Python 3.x

Structures

Une classe

```
class Dog(Animal):
  def __init__(self):
    # ...

  def hello(self):
    # ...
```

Une fonction

```
def ma_fonction(arg1, arg2):
  return 3
```

Les conditions

```
if a == b:
  # faire quelque chose
elif b == c:
  # autre chose
else:
  # encore autre chose
```

Les conditions ternaires

```
a = 5 if c == b else 8
# Si c == b, a = 5 sinon 8
```

Gestion des exceptions

```
try:
  # quelque chose
except:
  # autre chose
```

Conversions

Conversion entiers / chaines de caractères

```
int("1853") * 2 # 3706
str(1853) * 2   # "18531853"
```

Initialisations

Initialiser une liste 6 éléments à 3

```
l = [3,]*6
# [3, 3, 3, 3, 3, 3]
```

Initialiser 4 variables à None

```
a,b,c,d = (None,)*4
# a=None, b=None, c=None, d=None
```

Générer une liste de carrés

```
[i**2 for i in range(1,6)]
# Genere [1, 4, 9, 16, 25]
```

Iterables

Tout les exemples sont présentés avec une chaine de caractère mais fonctionnent également avec une liste ou d'autres iterables.

Itérer sur les charactères d'une liste

```
for carac in "hello world":
  print(carac, end="-")
# h-e-l-l-o- -w-o-r-l-d-
```

Accéder à des caractères d'une liste

```
"hello world"[2]  # 3eme element "l"
"hello world"[-1] # dernier element "d"
```

Accéder à des sous chaines de caractère

```
chaine = "hello world"
chaine[1:5]   # "ello"
chaine[-5:-1] # "worl"
chaine[-5:]   # "world"
chaine[4:]    # "o world"
```

Inverse la chaîne de caractère

```
chaine[::-1] # "dlrow olleh"
```

Listes

Ajout et concaténation

```
[1, 2, 3].append(4) # [1, 2, 3, 4]
[1, 2] += [3, 4]    # [1, 2, 3, 4]
```

Associer plusieurs listes

```
zip([1, 2, 3], [4, 5, 6])
# [(1, 4), (2, 5), (3, 6)]
```

Chaines de caractères

Conversion code ASCII / caractère

```
chr(97)  # 'a'
ord('a') # 97
```

Dictionnaire

Vérifier l'existence d'une clé

```
dic = {"a": 1}
if "a" in dic:
  # Verifie si la clef a existe
  pass
```

Itérer sur un dictionnaire

```
dic = {"a": 1, "b": 2}
for cle, valeur in dic.items():
  print(cle, valeur)
```

Tuple

Tuple packing et unpacking

```
t = 12345, 54321, 'hello!'
x, y, z = t
```

Cas particuliers (Tuple de 0 et 1 élément)

```
empty = ()
singleton = 'hello',
# notez la derniere virgule
```

Set

Un set ne contient qu'une seule fois chaque valeur et n'est pas ordonné.

```
{8, 9, 9, 1}
# {9, 8, 1}
```

Entrées / Sorties

Pour lire sur l'entrée standard :

```
input() # stdin
```

Pour écrire sur la sortie standard :

```
print(x, y, z) # print sur stdout
print("fatal error", file=sys.stderr)
# print sur stderr
```

Fonctionnel

Réduction (reduce)

```
from functools import reduce
reduce(lambda x, y: x*y, [2, 3, 4])
# 2 * 3 * 4 = 24
```

Filtre (filter)

```
list(filter(lambda x: x > 2, [1,2,3,4]))
# [3, 4]

[n for n in [1, 2, 3, 4] if n > 2]
# [3, 4]
```

Association (map)

```
list(map(lambda x: x**2, [2, 3, 4]))
# [4, 9, 16]

[n**2 for n in [2, 3, 4]]
# [4, 9, 16]
```

Mathématiques

Récupérer le minimum ou le maximum de plusieurs valeurs.

```
min(3, 5)        # 3
min(3, 2, 8, 7) # 2
min([13, 5, 8]) # 5
max(6, 3)        # 3
...
```

A la puissance n

```
i, n = (3, 2)
i ** n     # 9
pow(i, n) # 9
```

Valeur absolue

```
abs(-5) # 5
```

Tri

Retourner une nouvel iterable trié (Fonctionne avec tout iterable)

```
sorted([9,12,2])
# [2, 9, 12]

sorted({"F": 0, "D": 0, "A": 0, "B": 0})
# ['A', 'B', 'D', 'F']

sorted([9,12,2], reverse=True)
# [12, 9, 2]
```

Trier une liste (seulement)

```
a = [5, 2, 8]
a.sort()
# a = [2, 5, 8]
```

Threads et Queue

```
from Queue import Queue
from threading import Thread

def listener(q):
  while True:
    print(q.get())

q = Queue()
t = Thread(target=listener, args=(q))

t.start()
q.put("hello")
```

Réseau

```
import socket, select

sock = socket.socket( \
  socket.AF_INET, \
  socket.SOCK_STREAM)
rlist = []

sock.bind(('0.0.0.0', 1025))
sock.listen()

while True:
  rd, wr, err = select.select(rlist, [], [])
  for s in rd:
    if s is sock:
      client_socket, address = sock.accept()
      rlist.append(client_socket)
    else:
      data = s.recv(1024)
      if data: print(data); sock.send("OK")
      else: s.close(); rlist.remove(s)
```

HTTP

`http://flask.pocoo.org/docs/0.11/quickstart/`
Créer un dossier /static pour servir des fichiers.
Créer un dossier /templates pour mettre les templates au format JINJA2.

```
<h1>{{ name }} </h1>
```

Code d'exemple avec Flask

```
from flask import Flask, request, \
        render_template, url_for, session
app = Flask(__name__)

@app.route("/user")
@app.route("/user/<username>", \
        methods=['GET', 'POST'])
def hello(username=None):
    if request.method == 'POST':
        # request.form['hello']
        # session['username'] = xx
        return render_template( \
        'hello.html', name=username)
    else:
        return "Hello "+username+" !"

app.run()
```

Stocker données

TODO

Hash et encodage

base64
md5
hash

Tableaux ASCII

Lettres minuscules

dec	char	dec	char	dec	char
97	a	106	j	115	s
98	b	107	k	116	t
99	c	108	l	117	u
100	d	109	m	118	v
101	e	110	n	119	w
102	f	111	o	120	x
103	g	112	p	121	y
104	h	113	q	122	z
105	i	114	r		

Lettres majuscules

dec	char	dec	char	dec	char
65	A	74	J	83	S
66	B	75	K	84	T
67	C	76	L	85	U
68	D	77	M	86	V
69	E	78	N	87	W
70	F	79	O	88	X
71	G	80	P	89	Y
72	H	81	Q	90	Z
73	I	82	R		

Algorithmes

ROT N
Génération de nombres premiers
??