Introduction à la Programmation des Algorithmes

6.3. Python – Fichiers et modules

François Fleuret

https://fleuret.org/11x001/

UNIVERSITÉ
DE GENÈVE

Fichiers

Python offre de manière native, c'est à dire intégrées au langage, des fonctions de manipulation de fichiers similaires à `stdio.h` en C.

Un fichier Python ouvert en lecture est itérable, et parcours les lignes du fichier les unes après les autres.

```python
f = open('machin.txt', 'w')
f.write('Une première ligne\n')
f.write('Une seconde ligne\n')
f.close()

g = open('machin.txt', 'a')
for k in range(3, 6):
    g.write(f'ligne {k}\n')
g.close()

f = open('machin.txt', 'r')
for l in f:
    print(l.strip())
f.close()
```

affiche

```
Une première ligne
Une seconde ligne
ligne 3
ligne 4
ligne 5
```

Les erreurs de manipulations de fichiers sont gérées avec des exceptions.

```
>>> f = open('/truc', 'w')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
PermissionError: [Errno 13] Permission denied: '/truc'
>>> f = open('nexistepas.dat', 'r')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'nexistepas.dat'
```

Il arrive souvent qu'une partie d'un programme puisse être structurée avec la notion de **contexte**, qui correspond à un certain état dans lequel des opérations sont possibles, et dans lequel on entre et d'où on sort avec certaines opérations.

La manipulation d'un fichier correspond bien à cette situation: l'ouverture du fichier permet de se placer dans un contexte où on peut faire de opérations de lecture ou d'écriture, et on sort du contexte en fermant le fichier.

Il arrive souvent qu'une partie d'un programme puisse être structurée avec la notion de **contexte**, qui correspond à un certain état dans lequel des opérations sont possibles, et dans lequel on entre et d'où on sort avec certaines opérations.

La manipulation d'un fichier correspond bien à cette situation: l'ouverture du fichier permet de se placer dans un contexte où on peut faire de opérations de lecture ou d'écriture, et on sort du contexte en fermant le fichier.

Python offre le mot clé `with` qui permet de se placer dans un contexte. Ce mot clé doit être suivi d'un objet capable de traiter l'entrée et la sortie du contexte.

La seule utilisation que nous en ferons dans ce cours sera pour manipuler des fichiers.

```python
with open('machin.txt', 'w') as f:
    f.write('Une première ligne\n')
    f.write('Une seconde ligne\n')

with open('machin.txt', 'a') as f:
    for k in range(3, 6):
        f.write(f'ligne {k}\n')

with open('machin.txt', 'r') as f:
    for l in f:
        print(l.strip())
```

affiche

```
Une première ligne
Une seconde ligne
ligne 3
ligne 4
ligne 5
```

On peut combiner l'utilisation des contextes et la gestion des exceptions:

```
1  try:
2      with open('nexistepas.txt', 'r') as f:
3          for l in f:
4              print(l.strip())
5
6  except FileNotFoundError as e:
7      print(f'Le fichier {e.filename} est introuvable.')
```

affiche

```
Le fichier nexistepas.txt est introuvable.
```

Modules

Il est possible de structurer un long programme Python en plusieurs fichiers, et plus généralement de regrouper des classes, des variables, et des fonctions relatives à un type de données ou de problèmes spécifiques.

On peut utiliser des éléments définis dans un autre fichier à l'aide de l'instruction `import`, qui exécute un autre fichier.

Les identifiants créés durant cette exécution sont mémorisés de façon à pouvoir y accéder en les préfixant avec le nom du module, et à éviter ainsi les collisions de noms.

Si nous avons un fichier `truc.py` dont le contenu est

```
1  def bidule(n):
2      for k in range(n):
3          print(f'bidule-{k}')
```

Si nous avons un fichier `truc.py` dont le contenu est

```
1  def bidule(n):
2      for k in range(n):
3          print(f'bidule-{k}')
```

alors le programme Python dans le même répertoire

```
1  import truc
2
3  truc.bidule(4)
```

affiche

```
bidule-0
bidule-1
bidule-2
bidule-3
```

Ici l'identifiant `bidule` est défini dans l'**espace de nommage** du module `truc`.

Dans la forme la plus simple, la syntaxe est donc

```
import nom_1, nom_2, ..., nom_K
```

où les `nom_k` sont des noms de **modules**, qui sont des noms de fichiers sans l'extension `.py`.

Il est ensuite possible de faire référence aux fonctions, classes et variables définies dans ces fichiers en préfixant leurs identifiants avec le nom du module.

On peut également importer un élément particulier. Il est alors dans l'espace de nommage normal et on peut y faire référence sans le nom du module.

```
1  from truc import bidule
2
3  bidule(4)
```

On peut également importer un élément particulier. Il est alors dans l'espace de nommage normal et on peut y faire référence sans le nom du module.

```
1  from truc import bidule
2
3  bidule(4)
```

Et il est possible d'importer des modules ou des éléments de modules en aliasant leur noms, dans un but de simplicité ou pour éviter des collisions d'identifiants.

```
1  import truc as tr
2
3  tr.bidule(4)
```

ou

```
1  from truc import bidule as chose
2
3  chose(4)
```

En général un module ne faire que définir des classes, variables et fonctions, mais n'exécute pas de code.

