Programmation Python Partie 1 : Les branchements et les boucles

Alexandre Gramfort

Les branchements et les boucles

Branchements

Boucles

Branchements: if, elif, else

In [95]:
```
condition1 = False
condition2 = False

if condition1:
    print("condition1 est vraie.")
elif condition2:
    print("condition2 est vraie.")
else:
    print("condition1 et condition2 sont fausses.")
```

```
condition1 et condition2 sont fausses.
```

En Python **l'indentation est obligatoire** car elle influence l'exécution du code

Les branchements et les boucles

Branchements
Boucles

Exemples:

In [96]:
```
condition1 = condition2 = True

if condition1:
    if condition2:
        print("condition1 et condition2 sont vraies.")
```

```
condition1 et condition2 sont vraies.
```

Les branchements et les boucles

Branchements
Boucles

Exemple de mauvaise indentation:

In [97]:
```
if conditiont1:
    if condition2:
    print("condition1 et condition2 sont vraies.")
```

```
  File "<ipython-input-97-a5ea683b4094>", line 3
    print("condition1 et condition2 sont vraies.")
        ^
IndentationError: expected an indented block
```

Les branchements et les boucles

Branchements

Boucles

In [98]:
```
condition1 = True

if condition1:
    print("affiche si condition1 est vraie.")

    print("toujours dans le bloc du branchement if.")
```

```
affiche si condition1 est vraie.
toujours dans le bloc du branchement if.
```

In [99]:
```
condition1 = False

if condition1:
    print("affiche si condition1 est vraie.")

print("affiche en dehors du bloc du branchement if.")
```

```
affiche en dehors du bloc du branchement if.
```

Boucles

Boucles **`for`**:

In [100]:
```
for x in [1, 2, 3]:
    print(x)
```

```
1
2
3
```

La boucle `for` itère sur les éléments de la list fournie. Par exemple:

In [101]:
```
for x in range(4): # par défault range commence à 0
    print(x)
```

```
0
1
2
3
```

Les branchements et les boucles

Branchements

Boucles

In [102]:
```
for word in ["big", "data", "en", "python"]:
    print(word)
```

```
big
data
en
python
```

In [103]:
```
for c in "calcul":
    print(c)
```

```
c
a
l
c
u
l
```

Mot clé `enumerate`

Parfois il est utile d'accéder à la valeur et à l'index de l'élément. Il faut alors utiliser `enumerate`:

In [104]:
```
s = "calcul"
for idx, c in enumerate(s):
    print(idx, c)
```

```
(0, 'c')
(1, 'a')
(2, 'l')
(3, 'c')
(4, 'u')
(5, 'l')
```

Les branchements et les boucles

Branchements

Boucles

In [105]:
```
s = "calcul"
for idx in range(len(s)):
    print(idx, s[idx])
```

```
(0, 'c')
(1, 'a')
(2, 'l')
(3, 'c')
(4, 'u')
(5, 'l')
```

In [106]:
```
cnt = 0
for c in s:
    print(cnt, s[cnt])
    cnt += 1
```

```
(0, 'c')
(1, 'a')
(2, 'l')
(3, 'c')
(4, 'u')
(5, 'l')
```

Les branchements et les boucles

Branchements

Boucles

Pour itérer sur un dictionnaire:

In [107]:
```
for key, value in params.items():
    print("%s = %s" % (key, value))
```

```
param3 = 3.0
param2 = B
param1 = A
```

In [108]:
```
for key in params:
    print(key)
```

```
param3
param2
param1
```

list comprehension: création de liste avec `for`

In [109]:
```
l = [x ** 2 for x in range(0,5)]

print(l)

# est la version courte de :
l = list()
for x in range(0, 5):
    l.append(x ** 2)

print(l)
```

```
[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16]
```

Les branchements et les boucles

Branchements

Boucles

Boucles `while`:

In [110]:
```
i = 0

while i < 5:
    print(i)
    i += 1

print("OK")
```

```
0
1
2
3
4
OK
```