_{n-3} a_{n-2}$

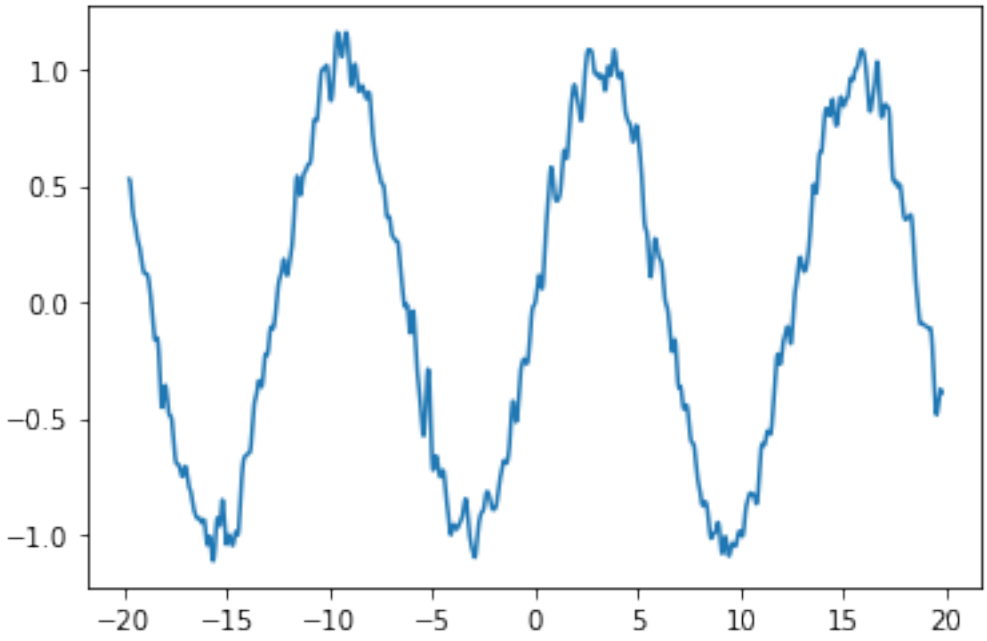
$a_1 a_2 a_3 a_4 ... a_{n-2} a_{n-1}$ un élement en moins

$a_2 a_3 a_4 a_5 ... a_{n-1} a_n$ 2 éléments en moins

[82]:
```
# Directement en vectoriel !
c = np.copy(a[1:-1])
c += a[:-2] + a[2:]
c *= 1./3

#c = (a[:-2]+a[-1:1]+a[2:])/3

# on exclut les bornes, mais d'autres choix sont possibles
f1 = pylab.plot(x[1:-1],c,"b")
f2 = pylab.plot(x,a,"r")
```




[87]:
```
# avec une fonction
def smooth3(a):
    c = np.zeros(len(a)-2)
    c += a[:-2] + a[1:-1] + a[2:]
    c *= 1./3
    return c

c = smooth3(a)
for i in range(5):
    c = smooth3(c)
    #pylab.plot(c)

#pylab.ylim((0.0,1.0))
f = pylab.plot(x[6:-6],c)
#g = pylab.plot(x,a)
```




en fait: opération plus générale (convolution) -> existe dans scipy: convolve, et smooth

### 1.9 Règles de composition (“broadcasting”)

[33]:
```
a = np.arange(100)
a = a.reshape(10,10)
a
```

[33]:
```
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, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

[34]:
```
# l'addition "comprend" comment composer des dimensions différentes
a+10
```

[34]:
```
array([[ 10,  11,  12,  13,  14,  15,  16,  17,  18,  19],
       [ 20,  21,  22,  23,  24,  25,  26,  27,  28,  29],
       [ 30,  31,  32,  33,  34,  35,  36,  37,  38,  39],
```

