Mise en situation 2020 (Python3)

Citer les principaux types en Python

https://docs.python.org/fr/3/library/datatypes.html (https://docs.python.org/fr/3/library/datatypes.html)

- numbers.Number : créés par les littéraux numériques et renvoyés en tant que résultats par les opérateurs et les fonctions arithmétiques natives. Python distingue les entiers (**int,bool**), les nombres à virgule flottante (**float**) et les nombres complexes (**complex**).

- Python fournit les types natifs suivant : **dict, list, set, frozenset, et tuple**.
- La classe **str** est utilisée pour stocker des chaînes de caractères Unicode.
- Les classes **bytes et bytearray** des données binaires.
- Ensembles (**set, frozenset**) : Ils représentent les ensembles d'objets, non ordonnés, finis et dont les éléments sont uniques.
  - Ensembles (set en anglais) : représentent les ensembles muables. Un ensemble est créé par la fonction native constructeur set() et peut être modifié par la suite à l'aide de différentes méthodes, par exemple add().
  - Ensembles gelés (frozenset en anglais) Ils représentent les ensembles immuables. Ils sont créés par la fonction native constructeur frozenset(). Comme un ensemble gelé est immuable et hachable, il peut être utilisé comme élément d'un autre ensemble ou comme clé de dictionnaire.
- **None** est un type qui ne possède qu'une seule valeur. Il n'existe qu'un seul objet avec cette valeur. Vous accédez à cet objet avec le nom natif None. Il est utilisé pour signifier l'absence de valeur dans de nombreux cas, par exemple pour des fonctions qui ne retournent rien explicitement. Sa valeur booléenne est fausse.

Quels types de données sont immuables (« immutables » en anglais) en Python 3 ?

https://docs.python.org/3/reference/datamodel.html?highlight=immutable (https://docs.python.org/3/reference/datamodel.html?highlight=immutable)

- Tous les types numbers.* tels que Number, Integral (bool,int), Real etc.. sont des objets immuables. Une fois créés, leur valeur ne change pas.
- Ensembles gelés (frozenset en anglais) Ils représentent les ensembles immuables.
- Séquences immuables : Un objet de type de séquence immuable ne peut pas être modifié une fois qu'il a été créé. Si l'objet contient des références à d'autres objets, ces autres objets peuvent être muables et peuvent être modifiés ; cependant, les objets directement référencés par un objet immuable ne peuvent pas être modifiés. Les types suivants sont des séquences immuables :
  - Chaînes de caractères (string en anglais) : est une séquence immuable de valeurs qui représentent des caractères Unicode.
  - n-uplets (ou tuples) Les éléments d'un tuple sont n'importe quels objets Python. Les tuples de deux ou plus éléments sont formés par une liste d'expressions dont les éléments sont séparés par des virgules.
  - Bytes : Les objets bytes sont des tableaux immuables. Les éléments sont des octets (donc composés de 8 bits), représentés par des entiers dans l'intervalle 0 à 255 inclus. Les littéraux bytes (tels que b'abc') et la fonction native constructeur bytes() peuvent être utilisés pour créer des objets bytes. Aussi, un objet bytes peut être décodé vers une chaîne via la méthode decode().

In [1]:

```
A = (1,2,3,4,5) # quel est le type de A, comment modifier le deuxième élément par 10
```

La variable A est de type "tuple", type qui permet de stocker des éléments pouvant être différents dans une structure "immutable"

- Un des avantages des tuples par rapport aux listes est la vitesse
- Tuples ont par définition une taille fixe (alors que les Listes sont dynamiques)
- Tuple est "immutable" alors qu'une structure de type list est "mutable".

On ne peut ni ajouter ni enlever des élements d'un "tuple".