Il peut arriver que l'on veuille qu'un module puisse être exécuté, par exemple pour faire des tests.

En général un module ne faire que définir des classes, variables et fonctions, mais n'exécute pas de code.

Il peut arriver que l'on veuille qu'un module puisse être exécuté, par exemple pour faire des tests.

On peut tester la variable `__name__` qui contient le nom du module s'il est exécuté via un `import` et `__main__` s'il est exécuté directement.

```
1  def bidule(n):
2      for k in range(n):
3          print(f'bidule-{k}')
4
5  if __name__ == '__main__':
6      bidule(3)
```

affiche

```
bidule-0
bidule-1
bidule-2
```

Il existe un très (TRÈS!) grand nombre de modules Python disponibles pour réaliser des tâches de manipulations de fichiers, créations graphiques, opérations réseaux, calculs mathématiques, etc.

Un exemple simple est la récupération d'un fichier sur le web.

Il existe un très (TRÈS!) grand nombre de modules Python disponibles pour réaliser des tâches de manipulations de fichiers, créations graphiques, opérations réseaux, calculs mathématiques, etc.

Un exemple simple est la récupération d'un fichier sur le web.

Récupérer des données depuis un site distant est souvent pratique mais pose un certains nombre de problèmes.

- Le programme ne fonctionne qu'avec une connexion réseau.
- Cela peut entraîner des coûts et de l'engorgement pour le serveur web correspondant.
- La durée de vie du programme est limitée par celle du document web utilisé.

Accès et manipulations de fichiers distants

```python
from urllib.request import urlopen

url = 'https://api.blockchain.info/charts/market-price?format=csv'

with urlopen(url) as f:
    l = list(f)

for s in l[-5:]:
    print(s)
```

affiche

```
b'2021-10-19 00:00:00,61971.59\n'
b'2021-10-20 00:00:00,64287.64\n'
b'2021-10-21 00:00:00,66063.56\n'
b'2021-10-22 00:00:00,62354.86\n'
b'2021-10-23 00:00:00,60697.06\n'
```

Ce sont des chaînes binaires, qu'il faut convertir en chaînes de caractères avec `decode()`. On peut également supprimer le `\n` final avec `strip()` et les découper en champ séparés par des virgules avec `split(',')`.

```python
from urllib.request import urlopen

url = 'https://api.blockchain.info/charts/market-price?format=csv'

with urlopen(url) as f:
    l = list(f)

for s in l[-5:]:
    x = s.decode().strip().split(',')
    print(f'{x[0]} -> ${x[1]}')
```

affiche

```
2021-10-19 00:00:00 -> $61971.59
2021-10-20 00:00:00 -> $64287.64
2021-10-21 00:00:00 -> $66063.56
2021-10-22 00:00:00 -> $62354.86
2021-10-23 00:00:00 -> $60697.06
```

La gestion des dates est complexe à faire correctement. On peut pour cela utiliser le module `datetime`.

```
>>> import datetime
>>> datetime.date(2019, 4, 15)
datetime.date(2019, 4, 15)
>>> t = datetime.datetime.now()
>>> t
datetime.datetime(2021, 10, 24, 13, 24, 21, 636715)
>>> t.year
2021
>>> t.month
10
>>> t.timetuple()
time.struct_time(tm_year=2021, tm_mon=10, tm_mday=24, tm_hour=13,
   tm_min=24, tm_sec=21, tm_wday=6, tm_yday=297, tm_isdst=-1)
>>> t0 = datetime.datetime.fromisoformat('2011-11-04 14:05:00')
>>> t - t0
datetime.timedelta(days=3641, seconds=83961, microseconds=636715)
>>> datetime.datetime.strptime('Sun Oct 24 13:18:13 CEST 2021',
                               '%a %b %d %H:%M:%S %Z %Y')
datetime.datetime(2021, 10, 24, 13, 18, 13)
```

```python
from datetime import datetime
from urllib.request import urlopen

url = 'https://api.blockchain.info/charts/market-price?format=csv'

with urlopen(url) as f:
    l = list(f)

for s in l[-5:]:
    x = s.decode().strip().split(',')
    date = datetime.fromisoformat(x[0]).strftime('%d %b %Y')
    prix = x[1]
    print(f'{date} -> ${prix}')
```

affiche

```
20 Oct 2021 -> $64287.64
21 Oct 2021 -> $66063.56
22 Oct 2021 -> $62354.86
23 Oct 2021 -> $60697.06
24 Oct 2021 -> $61277.28
```