```
       [ 40,  41,  42,  43,  44,  45,  46,  47,  48,  49],
       [ 50,  51,  52,  53,  54,  55,  56,  57,  58,  59],
       [ 60,  61,  62,  63,  64,  65,  66,  67,  68,  69],
       [ 70,  71,  72,  73,  74,  75,  76,  77,  78,  79],
       [ 80,  81,  82,  83,  84,  85,  86,  87,  88,  89],
       [ 90,  91,  92,  93,  94,  95,  96,  97,  98,  99],
       [100, 101, 102, 103, 104, 105, 106, 107, 108, 109]])
```

[35]:
```
a[:2,0:3]
```

[35]:
```
array([[ 0,  1,  2],
       [10, 11, 12]])
```

[36]:
```
b = np.arange(10)-5
print(b)
```

```
[-5 -4 -3 -2 -1  0  1  2  3  4]
```

[37]:
```
print(a)
a+b
```

```
[[ 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 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

[37]:
```
array([[ -5,  -3,  -1,   1,   3,   5,   7,   9,  11,  13],
       [  5,   7,   9,  11,  13,  15,  17,  19,  21,  23],
       [ 15,  17,  19,  21,  23,  25,  27,  29,  31,  33],
       [ 25,  27,  29,  31,  33,  35,  37,  39,  41,  43],
       [ 35,  37,  39,  41,  43,  45,  47,  49,  51,  53],
       [ 45,  47,  49,  51,  53,  55,  57,  59,  61,  63],
       [ 55,  57,  59,  61,  63,  65,  67,  69,  71,  73],
       [ 65,  67,  69,  71,  73,  75,  77,  79,  81,  83],
       [ 75,  77,  79,  81,  83,  85,  87,  89,  91,  93],
       [ 85,  87,  89,  91,  93,  95,  97,  99, 101, 103]])
```

[38]:
```
a = np.eye(5)
a[1,2] = 1
print("la matrice de départ\n",a)
print()
```

```
print("sous bloc 1\n",a[0:-2,0:-2] )
print()
print("sous bloc 2\n",a[1:-1,1:-1])
print("la somme")
a[0:-2,0:-2] + a[1:-1,1:-1]
```

```
la matrice de départ
 [[1. 0. 0. 0. 0.]
 [0. 1. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]

sous bloc 1
 [[1. 0. 0.]
 [0. 1. 1.]
 [0. 0. 1.]]

sous bloc 2
 [[1. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
la somme
```

[38]:
```
array([[2., 1., 0.],
       [0., 2., 1.],
       [0., 0., 2.]])
```

### 1.10 Calculs / grilles en 2D

[89]:
```
x = np.linspace(0.,5.,3)
y = np.linspace(0.,3.,4)
(X,Y) = np.meshgrid(x,y)
print(X)
print()
print(Y)
```

```
[[0.  2.5 5. ]
 [0.  2.5 5. ]
 [0.  2.5 5. ]
 [0.  2.5 5. ]]

[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]
 [3. 3. 3.]]
```

[90]:
```
a = X*X + Y*Y
a
```

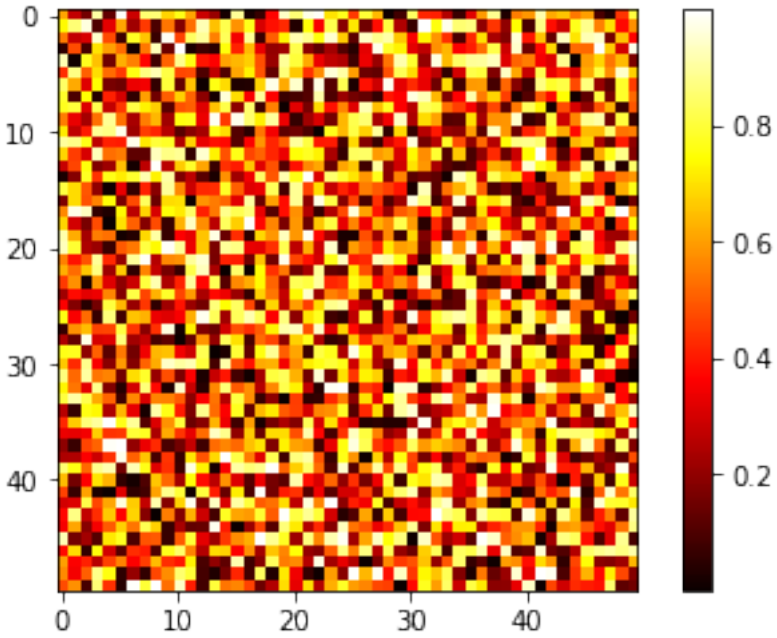
[90]:
```
array([[ 0.  ,  6.25, 25.  ],
       [ 1.  ,  7.25, 26.  ],
       [ 4.  , 10.25, 29.  ],
       [ 9.  , 15.25, 34.  ]])
```