In [2]:

```
A = (1,2,3,4,5)
print(type(x))
print(2 in x)
x[1]=10       # va générer une erreur : TypeError: 'tuple' object does not suppo
rt item assignment
print(x)
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-2-73a4d7dbe54d> in <module>
      1 A = (1,2,3,4,5)
----> 2 print(type(x))
      3 print(2 in x)
      4 x[1]=10       # va générer une erreur : TypeError: 'tuple' o
bject does not support item assignment
      5 print(x)

NameError: name 'x' is not defined
```

Pour pouvoir modifier un des éléments il faut utiliser le type List comme ci-après:

In [3]:

```
x = [1,2,3,4,5]  # ici on crée une liste
print(type(x))
x[1]=10       # avec une liste c'est possible
print(x)
print(10 in x)
```

```
<class 'list'>
[1, 10, 3, 4, 5]
True
```

A partir d'une Liste on peut extraire les éléments distincts en utilisants un set :

In [4]:

```
a = [1,2,2,2,4,6,1,8,9]    # ici on crée une liste
ensemble = set(a) # que l'on convertit en set
print(ensemble)
```

```
{1, 2, 4, 6, 8, 9}
```

Comment obtenir une liste de mots uniques (séparés par des espaces) avec le nombre d'occurrences de chacun de ces mots ?

In [5]:

```
texte = "Les sanglots longs des violons de l'automne, blessent mon coeur d'une l
angeur monotone. Les longs mois de l'automne me blessent !"
liste = texte.split()         # on convertit la chaine de caracatères en liste
 (espaces comme séparateurs par défaut)
mots_uniques = set(liste)     # on extrait les mots uniques en utilisant un set
print(mots_uniques)
[(mot,liste.count(mot)) for mot in mots_uniques]  # on peut maintenant utiliser
 liste.count(mot) pour gener
```

```
{'langeur', "l'automne,", 'des', 'longs', 'sanglots', 'mon', 'me',
'blessent', 'de', '!', "d'une", 'violons', 'Les', 'monotone.', 'moi
s', "l'automne", 'coeur'}
```

Out[5]:

```
[('langeur', 1),
 ("l'automne,", 1),
 ('des', 1),
 ('longs', 2),
 ('sanglots', 1),
 ('mon', 1),
 ('me', 1),
 ('blessent', 2),
 ('de', 2),
 ('!', 1),
 ("d'une", 1),
 ('violons', 1),
 ('Les', 2),
 ('monotone.', 1),
 ('mois', 1),
 ("l'automne", 1),
 ('coeur', 1)]
```

In [6]:

```
s = "Momo le haricot !"
# impossible de faire s[0]='T' car le type string est immutable
print(type(s))
l = list(s)
l[0]='T'
l[2]='t'
print('_'.join(l))
```

```
<class 'str'>
T_o_t_o_ _l_e_ _h_a_r_i_c_o_t_ _!
```

Python Orienté Objet, implémenter une classe Point ?

In [7]:

```
class Point(object):
    """
    A simple 2D cartesian Point with x and y coordinates
    """

    def __init__(self,x=0.0,y=0.0):
        self.x = float(x)
        self.y = float(y)

    def __repr__(self):
        return ("Point(x={x}, y={y})".format(x=self.x,y=self.y))

    def move(self,newx,newy):
        """
        This method moves the point to newx,newy position
        """
        self.x = float(newx)
        self.y = float(newy)

    def moverel(self,dx,dy):
        """
        This method moves the point ny dx,dy relative to its current position
        """
        self.x += float(dx)
        self.y += float(dy)
```

In [8]:

```
P1 = Point(3,5)
print(P1)
```

```
Point(x=3.0, y=5.0)
```

Python Orienté Objet, implémenter une classe NamedPoint ?

In [9]:

```
class NamedPoint(Point):

    def __init__(self,name,x,y):
        Point.__init__(self,x,y)
        self.name = name

    def __repr__(self):
        return ("NamedPoint(name={name},{point}) ".format(name=self.name, point=
Point.__repr__(self)))
```

In [10]:

```
P3 = NamedPoint('P3',3,3)
print(P3)
```

```
NamedPoint(name=P3,Point(x=3.0, y=3.0))
```

