Python for DataScience TD3

Partie 1: Quiz nowledgeable

Partie 2: Retour sur Python basics

Partie 3: numpy

CI-dessous deux cheat sheets très intéressantes sur numpy et scipy:

http://datacamp-community-prod.s3.amazonaws.com/ba1fe95a-8b70-4d2f-95b0-bc954e9071b0

http://datacamp-community-prod.s3.amazonaws.com/dfdb6d58-e044-4b38-bab3-5de0b825909b

Attention : c'est une bonne chose d'avoir des résumés ou des cheat sheets à notre disposition, mais si on veut aller loin en data science et machine learning il faut s'entraîner au maximum pour y avoir recours au minimum.

Notez que certains sous modules existent dans numpy et scipy en particulier pour l'algèbre linéaire par exemple le sous module **linalg** existe dans numpy et scipy mais celui de scipy est plus élargi.

Exo 1 : mini introduction de numpy

Essayez à votre manière de découvrire (ne pas oublier dir, pour le module et pour les classes et méthodes)

```
import numpy as np
```

Tester le code suivant :

```
In [18]: a=np.arange(10)

In [19]: a
Out[19]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [20]: print(a)
[0 1 2 3 4 5 6 7 8 9]

In [21]: type(a)
Out[21]: numpy.ndarray

In [22]: L=np.arange(2,15,3)

In [23]: print(L)
[ 2  5  8 11 14]

In [24]: p1=np.array([1,2,3])

In [25]: p1
Out[25]: array([1, 2, 3])

In [26]: print(p1)
[1 2 3]

In [27]: type(p1)
Out[27]: numpy.ndarray

In [28]: l2=[10,12,13]

In [29]: l1=[1,2,3]

In [30]: l1+l2
Out[30]: [1, 2, 3, 10, 12, 13]

In [31]: p2=np.array(l2)

In [32]: p1+p2
Out[32]: array([11, 14, 16])
```

Accordez une importance à p1+p2 (comparer avec l1+l2), pour les listes on aurait dû faire une boucle pour additionner les éléments 1 à 1. Il s'agit d'une vectorisation en numpy.

```
/import numpy as np
l1=list(range(10000))


>>>import numpy as np
>>>l1=list(range(10000))
>>>l2=list(range(10000))
>>>a1=np.array(range(10000))
>>>a2=np.array(range(10000))
>>>%timeit [l1[i]+l2[i] for i in range(10000)]
1.4 ms ± 22.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
>>>%timeit list(map(lambda x,y:x+y,l1,l2))
```

```
1.27 ms ± 32.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
>>>%timeit a1+a2
7.23 µs ± 37.1 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

10 000 opérations en 7µs contre 1.4ms (soit 1400µs)

Exo 2 : **arange et linspace**

Accédez à la documentation de arange (help(np.arange) ou tout simplement (np.arange?))

Accédez à la documentation de linspace (help(np.linspace) ou tout simplement (np.linspace?))

Observez la différence.

Exo 3 : **Complétez ce code pour tracer sin(x), pour x entre 0 et 2π, avec 10 échantillons.**

```
import matplotlib.pyplot as plt
import numpy as np

x=...
y=...
plt.plot(x,y, 'ro-')
plt.show()
```

Astuce: *Pour afficher le plot dans une fenêtre séparée sous spyder :*
Tapez ***%matplotlib auto*** *dans l'interpréteur python, pour rechanger tapez* ***%matplotlib inline***

Exo 4 : **Complétez ce code pour tracer sin(x), pour x entre 0 et 2π, avec 0.2 comme pas d'échantillonnage.**

```
import matplotlib.pyplot as plt
import numpy as np

x=...
y=...
plt.plot(x,y, 'ro-')
plt.show()
```

Astuce: *Pour afficher le plot dans une fenêtre séparée sous spyder :*
Tapez ***%matplotlib auto*** *dans l'interpréteur python, pour rechanger tapez* ***%matplotlib inline***

Partie 4: nowledgeable https://nowledgeable.com