Le format très simple qui consiste en des lignes successives de champs séparés par des virgules est le CSV (*comma-separated values*)

Nous pouvons par exemple récupérer le nombre de cas de covid par région et pays sur

`https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv`

Python offre des outils pour manipuler facilement ce format.

```
Province/State,Country/Region,Lat,Long,1/22/20,1/23/20,1/24/20,1/25/20,1/26/20,1/27/20,1/28/20,1/29/20,1/30/20,
,Afghanistan,33.93911,67.709953,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,5,5,5,5,5,5
,Albania,41.1533,20.1683,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Algeria,28.0339,1.6596,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,3,5,12,
,Andorra,42.5063,1.5218,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1
,Angola,-11.2027,17.8739,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Antigua and Barbuda,17.0608,-61.7964,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,Argentina,-38.4161,-63.6167,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Armenia,40.0691,45.0382,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,
Australian Capital Territory,Australia,-35.4735,149.0124,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
New South Wales,Australia,-33.8688,151.2093,0,0,0,0,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4
Northern Territory,Australia,-12.4634,130.8456,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Queensland,Australia,-27.4698,153.0251,0,0,0,0,0,0,0,1,3,2,3,2,2,3,3,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
South Australia,Australia,-34.9285,138.6007,0,0,0,0,0,0,0,0,0,0,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
Tasmania,Australia,-42.8821,147.3272,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
Victoria,Australia,-37.8136,144.9631,0,0,0,0,1,1,1,1,2,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
Western Australia,Australia,-31.9505,115.8605,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,Austria,47.5162,14.5501,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,3,3,9,14,18,21
,Azerbaijan,40.1431,47.5769,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,3,3
,Bahamas,25.025885,-78.035889,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,Bahrain,26.0275,50.55,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,23,33,33,36,41,47,49
,Bangladesh,23.685,90.3563,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Barbados,13.1939,-59.5432,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Belarus,53.7098,27.9534,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,6,
,Belgium,50.8333,4.469936,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,8,13,
,Belize,17.1899,-88.4976,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Benin,9.3077,2.3158,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Bhutan,27.5142,90.4336,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,Bolivia,-16.2902,-63.5887,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Bosnia and Herzegovina,43.9159,17.6791,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,Botswana,-22.3285,24.6849,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,Brazil,-14.235,-51.9253,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,2,2,2,2,4,
,Brunei,4.5353,114.7277,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/20	2/7/20	2/8/20	2/9/20
2		Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3		Albania	41.1533	20.1683	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4		Algeria	28.0339	1.6596	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5		Andorra	42.5063	1.5218	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6		Angola	-11.2027	17.8739	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7		Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8		Argentina	-38.4161	-63.6167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9		Armenia	40.0691	45.0382	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	Northern Territory	Australia	-12.4634	130.8456	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	Queensland	Australia	-27.4698	153.0251	0	0	0	0	0	0	0	1	3	2	3	2	2	3	3	4	5	5	5
14	South Australia	Australia	-34.9285	138.6007	0	0	0	0	0	0	0	0	0	0	1	2	2	2	2	2	2	2	2
15	Tasmania	Australia	-42.8821	147.3272	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	Victoria	Australia	-37.8136	144.9631	0	0	0	0	1	1	1	1	2	3	4	4	4	4	4	4	4	4	4
17	Western Australia	Australia	-31.9505	115.8605	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18		Austria	47.5162	14.5501	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19		Azerbaijan	40.1431	47.5769	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20		Bahamas	25.025885	-78.035889	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21		Bahrain	26.0275	50.55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22		Bangladesh	23.685	90.3563	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23		Barbados	13.1939	-59.5432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24		Belarus	53.7098	27.9534	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25		Belgium	50.8333	4.469936	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
26		Belize	17.1899	-88.4976	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27		Benin	9.3077	2.3158	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28		Bhutan	27.5142	90.4336	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29		Bolivia	-16.2902	-63.5887	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30		Bosnia and Herzegovina	43.9159	17.6791	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31		Botswana	-22.3285	24.6849	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32		Brazil	-14.235	-51.9253	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33		Brunei	4.5353	114.7277	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34		Bulgaria	42.7339	25.4858	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35		Burkina Faso	12.2383	-1.5616	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36		Burma	21.9162	95.956	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37		Burundi	-3.3731	29.9189	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38		Cabo Verde	16.5388	-23.0418	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39		Cambodia	11.55	104.9167	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
40		Cameroon	3.848	11.5021	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	Alberta	Canada	53.9333	-116.5765	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	British Columbia	Canada	53.7267	-127.6476	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	4	4	4
43	Diamond Princess	Canada	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	Grand Princess	Canada	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	Manitoba	Canada	53.7609	-98.8139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	New Brunswick	Canada	46.5653	-66.4619	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	Newfoundland and Labrador	Canada	53.1355	-57.6604	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	Northwest Territories	Canada	64.8255	-124.8457	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```python
import csv

with open('time_series_covid19_confirmed_global.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row_nb, row in enumerate(reader):
        if row_nb == 0 or row[1] in { 'Switzerland', 'Austria' }:
            print(row[1], row[-4:])
```

affiche

```
Country/Region ['10/20/21', '10/21/21', '10/22/21', '10/23/21']
Austria ['783996', '787644', '791226', '794982']
Switzerland ['859646', '861123', '862411', '862411']
```