Python Orienté Objet : référence, valeur et copie

In [11]:

```
P3 = NamedPoint('P3',3,3)
print("Au début voici le contenu de P3 :\n {p}".format(p=P3))
P4 = P3     # P3 et P4 pointe tous les deux sur le même objet P4 est un alias de P3
P4.name = 'P4'
P4.move(4,4)
P3.move(1,1)
print("A la fin toutes les modifications sur P3 ou P4 s'appliquent au même objet P4 est un alias de P3")
print(P3, id(P3))
print(P4, id(P4))
```

```
Au début voici le contenu de P3 :
 NamedPoint(name=P3,Point(x=3.0, y=3.0))
A la fin toutes les modifications sur P3 ou P4 s'appliquent au même
objet P4 est un alias de P3
NamedPoint(name=P4,Point(x=1.0, y=1.0))  140189680788312
NamedPoint(name=P4,Point(x=1.0, y=1.0))  140189680788312
```

In [12]:

```
import copy                 # le module copy permet de palier a ce probleme si une methode clone n'existe pas
P3 = NamedPoint('P3',3,3)
P4 = copy.copy(P3)          # ici on cree une copie de P3 dans la varaible P4 avec un nouvel emplacement mémoire
P4.name = 'P4'
P4.move(4,4)
P3.move(1,1)
print(P3, id(P3))
print(P4, id(P4))
```

```
NamedPoint(name=P3,Point(x=1.0, y=1.0))  140189680787864
NamedPoint(name=P4,Point(x=4.0, y=4.0))  140189680789376
```

Qu’est-ce qui est affiché par la dernière ligne ?

In [13]:

```
id=54321   # attention danger ! ici on vient de redefinir la fonction builtin id PAS BIEN DU TOUT !
class Account:
    def __init__(self, id):
        self.id = id + 1 # on stocke la valeur du parametre id dans l'attribut de la classe id en y ajoutant 1
        id = 0        # ici id correspond à une variable locale n'existant que dans la fonction (portée locale)
print(id)
acc = Account(id=12345) # id ici correspond au nom du paramètre
print(acc.id)
```

```
54321
12346
```

Qu'est-ce qui est affiché par la dernière ligne ?

In [14]:

```
confusion = {}                  # on initialise un dictionary vide
confusion[1] = 1                # on ajoute une entrée au dict avec la clé d'accès 1
confusion['1'] = 2              # on ajoute une nouvelle entrée au dict avec la clé
 d'accès '1'
confusion[1.0] = 4              # on modifie l'entrée du dict avec la clé d'accès 1
1.0 ou 1 c'est pareil

somme = 0
for k in confusion:
    somme += confusion[k]

print(somme)
print(confusion)
```

```
6
{1: 4, '1': 2}
```

Qu'est-ce qui est affiché par la dernière ligne ?

In [15]:

```
x = True
y = False
z = False

if x or y and z:
    print("yes")
else:
    print("no")
```

```
yes
```

Classer la liste par ordre décroissant de quantité :

In [16]:

```
inventaire = [("pommes", 22),("melons", 4),("poires", 18),("fraises", 76),("prun
es", 51)]
#Classer cette liste selon la quantité de chaque fruit par ordre décroissant
```

In [17]:

```
inventaire_inverse = [(qtt, nom_fruit) for nom_fruit, qtt in inventaire]
print(inventaire_inverse)
inventaire_inverse.sort(reverse=True)
inventaire = [(nom_fruit, qtt) for qtt , nom_fruit in inventaire_inverse]
print(inventaire)
```

```
[(22, 'pommes'), (4, 'melons'), (18, 'poires'), (76, 'fraises'), (5
1, 'prunes')]
[('fraises', 76), ('prunes', 51), ('pommes', 22), ('poires', 18),
('melons', 4)]
```

calculer la somme d'une liste de nombres

In [18]:

```
import numpy as np
import numpy.random as nprnd
liste = nprnd.randint(1000, size=1000)
print(type(list(liste)))
print(sum(list(liste)))
%timeit sum(list(liste))
```

```
<class 'list'>
503194
69.4 µs ± 1.39 µs per loop (mean ± std. dev. of 7 runs, 10000 loops
each)
```

In [19]:

```
import functools
print(functools.reduce(lambda x,y: x+y, liste))
%timeit functools.reduce(lambda x,y: x+y, liste)
```

```
503194
130 µs ± 910 ns per loop (mean ± std. dev. of 7 runs, 10000 loops ea
ch)
```

In [20]:

```
print(np.sum(liste))
%timeit np.sum(liste)
```

```
503194
2.44 µs ± 6.66 ns per loop (mean ± std. dev. of 7 runs, 100000 loops
each)
```