Traitement du signal en python : Scipy, une surcouche de numpy

### 1.11 Image et matrice

[97]:
```
image = np.random.rand(50, 50)
plt.imshow(image, cmap=plt.cm.hot)
plt.colorbar()
```

[97]:
```
<matplotlib.colorbar.Colorbar at 0x11ca80850>
```

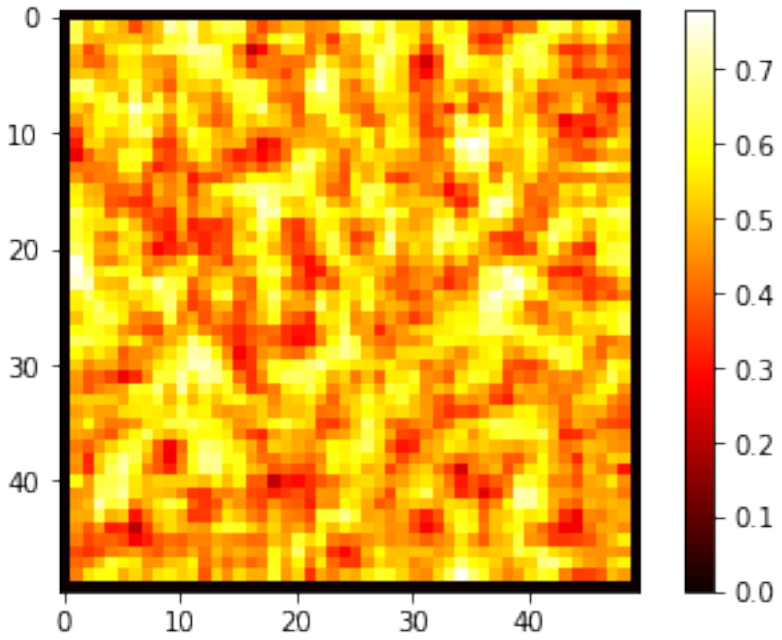



[98]:
```
def smooth(image):
    new = np.zeros(image.shape)
    Z = image
    new[1:-1,1:-1] = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
                      Z[1:-1,0:-2] + Z[1:-1,1:-1] + Z[1:-1,2:] +
                      Z[2:  ,0:-2] + Z[2:  ,1:-1] + Z[2:  ,2:])/9.
```

```
    return new
```

[46]:
```
new = smooth(image)

plt.imshow(new, cmap=plt.cm.hot)
plt.colorbar()
```

[46]:
```
<matplotlib.colorbar.Colorbar at 0x11c839880>
```




[47]:
```
filtre = np.ones((3,3))/9
```

[48]:
```
filtre, image[0:3,0:3]
```

[48]:
```
(array([[0.11111111, 0.11111111, 0.11111111],
        [0.11111111, 0.11111111, 0.11111111],
        [0.11111111, 0.11111111, 0.11111111]]),
 array([[0.89071463, 0.36519483, 0.44680401],
        [0.52499459, 0.70864701, 0.90626983],
        [0.89795117, 0.3145517 , 0.71820157]]))
```