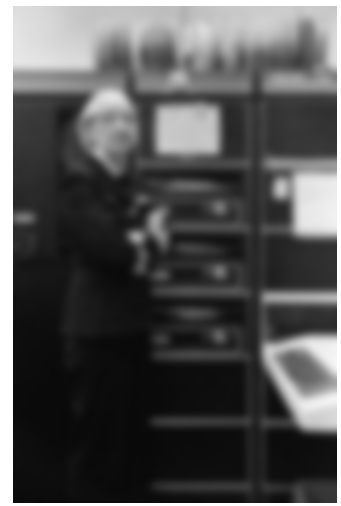
Manipulations d'images

```python
from PIL import Image, ImageFilter

im = Image.open('GraceHopper.png')

new_im = im.transpose(Image.ROTATE_90)
new_im.save('GraceHopper_r90.png')
```

```python
from PIL import Image, ImageFilter

im = Image.open('GraceHopper.png')

new_im = im.resize((im.width // 4, im.height // 4))
new_im.save('GraceHopper_resize.png')
```

```python
from PIL import Image, ImageFilter

im = Image.open('GraceHopper.png')

for k in { 5, 10, 15 }:
    new_im = im.filter(ImageFilter.GaussianBlur(k))
    new_im.save(f'GraceHopper_blurry_{k}.png')
```






Numpy, SciPy et Matplotlib

Grâce à sa souplesse d'utilisation et aux notebooks interactifs, Python a remplacé en grande partie des outils utilisés en sciences expérimentales tels que R ou Matlab.

Trois importantes librairies qui contribuent à ce succès sont Numpy pour le calcul algébrique rapide, SciPy pour le calcul scientifique et Matplotlib pour les représentation graphiques.

Python est un langage très lent et programmer une opération telle qu'un produit de matrice serait horriblement inefficace.

NumPy offre une très large palette d'opérations qui manipulent des matrices multi-dimensionnelles `ndarray`.

Cette structure permet en particulier de représenter des vecteurs classiques, dont les coefficients sont indexés par un seul indexe, et les matrices, avec deux indexes.

```
1  import numpy
2
3  m = numpy.array([[ 1, 1 ], [ -1, 1 ]])
4
5  v = numpy.array([ 2, 3 ])
6
7  print(v, m @ v)
```