Calculer la somme d'une liste des nombres impairs :

In [21]:

```
liste = [2,1,1,2,4,1,3]
```

In [22]:

```
liste_filtree = [i for i in liste if i % 2 ==1]
print(liste_filtree)
sum(liste_filtree)
```

```
[1, 1, 1, 3]
```

Out[22]:

6

Transformez cette liste pour qu'il ne reste que des éléments unique :

In [23]:

```
liste = [2,1,1,2,4,1,3]
list(set(liste))
```

Out[23]:

```
[1, 2, 3, 4]
```

Meilleur pratique pour ouvrir un fichier:

In [24]:

```
%%writefile test.txt
Ceci est la ligne numéro 1
et la ligne numéro deux
d'un fichier texte qui au final contiendra
quatre lignes !
```

```
Overwriting test.txt
```

In [25]:

```
# 1) Comment ouvrir un fichier en lecture en traitant les exceptions possibles ?
my_file = 'test2.txt'
try:
    f = open(my_file)

    try:
        all = f.readlines()
        print("FILE : {file} contains {numlines:06d} lines ".format(file=my_file
,numlines=len(all)))
    except EXPECTED_EXCEPTION_TYPES as e:
        print("ERROR READING FILE : {file}".format(file=my_file))
        print(e)
    finally:
        f.close()

except (IOError, OSError) as e:
    print("ERROR OPENING FILE : {file}".format(file=my_file))
    print(e)
# https://pyformat.info/
```

```
ERROR OPENING FILE : test2.txt
[Errno 2] No such file or directory: 'test2.txt'
```

In [26]:

```
my_file = 'test.txt'
with open(my_file) as f:
    all = f.readlines()
    print(all)
    print("FILE : {file} contains {numlines:06d} lines ".format(file=my_file,num
lines=len(all)))
```

```
['Ceci est la ligne numéro 1\n', 'et la ligne numéro deux\n', "d'un
fichier texte qui au final contiendra\n", 'quatre lignes !\n']
FILE : test.txt contains 000004 lines
```

Connaissances web framework python :

Python Web frameworks (http://www.fullstackpython.com/web-frameworks.html)

Citer 5 web frameworks python, lesquels connaissez-vous ?

1. Django (https://www.djangoproject.com/)
2. Flask (http://flask.pocoo.org/)
3. Pyramid (http://www.pylonsproject.org/)
4. Web2Py (http://www.web2py.com/)
5. Turbogears2 (http://www.turbogears.org/)
6. Falcon (http://falconframework.org/)
7. Bottle (http://bottlepy.org/docs/dev/index.html)
8. CherryPy (http://www.cherrypy.org/)
9. Muffin (https://github.com/klen/muffin)
10. Tornado (http://www.tornadoweb.org/en/stable/)

Quels sont les fonctions essentielles d'un web framework ?