[49]:
```
filtre*image[0:3,0:3]
```

[49]:
```
array([[0.09896829, 0.0405772 , 0.04964489],
       [0.05833273, 0.07873856, 0.10069665],
       [0.09977235, 0.03495019, 0.07980017]])
```

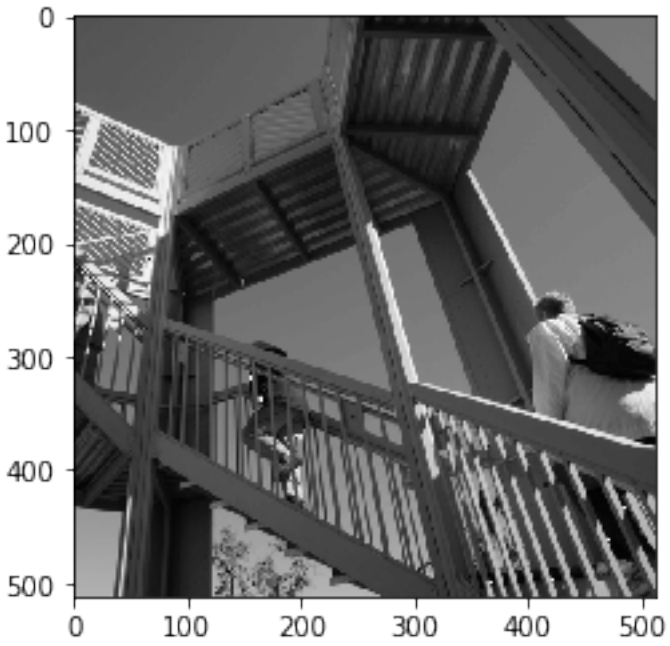
[50]:
```
weights = filtre
```

[51]:
```
weights
```

[51]:
```
array([[0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111]])
```

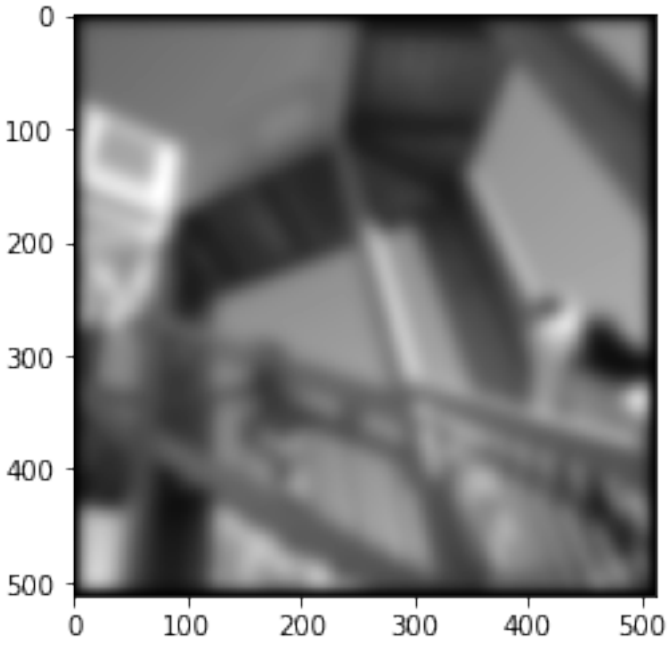
[99]:
```
from scipy import misc
example = misc.ascent()
plt.imshow(example,cmap=plt.cm.gray)
print(type(example))
print(example.shape, example.dtype)
```

```
<class 'numpy.ndarray'>
(512, 512) int64
```




[53]:
```
new = example
for i in range(100):
    new = smooth(new)
plt.imshow(new,cmap=plt.cm.gray)
```