affiche

```
[2 3] [5 1]
```

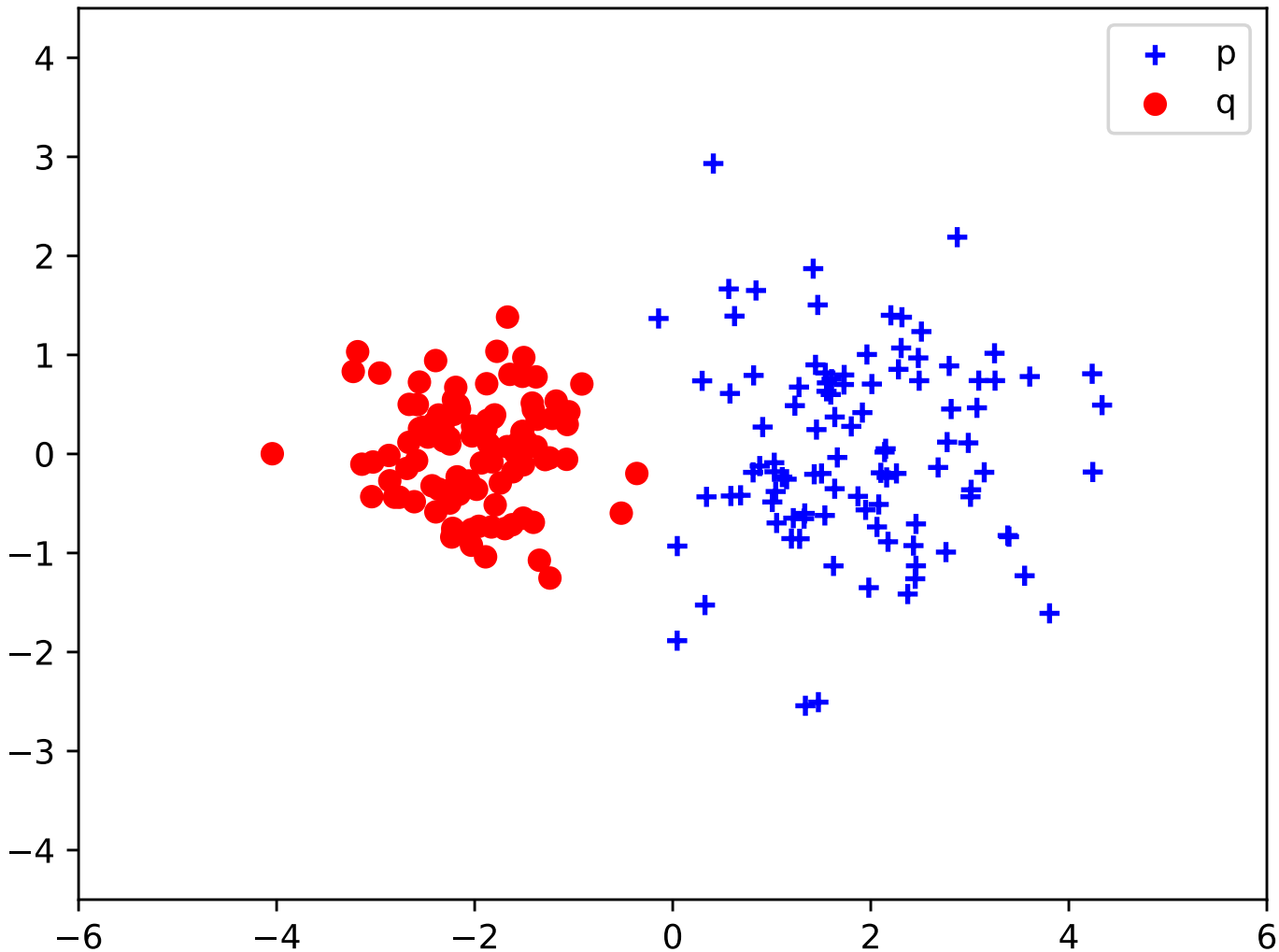
```
>>> m = numpy.random.normal(0, 1, (4, 4))
>>> m
array([[-0.82999735, -0.91818592, -0.96424688,  2.14184176],
       [-1.92361767, -0.66849122, -0.96641268, -0.14764259],
       [ 1.31496295, -0.20098614,  0.08986932, -0.1929336 ],
       [-0.42590301,  1.00492111,  0.13936688, -1.35530879]])
>>> m[2]
array([ 1.31496295, -0.20098614,  0.08986932, -0.1929336 ])
>>> m[:, 0]
array([-0.82999735, -1.92361767,  1.31496295, -0.42590301])
>>> m = m @ m.transpose()
>>> numpy.linalg.eig(m)[0]
array([10.76422836,  5.53350223,  0.1160834 ,  0.60071016])
```

```
>>> p = numpy.random.normal(0, 1, (1000, 1000))
>>> q = numpy.random.normal(0, 1, (1000, 1000))
>>> p @ q
array([[ -6.59884453,   4.33408239, -12.62841631, ..., -15.51141445,
         48.06933436,  41.36263837],
       [  6.89179897,  14.16530801,  11.06033872, ...,  16.96517321,
         50.21274734,  14.58360869],
       [-16.24756087,  42.41229225,  18.44074437, ..., -23.76344462,
         46.5083391 , -10.35787684],
       ...,
       [  7.85340925,  12.83766931,  24.97148857, ...,  39.95634369,
        -13.55957854, -52.46306412],
       [-16.55927574, -20.93303248,  19.87050555, ...,   9.04560522,
          9.17857406, -17.38930732],
       [ 17.89102294, -43.35488196, -33.76625869, ...,  -4.99645913,
         25.33190269,   8.47263572]])
```

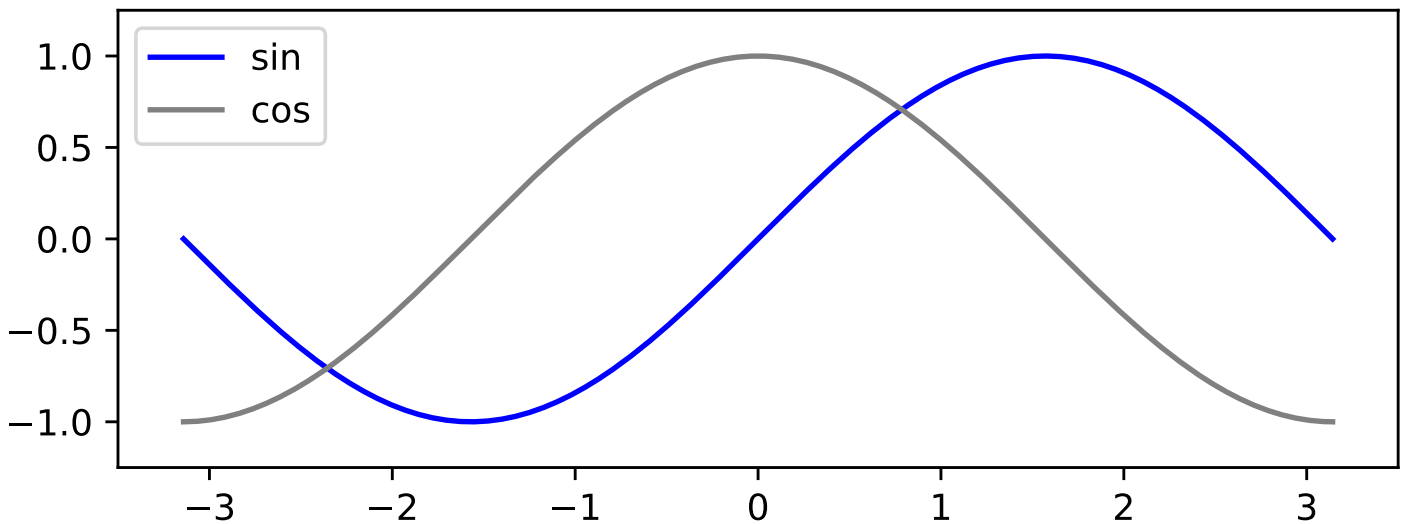
Ce calcul qui demande 10^9 multiplications est exécuté sur un ordinateur portable standard (Intel i7-8650U 1.90GHz) en 16ms. Soit 62.5 milliards de multiplications par seconde.

On peut combiner Numpy avec Matplotlib pour tracer des graphiques.

```
1  import matplotlib.pyplot as plt
2  import numpy
3
4  p = numpy.random.normal(0.0, 1.0, (100, 2)) + numpy.array([  2, 0 ])
5  q = numpy.random.normal(0.0, 0.5, (100, 2)) + numpy.array([ -2, 0 ])
6
7  fig = plt.figure()
8  subfig = fig.add_subplot()
9  subfig.set(xlim = (-6.0, 6.0), ylim = (-4.5, 4.5), aspect = 1)
10
11 subfig.scatter(p[:, 0], p[:, 1], marker = '+', color = 'blue', label = 'p')
12 subfig.scatter(q[:, 0], q[:, 1], marker = 'o', color = 'red', label = 'q')
13 subfig.legend()
14
15 plt.savefig('figure.pdf')
```

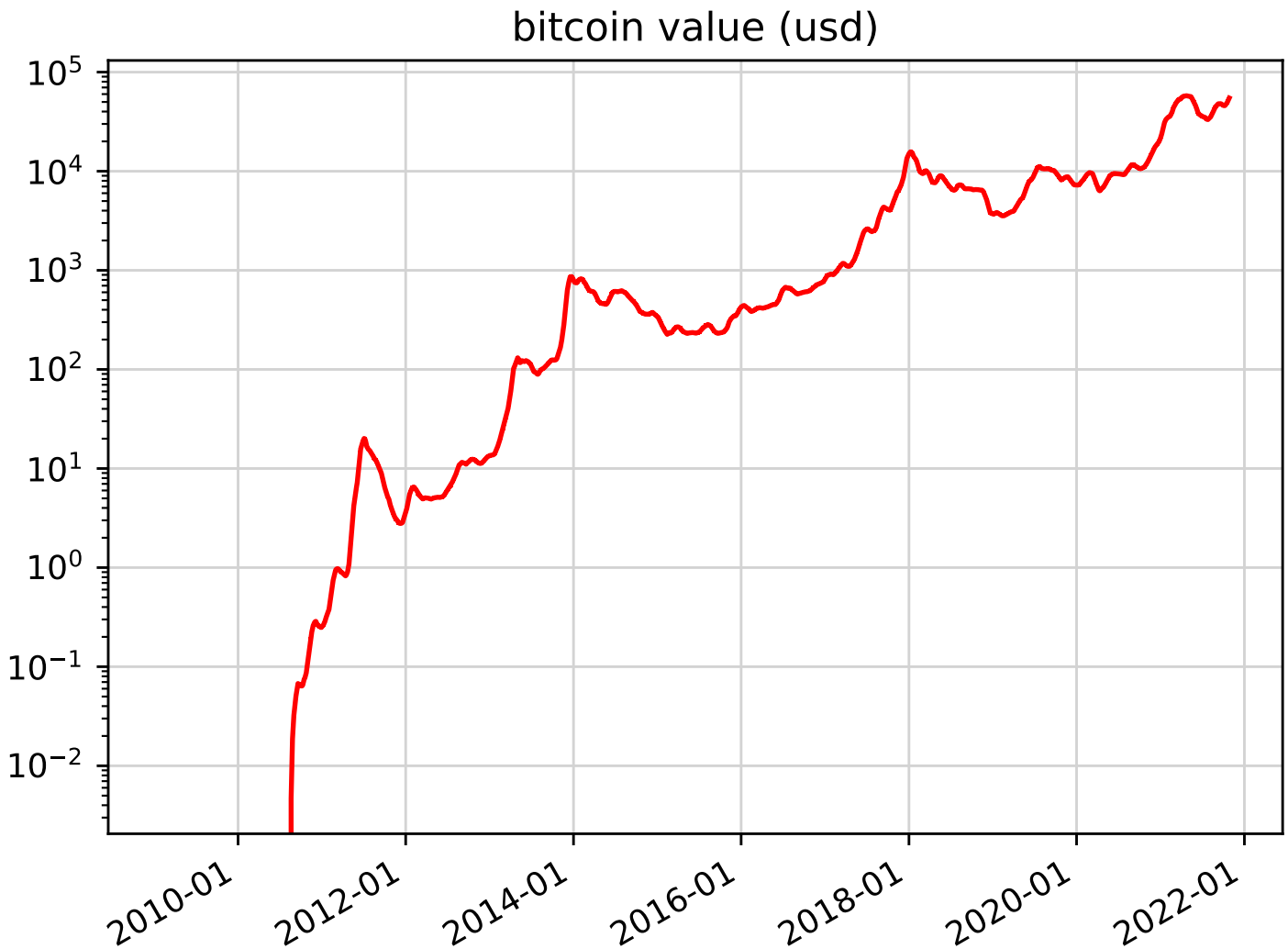

p
q
4
3
2
1
0
−1
−2
−3
−4
−6
−4
−2
0
2
4
6

```python
import matplotlib.pyplot as plt
import numpy, math

x = numpy.linspace(-math.pi, math.pi, 250)

fig = plt.figure()
subfig = fig.add_subplot()
subfig.set(xlim = (-3.5, 3.5), ylim = (-1.25, 1.25), aspect = 1)

subfig.plot(x, numpy.sin(x), color = 'blue', label = 'sin')
subfig.plot(x, numpy.cos(x), color = 'gray', label = 'cos')
subfig.legend()

plt.savefig('figure.pdf')
```


sin
cos
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3

```python
from datetime import datetime
from urllib.request import urlopen

url = 'https://api.blockchain.info/charts/market-price?\
daysAverageString=30D&timespan=all&sampled=true&\
metadata=false&cors=true&format=csv'

with urlopen(url) as f:
    l = list(f)

dates, prix = [], []

for s in l:
    x = s.decode().strip().split(',')
    dates.append(datetime.fromisoformat(x[0]))
    prix.append(float(x[1]))
```

```python
import matplotlib.pyplot as plt
import matplotlib.dates as dts

fig, subfig = plt.subplots()

subfig.set_title('bitcoin value (usd)')

date_form = dts.DateFormatter("%Y-%m")
subfig.xaxis.set_major_formatter(date_form)
fig.autofmt_xdate()
subfig.set_yscale('log')

subfig.grid(color = 'lightgray')
subfig.plot_date(dates, prix, fmt = ',-r')

plt.savefig('figure.pdf')
```

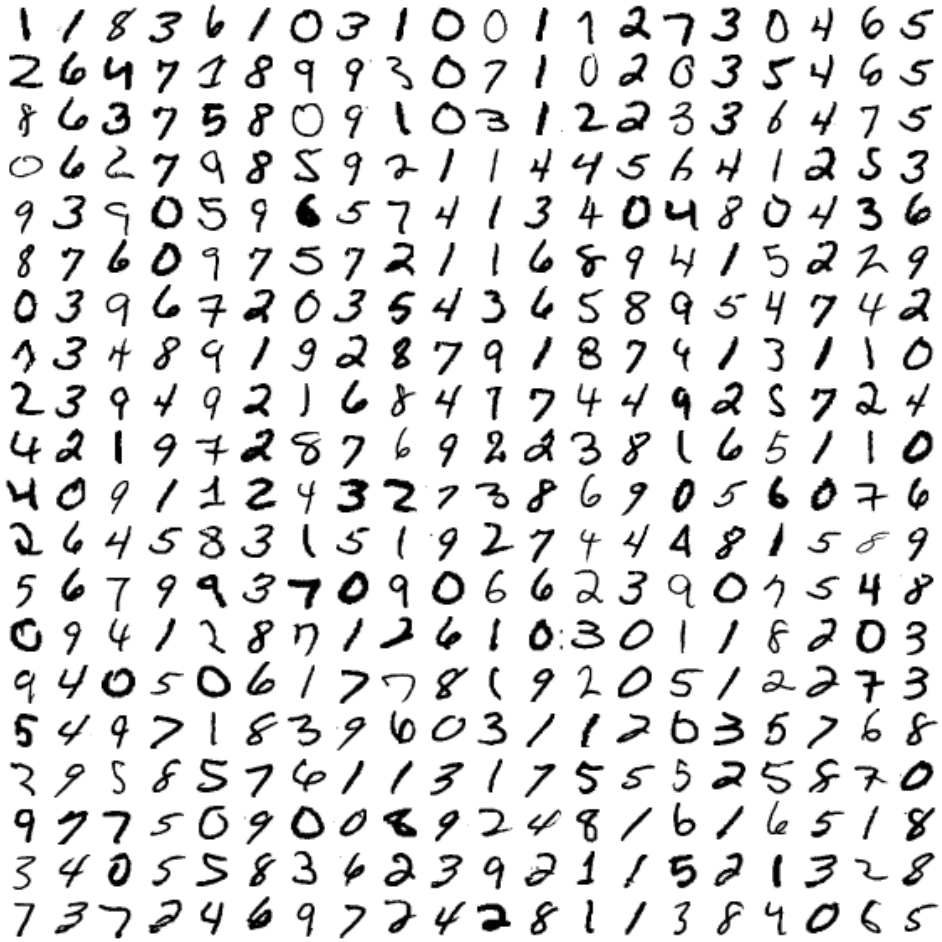

bitcoin value (usd)
10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
2010-01
2012-01
2014-01
2016-01
2018-01
2020-01
2022-01

PyTorch et IA

Python est le langage le plus utilisé en Intelligence Artificielle.

	Language(s)	**License**	**Main backer**
PyTorch	**Python**, C++	BSD	Facebook
TensorFlow	Python, C++	Apache	Google
JAX	Python	Apache	Google
MXNet	Python, C++, R, Scala	Apache	Amazon
CNTK	Python, C++	MIT	Microsoft
Torch 7	Lua	BSD	Facebook
Theano	Python	BSD	U. of Montreal
Caffe	C++	BSD 2 clauses	U. of CA, Berkeley

MNIST data-set



Images 28×28, 60K exemples d'apprentissage, 10K exemples de test.

```
model = nn.Sequential(
    nn.Conv2d( 1, 32, 5), nn.MaxPool2d(3), nn.ReLU(),
    nn.Conv2d(32, 64, 5), nn.MaxPool2d(2), nn.ReLU(),
    nn.Flatten(),
    nn.Linear(256, 200), nn.ReLU(),
    nn.Linear(200, 10)
)

nb_epochs, batch_size = 10, 100
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.1)

model.to(device)
criterion.to(device)
train_input, train_targets = train_input.to(device), train_targets.to(device)

for e in range(nb_epochs):
    for input, targets in zip(train_input.split(batch_size),
                              train_targets.split(batch_size)):
        output = model(input)
        loss = criterion(output, targets)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Pytorch offre un grand nombre de réseaux de neurones déjà entraînés.

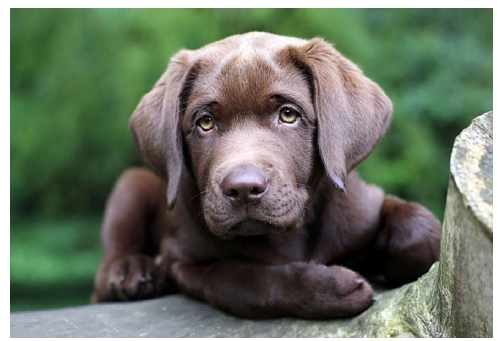
Nous pouvons tester un de ces réseaux de classification d'image sur cet exemple:

```
import PIL, torch, torchvision

# Load and normalize the image
to_tensor = torchvision.transforms.ToTensor()
img = to_tensor(PIL.Image.open('blacklab.jpg'))
img = img.unsqueeze(0)
img = 0.5 + 0.5 * (img - img.mean()) / img.std()

# Load and evaluate the network
alexnet = torchvision.models.alexnet(weights = 'IMAGENET1K_V1')
alexnet.eval()

output = alexnet(img)

# Prints the classes
scores, indexes = output.view(-1).sort(descending = True)

class_names = eval(open('imagenet1000_clsid_to_human.txt', 'r').read())

for k in range(12):
    print(f'#{k+1} {scores[k].item():.02f} {class_names[indexes[k].item()]}')
```

```
12.26 Weimaraner
10.95 Chesapeake Bay retriever
10.87 Labrador retriever
10.10 Staffordshire bullterrier, Staffordshire bull terrier
9.55 flat-coated retriever
9.40 Italian greyhound
9.31 American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier
9.12 Great Dane
8.94 German short-haired pointer
8.53 Doberman, Doberman pinscher
8.35 Rottweiler
8.25 kelpie
8.24 barrow, garden cart, lawn cart, wheelbarrow
8.12 bucket, pail
8.07 soccer ball
```

```
12.26 Weimaraner
10.95 Chesapeake Bay retriever
10.87 Labrador retriever
10.10 Staffordshire bullterrier, Staffordshire bull terrier
9.55 flat-coated retriever
9.40 Italian greyhound
9.31 American Staffordshire terrier, Staffordshire terrier, American pit bull terrier, pit bull terrier
9.12 Great Dane
8.94 German short-haired pointer
8.53 Doberman, Doberman pinscher
8.35 Rottweiler
8.25 kelpie
8.24 barrow, garden cart, lawn cart, wheelbarrow
8.12 bucket, pail
8.07 soccer ball
```

Weimaraner



Chesapeake Bay retriever

Fin