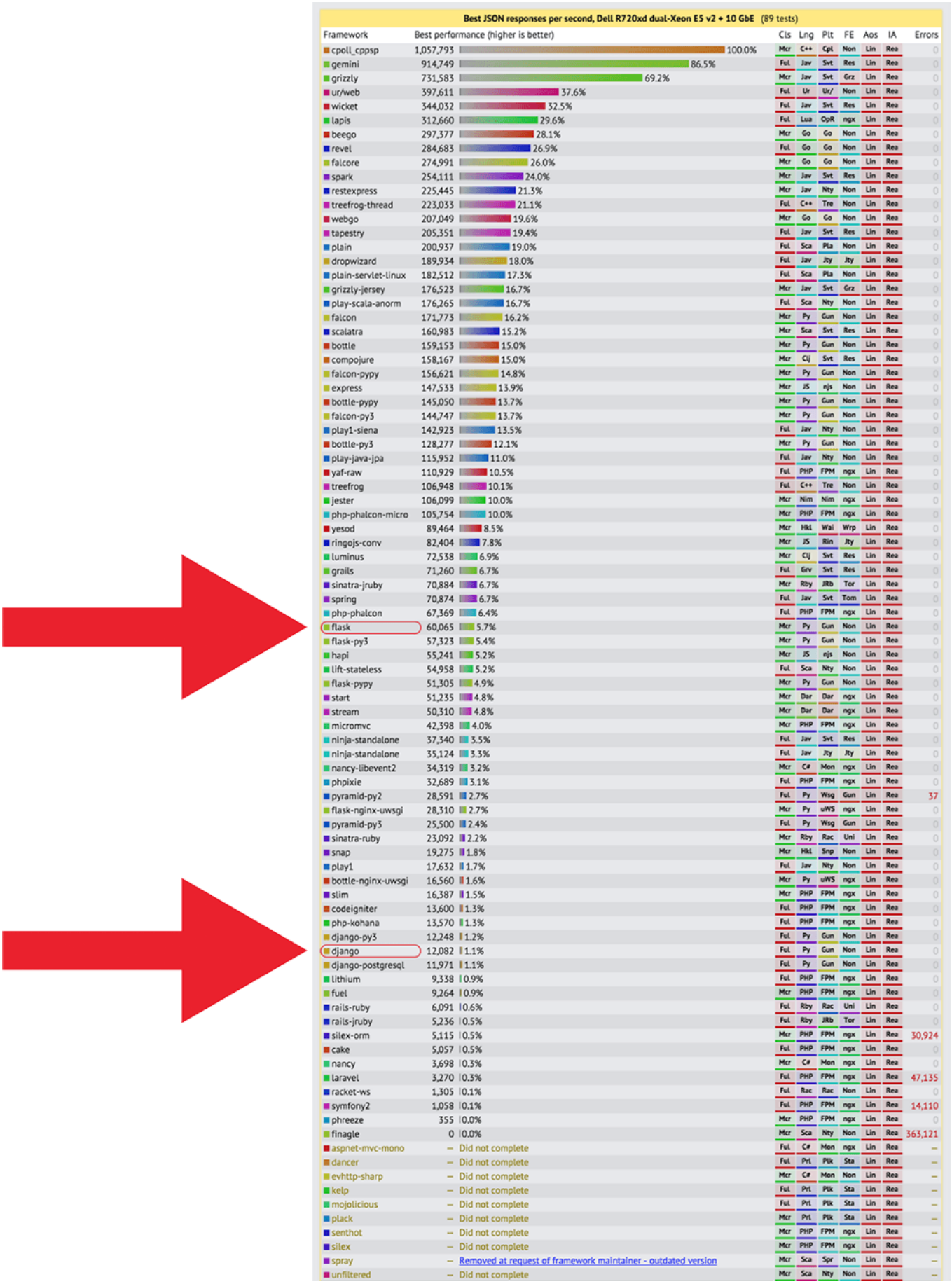
certaines fonctions de base incluent :

- URL routing
- HTML, XML, JSON, and other output format templating
- Database manipulation
- Security against Cross-site request forgery (CSRF) and other attacks
- Session storage and retrieval

Quels sont les avantages / inconvénients de ces frameworks ?

Django vs Flask vs Pyramid: Choosing a Python Web Framework (https://www.airpair.com/python/posts/django-flask-pyramid) Pirates use Flask, the Navy uses Django (https://wakatime.com/blog/25-pirates-use-flask-the-navy-uses-django)

Performance of some python web frameworks (http://klen.github.io/py-frameworks-bench/)



Best JSON responses per second, Dell R720xd dual-Xeon E5 v2 + 10 GbE (89 tests)

Framework	Best performance (higher is better)		Cls	Lng	Plt	FE	Aos	IA	Errors
cpoll_cppsp	1,057,793	100.0%	Mcr	C++	Cpl	Non	Lin	Rea	0
gemini	914,749	86.5%	Ful	Jav	Svt	Res	Lin	Rea	0
grizzly	731,583	69.2%	Mcr	Jav	Svt	Grz	Lin	Rea	0
ur/web	397,611	37.6%	Ful	Ur	Ur/	Non	Lin	Rea	0
wicket	344,032	32.5%	Ful	Jav	Svt	Res	Lin	Rea	0
lapis	312,660	29.6%	Ful	Lua	OpR	ngx	Lin	Rea	0
beego	297,377	28.1%	Mcr	Go	Go	Non	Lin	Rea	0
revel	284,683	26.9%	Ful	Go	Go	Non	Lin	Rea	0
falcore	274,991	26.0%	Mcr	Go	Go	Non	Lin	Rea	0
spark	254,111	24.0%	Mcr	Jav	Svt	Res	Lin	Rea	0
restexpress	225,445	21.3%	Mcr	Jav	Nty	Non	Lin	Rea	0
treefrog-thread	223,033	21.1%	Ful	C++	Tre	Non	Lin	Rea	0
webgo	207,049	19.6%	Mcr	Go	Go	Non	Lin	Rea	0
tapestry	205,351	19.4%	Ful	Jav	Svt	Res	Lin	Rea	0
plain	200,937	19.0%	Ful	Sca	Pla	Non	Lin	Rea	0
dropwizard	189,934	18.0%	Ful	Jav	Jty	Jty	Lin	Rea	0
plain-servlet-linux	182,512	17.3%	Ful	Sca	Pla	Non	Lin	Rea	0
grizzly-jersey	176,523	16.7%	Mcr	Jav	Svt	Grz	Lin	Rea	0
play-scala-anorm	176,265	16.7%	Ful	Sca	Nty	Non	Lin	Rea	0
falcon	171,773	16.2%	Mcr	Py	Gun	Non	Lin	Rea	0
scalatra	160,983	15.2%	Mcr	Sca	Svt	Res	Lin	Rea	0
bottle	159,153	15.0%	Mcr	Py	Gun	Non	Lin	Rea	0
compojure	158,167	15.0%	Mcr	Clj	Svt	Res	Lin	Rea	0
falcon-pypy	156,621	14.8%	Mcr	Py	Gun	Non	Lin	Rea	0
express	147,533	13.9%	Mcr	JS	njs	Non	Lin	Rea	0
bottle-pypy	145,050	13.7%	Mcr	Py	Gun	Non	Lin	Rea	0
falcon-py3	144,747	13.7%	Mcr	Py	Gun	Non	Lin	Rea	0
play1-siena	142,923	13.5%	Ful	Jav	Nty	Non	Lin	Rea	0
bottle-py3	128,277	12.1%	Mcr	Py	Gun	Non	Lin	Rea	0
play-java-jpa	115,952	11.0%	Ful	Jav	Nty	Non	Lin	Rea	0
yaf-raw	110,929	10.5%	Ful	PHP	FPM	ngx	Lin	Rea	0
treefrog	106,948	10.1%	Ful	C++	Tre	Non	Lin	Rea	0
jester	106,099	10.0%	Mcr	Nim	Nim	ngx	Lin	Rea	0
php-phalcon-micro	105,754	10.0%	Mcr	PHP	FPM	ngx	Lin	Rea	0
yesod	89,464	8.5%	Mcr	Hkl	Wai	Wrp	Lin	Rea	0
ringojs-conv	82,404	7.8%	Mcr	JS	Rin	Jty	Lin	Rea	0
luminus	72,538	6.9%	Mcr	Clj	Svt	Res	Lin	Rea	0
grails	71,260	6.7%	Ful	Grv	Svt	Res	Lin	Rea	0
sinatra-jruby	70,884	6.7%	Mcr	Rby	JRb	Tor	Lin	Rea	0
spring	70,874	6.7%	Ful	Jav	Svt	Tom	Lin	Rea	0
php-phalcon	67,369	6.4%	Ful	PHP	FPM	ngx	Lin	Rea	0
flask	60,065	5.7%	Mcr	Py	Gun	Non	Lin	Rea	0
flask-py3	57,323	5.4%	Mcr	Py	Gun	Non	Lin	Rea	0
hapi	55,241	5.2%	Mcr	JS	njs	Non	Lin	Rea	0
lift-stateless	54,958	5.2%	Ful	Sca	Nty	Non	Lin	Rea	0
flask-pypy	51,305	4.9%	Mcr	Py	Gun	Non	Lin	Rea	0
start	51,235	4.8%	Mcr	Dar	Dar	ngx	Lin	Rea	0
stream	50,310	4.8%	Mcr	Dar	Dar	ngx	Lin	Rea	0
micromvc	42,398	4.0%	Mcr	PHP	FPM	ngx	Lin	Rea	0
ninja-standalone	37,340	3.5%	Ful	Jav	Svt	Res	Lin	Rea	0
ninja-standalone	35,124	3.3%	Ful	Jav	Jty	Jty	Lin	Rea	0
nancy-libevent2	34,319	3.2%	Mcr	C#	Mon	ngx	Lin	Rea	0
phpixie	32,689	3.1%	Ful	PHP	FPM	ngx	Lin	Rea	0
pyramid-py2	28,591	2.7%	Ful	Py	Wsg	Gun	Lin	Rea	37
flask-nginx-uwsgi	28,310	2.7%	Mcr	Py	uWS	ngx	Lin	Rea	0
pyramid-py3	25,500	2.4%	Ful	Py	Wsg	Gun	Lin	Rea	0
sinatra-ruby	23,092	2.2%	Mcr	Rby	Rac	Uni	Lin	Rea	0
snap	19,275	1.8%	Mcr	Hkl	Snp	Non	Lin	Rea	0
play1	17,632	1.7%	Ful	Jav	Nty	Non	Lin	Rea	0
bottle-nginx-uwsgi	16,560	1.6%	Mcr	Py	uWS	ngx	Lin	Rea	0
slim	16,387	1.5%	Mcr	PHP	FPM	ngx	Lin	Rea	0
codeigniter	13,600	1.3%	Ful	PHP	FPM	ngx	Lin	Rea	0
php-kohana	13,570	1.3%	Ful	PHP	FPM	ngx	Lin	Rea	0
django-py3	12,248	1.2%	Ful	Py	Gun	Non	Lin	Rea	0
django	12,082	1.1%	Ful	Py	Gun	Non	Lin	Rea	0
django-postgresql	11,971	1.1%	Ful	Py	Gun	Non	Lin	Rea	0
lithium	9,338	0.9%	Ful	PHP	FPM	ngx	Lin	Rea	0
fuel	9,264	0.9%	Mcr	PHP	FPM	ngx	Lin	Rea	0
rails-ruby	6,091	0.6%	Ful	Rby	Rac	Uni	Lin	Rea	0
rails-jruby	5,236	0.5%	Ful	Rby	JRb	Tor	Lin	Rea	0
silex-orm	5,115	0.5%	Mcr	PHP	FPM	ngx	Lin	Rea	30,924
cake	5,057	0.5%	Ful	PHP	FPM	ngx	Lin	Rea	0
nancy	3,698	0.3%	Mcr	C#	Mon	ngx	Lin	Rea	0
laravel	3,270	0.3%	Ful	PHP	FPM	ngx	Lin	Rea	47,135
racket-ws	1,305	0.1%	Ful	Rac	Rac	Non	Lin	Rea	0
symfony2	1,058	0.1%	Ful	PHP	FPM	ngx	Lin	Rea	14,110
phreeze	355	0.0%	Mcr	PHP	FPM	ngx	Lin	Rea	0
finagle	0	0.0%	Mcr	Sca	Nty	Non	Lin	Rea	363,121
aspnet-mvc-mono	–	Did not complete	Ful	C#	Mon	ngx	Lin	Rea	–
dancer	–	Did not complete	Ful	Prl	Plk	Sta	Lin	Rea	–
evhttp-sharp	–	Did not complete	Mcr	C#	Mon	Non	Lin	Rea	–
kelp	–	Did not complete	Ful	Prl	Plk	Sta	Lin	Rea	–
mojolicious	–	Did not complete	Ful	Prl	Plk	Sta	Lin	Rea	–
plack	–	Did not complete	Mcr	Prl	Plk	Sta	Lin	Rea	–
senthot	–	Did not complete	Mcr	PHP	FPM	ngx	Lin	Rea	–
silex	–	Did not complete	Mcr	PHP	FPM	ngx	Lin	Rea	–
spray	–	Removed at request of framework maintainer - outdated version	Mcr	Sca	Spr	Non	Lin	Rea	–
unfiltered	–	Did not complete	Mcr	Sca	Nty	Non	Lin	Rea	–

In [ ]:

```
```