



Image: pngtree

Option IA & Programmation

Rappels de syntaxes

Alexandre Mazel

Année 2022-2023

alexandre.zelma@gmail.com

Les variables de type tableau

Exemple de variable contenant un tableau:

```
a = [1, 2, 3]
```

```
prenoms = ["Alex", "Elsa", "Gaia"]
```

```
prenoms_et_age = [ ["Alex", 13], ["Gaia", 12] ]
```

le premier index est le **0** !!!

```
print(a[0])          # => 1
```

```
print(prenoms[2])    # => "Gaia"
```

Les tableaux

```
# comment modifier un tableau
```

```
a = [1, 8, 3]
```

```
a[1] = 2          # a vaut [1, 2, 3]
```

```
a[3] = 4          # => out of range (hors du tableau)
```

```
len(a)            # retourne 3
```

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

```
len(a)            # retourne 4
```

Les tableaux à n dimensions

Exemple de variable contenant un tableau hétérogènes:

```
a = [ [1,2,3], "Alex", 18, [100], [4,5,6] ]
```

```
prenoms_et_age = [ ["Alex",13], ["Gaia",12] ]
```

```
bbb = [[[10,20]]]
```

le premier index est toujours le **0** !!!

<code>len(a)</code>	# => 5	<code>a[2]</code>	=> 18
<code>a[0]</code>	# => [1,2,3]	<code>prenoms_et_age[0][1]</code>	=> 13
<code>len(a[0])</code>	# => 3	<code>prenoms_et_age[1][1]</code>	=> 12
<code>a[0][0]</code>	# => 1	<code>bbb[0][0][1]</code>	=> 20
<code>a[1]</code>	# => "Alex"	<code>len(bbb[0][0])</code>	=> 2

Indices de chaînes de caractères

```
a = "Alex"
```

En mémoire a est en fait un tableau homogène de caractères:



le premier index est toujours le 0 !!!

<code>len(a)</code>	# => 4	<code>a[:-1]</code>	=> 'Ale'
<code>a[2]</code>	# => 'e'	<code>a[0] + a[1:]</code>	=> 'Alex'
<code>a[1:3]</code>	# => 'le'	<code>a[0]*3 + a[1:]</code>	=> 'AAAlex'
<code>a[1:]</code>	# => 'lex'	<code>a[:]</code>	=> 'Alex'
<code>a[:2]</code>	# => 'Al'	<code>a[::2]</code>	=> 'Ae'
<code>a[-1]</code>	# => 'x'	<code>a[::-1]</code>	=> 'xelA'
<code>a[-2:]</code>	# => 'ex'	<code>a[0] = "b"</code>	impossible de changer un caractère dans une chaîne.

Les fichiers

usage des fichiers:

ecriture

```
file = open("toto.txt", "w")
file.write( "Salut toto!")
file.close()
```

lecture

```
file = open("toto.txt", "r")
s = file.read()
file.close()
print(s)
# autre méthode pratique:
file.readline() # lit une ligne d'un
fichier
```

autour des fichiers

récupérer la liste des fichiers d'un dossier

```
files = os.listdir("c:\\")
```

savoir si un fichier existe

```
is_exist = os.path.isfile("toto.txt")
```

récupérer la taille d'un fichier

```
size = os.path.getsize("toto.txt")
```

Les dictionnaires ou liste associatives

Un dictionnaire est une liste indexé par une chaîne de caractère au lieu d'un indice.

```
d = {}                                # Déclaration d'un tableau
d["chien"] = "wouf wouf"             # Ajoute une entrée dans le dictionnaire "chien" => "wouf wouf"
d["chat"] = "miaou"                  # Ajoute une entrée dans le dictionnaire "chat" => "miaou"

d["age marc"] = 34                   # Je peux associer ce que je veux !
d["age marc"] += 1                   # Maintenant d["age marc"] == 35

print(d["le"])                       # IndexError: "le" n'est pas dans le dictionnaire.
len(d)                              # Retourne 3, car 3 elements: chien, chat et age marc

for k,v in mydict.items():           # Parcours le dictionnaire. Va afficher:
    print(k,v)                       chien wouf wouf
                                     chat miaou
                                     age marc 35
```

Les fonctions

définition des fonctions

```
def mafonction1(param1,param2 = 1):  
    # le parametre2 a une valeur par défaut qui peut etre ommise lors de l'appel  
  
    if param1 > 4:  
        return 4  
    return param2 # valeur de retour
```

```
def myfunc2(parametre1,parametre2):  
    val1 = parametre1*parametre2  
    val3 = parametre2  
    val3 = parametre2*38  
    return val1,val2,val3 # la fonction retourne 3 valeurs!
```

appel des fonctions

```
y = mafonction1(3) # le deuxieme parametre n'est pas spécifié  
a,b,c = myfunc2(5,6)
```


OpenCV: function sheet

# I/O	# filter
im = cv2.imread(filename)	Sobel
cv2.imwrite(filename, im)	CannyEdge
cv2.imshow("window_name", im)	# numpy method
cv2.waitKey(milliseconds)	im[y,x] => couleur du pixel en (x,y)
# cree une image vide:	
im = np.zeros((h,w,3), dtype=np.uint8)	im[:, :, 1] = 0 # efface tout les pixels vert
 # conversion	 im = im[10:,:] # efface la bande de 10 pixels supérieur
cvtColor	
resize	im[0:4,0:8] = im2 # colle im2 dans le coin de im (im2 doit ici faire 8x4)
rot90	
normalize	