[53]:
```
<matplotlib.image.AxesImage at 0x11d069520>
```




Plusieurs librairies python (ou avec interface python) pour l'image

- PIL python imaging library : manipulation de base des formats
- opencv : traitements avancés / vision par ordinateur (originellement en C++) / reconnaissance des formes http://opencv.org
- scikit.image

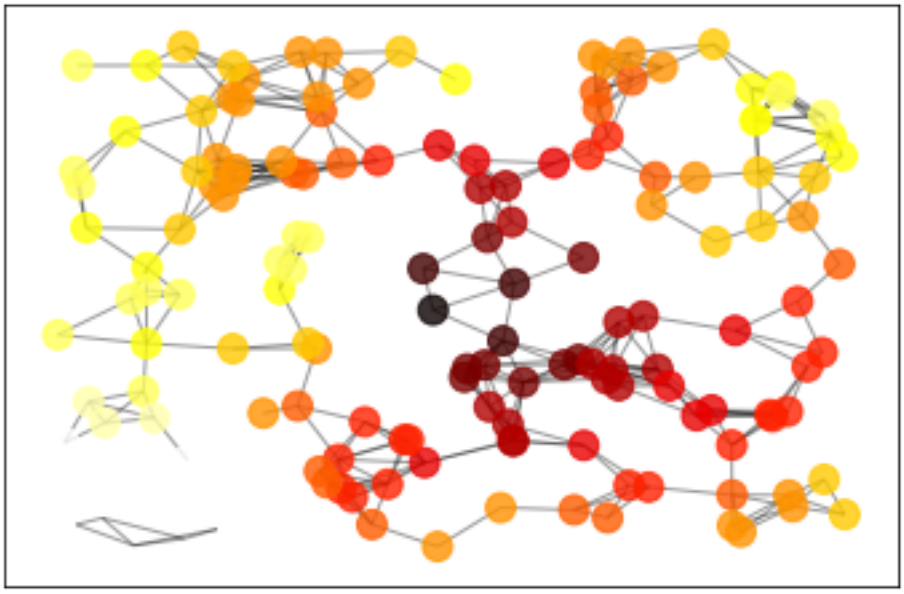
### 1.12 Graphes

plusieurs librairies disponibles

- networkx (pur python)
- igraph (C avec API python)
- matrices creuses en numpy

[55]:
```
#Exemple

import networkx as nx

# graph aléatoire
G = nx.random_geometric_graph(150, 0.12)
pos = nx.get_node_attributes(G, 'pos')
#  positions des noeuds,
# -> trouver le noeud le plus près du centre à (0.5,0.5)
dists = [(x - 0.5)**2 + (y - 0.5)**2 for x, y in pos.values()]
ncenter = np.argmin(dists)
```

```
# calculer la longueur des chemins depuis le centre
p = nx.single_source_shortest_path_length(G, ncenter)

# faire la figure: couleur reliée à la longueur des chemins
pylab.figure()
nx.draw_networkx_edges(G, pos, alpha=0.4)
nx.draw_networkx_nodes(G, pos, nodelist=list(p.keys()),
                       node_size=120, alpha=0.8,
                       node_color=list(p.values()), cmap=pylab.cm.hot)
pylab.show()
```



[50]:
```
# exemple igraph
# exemple numpy
```

### 1.13 Apprentissage automatique / reconnaissance des formes

dès le prochain semestre

A partir d'exemples, construire un modèle qui prend des décisions sur des données nouvelles

- classification : déterminer la catégorie d'un événement; exemple détection de spam
- régression : prédire des valeurs d'une fonction à partir d'exemple; exemple:
- apprendre des politiques d'action pour un robot

scikit-learn (surcouche de numpy) http://scikit-learn.org

pytorch, tensorflow, keras, etc: réseaux de neurones en folie

opencv a aussi des outils reliés, spécialisés pour le traitement d'image

### 1.14 Robotique

ROS: robot operating system, utilisé dans les TP de robotique.

http://wiki.ros.org/

- gestion du hardware
- visualisation
- communication entre modules

avec une API python

[ ]: