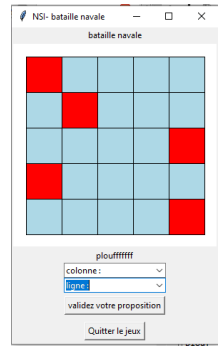


# Bataille navale

Vous pouvez télécharger  
l'exécutable ici :  
<https://bit.ly/39TWH8Q>



## 1/ échauffement

Pour ne pas avoir de soucis de compréhension sur notre programme de bataille navale, nous allons déjà faire quelques exercices d'échauffement.

→ soit le code suivant :

```
listeA=[0,2]
listeB=listeA
listeB.append(4)
```

→ Que vaut listeA et listeB après l'exécution de ces lignes de codes ?

listeA :                                      listeB :

→ Testez dans Spyder. Surpris ?

Surprise? C'est parce que faire `listeB=listeA` ne crée pas une nouvelle liste mais se contente d'ajouter une étiquette à la liste existante ! `listeA` et `listeB` pointent tous les deux sur la même liste ! *C'est donc différent que pour les variables de base (si on fait `x=4` puis `y=x` puis `y=3`, on aura bien `y` qui vaut 3 et `x` qui vaut 4)*

Ouvrir dans spyder le fichier *echauffement.py*

### Etude de la fonction *saisir\_liste()*

→ ôtez le # ligne 14 afin de rendre cette ligne de code exécutable  
→ étudiez le code de cette fonction. Résumez en quelques mots son rôle

→ quel est le type de *liste\_saisie* ? \_\_\_\_\_  
→ quel est le type des deux variables contenues dans *liste\_saisie* ? \_\_\_\_\_  
→ que va donner la ligne de code suivante ? : **type(liste1)** ? Essayez

→ que va donner la ligne de code suivante ? : **type(liste1[0])** ? Essayez

→ essayez cette instruction : **c=ord(liste1[0])** . Que vaut **c** ? chercher le code ASCII du caractère que vous avez tapé en premier et comparez avec la valeur de **c** (vous avez un convertisseur ici si vous voulez : <https://fr.rakko.tools/tools/76/> ) . Conclure quant à l'action de **ord()**

→ relancer le programme et saisissez deux nombres en guise de lettres. que va donner la ligne de code suivante ? : **type(liste1[0])** ? Essayez

→ essayez cette instruction : **d=int(liste1[0])** toujours en entrant pour les deux valeurs a et b deux nombres. Que vaut **d** ? quel est son type ? Conclure quant à l'action de **int()**

### Etude de la fonction `changer_type()` (code 2)

- ➔ remettez le # en début de ligne 14 que le code 1 ne vienne pas nous embêter.
- ➔ que vaut `liste2_nb` ? Expliquez ce que fait ce code.

### Etude de la fonction `changer_type2()` (code 3)

Remarquez juste : on n'a pas besoin d'envoyer la liste en paramètre !!! Ça fonctionne aussi bien !!

### Etude de la fonction `fct4()`

Imaginons un tableau de 9 cases où chaque case est repérée par un chiffre en A abscisse et par un nombre en ordonnée. Par exemple, les coordonnées de la case A0 sont [ " A ", " 0 "], les coordonnées de la case B2 sont [ " B ", " 2 "].

	0	1	2
A	A0	A1	A2
B	B0	B1	B2
C	C0	C1	C2

Imaginons aussi que ce tableau soit mémorisé dans une liste à deux dimensions

```
Liste = [ [A0,A1,A2] ,  
          [B0,B1,B2] ,  
          [C0, C1, C2 ] ]
```

Soit le code ci-dessous :

```
def fct4():
```

```
    A0=12  
    B0=4  
    A1=8  
    A2=7  
    B1=5  
    B2=9  
    C0=6  
    C1=78  
    C2=0
```

```
    Liste = [ [A0,A1,A2] ,  
              [B0,B1,B2] ,  
              [C0,C1,C2 ]]
```

```
    coordo_str = [input("entrez la lettre (A,B ou C): "), input("entrez le chiffre (0,1 ou 2): ")]  
    print ("coordo_str = ", coordo_str )  
    coordo_int = [ ord(coordo_str[0])-65 , int(coordo_str[1]) ]  
    print ("coordo_int = ", coordo_int )  
    valeur = Liste [ coordo_int [0] ] [ coordo_int [1] ]  
    print(" la valeur à cet emplacement est : ", valeur)
```

- ➔ que s'imprime-t-il à la console si on lance `fct4()` et qu'aux invitation on saisit B comme lettre et 2 comme chiffre ? Vérifiez dans *Spyder* (code 4)

## 2/ bataille navale (exécutable disponible ici : <http://ressource.elec.free.fr/docs/NSI/tkinter/bataille-navale-executable.zip> ou <https://bit.ly/39TWH8Q> )

Ouvrez dans *spyder* le fichier *bataille-navale.py*

On se propose de compléter un programme de bataille navale rudimentaire qui n'a pas été achevé.

Le fonctionnement doit être le suivant :

On choisit une coordonnée via les deux menus déroulant puis on clique sur « validez ». Si on a vu juste, il s'écrit « coulé » et la case devient verte.

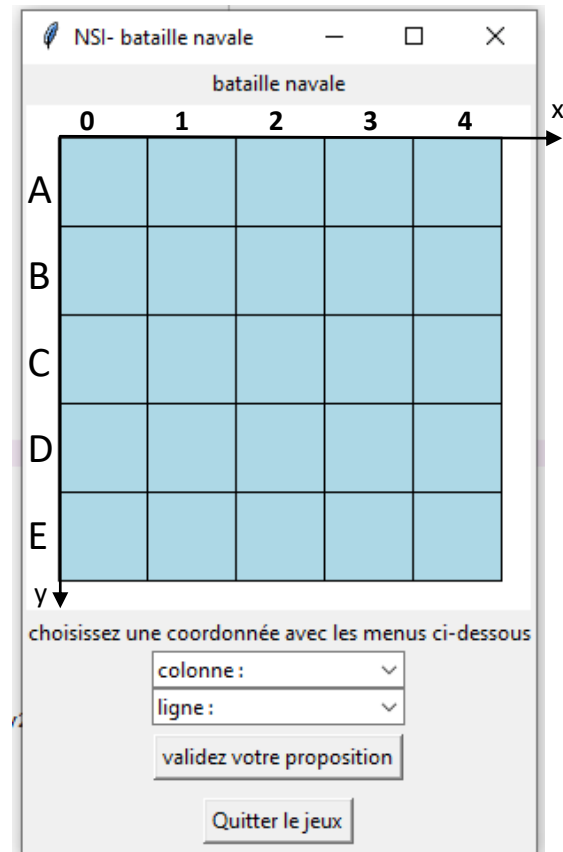
Si on loupe, il s'écrit « plouf » et la case devient rouge.

L'emplacement des bateaux est codé dans la liste **grille** (ligne 17 du programme). Un « 1 » veut dire qu'il y a un bateau ; un « 0 » veut dire qu'il n'y a pas de bateau.

Les caractéristiques graphiques des cases sont stockées dans les variables A0,A1 etc qui sont dans la liste **grilleG** (en vrai, c'est juste l'identificateur de la case qui est mémorisé).

Lorsqu'on choisit une ligne et une colonne, on mémorise dans la liste **choix** les valeurs (en char). Exemple : si vous choisissez A puis 2, alors on aura **choix** = [ " A ", " 2" ] .

Lorsqu'on clique sur « validez », on lance la fonction **jouer()** (ligne 45)



**Notre travail consiste à compléter cette fonction **jouer()** pour que le programme fonctionne.**

Pour ce, On dispose de deux fonctions qu'on pourra appeler dans **jouer()**:

- **fct\_coule()** qui fait le boulot à faire sur une case « coulé » (la mettre en vert et afficher « coulé »). Elle doit recevoir en argument une des variables de la liste **grilleG** (par exemple A1, ou B3, etc) Par exemple **fct\_coule(A1)** rendra la case de coordonnée A1 verte et affichera le message « coulé »
- **fct\_plouf()** qui fait le boulot à faire sur une case « à l'eau ». Elle doit recevoir en argument une des variables de la liste **grilleG** (par exemple A1, ou B3, etc) Par exemple **fct\_plouf(A1)** rendra la case de coordonnée A1 verte et affichera le message « loupé »

**jouer()** doit :

- créer une liste **choix\_nb** qui sera constitué des équivalent en entier de **choix** ( si **choix** = [ " A ", " 2 " ], alors **choix\_nb** devra valoir [0, 2]
- faire un affichage en console. Voila le code pour ça : **print ("la liste choix convertie en coordonnées pour grille: choix\_nb=",choix\_nb)**
- déterminer s'il y a un bateau. Pour ce, voir si à la coordonnée choisie, grille[][] vaut 1. Il faudra donc une structure en if .  
Par exemple, si **choix\_nb**= [2,1] , il faudra tester si **grille[2][1]** vaut 1 ou 0. Si c'est 1 -> il faut exécuter **fct\_plouf(...)** , si c'est 0 -> il faut exécuter **fct\_coule(...)**

A vous de jouer, si je puis dire 😊

# Créer un exécutable de notre programme :

---

Vous êtes peut-être frustré.e.s d'avoir créé un beau programme Python et de ne pas l'avoir en exécutable sous Windows.

Voilà comment transformer votre fichier .py en fichier .exe :

Nous allons utiliser un programme qui va faire ce travail, et qui s'exécutera en mode console : **PyInstaller**

## 1/ ouvrir le prompt anaconda (anaconda prompt)

1/ **voir si PyInstaller est déjà installé.** Pour ce, taper simplement PyInstaller dans la console. Si la console vous répond des trucs, c'est qu'il est installé. Si elle vous répond « *PyInstaller n'est pas reconnu...* », c'est qu'il n'y est pas et qu'il faut l'installer.

### **Pour Installer PyInstaller :**

Dans le prompt anaconda (anaconda prompt), saisissez : **python -m pip install PyInstaller**

Après téléchargement et installation, c'est bon.

Utiliser PyInstaller : **PyInstaller nom du fichier.py** depuis le répertoire où se trouve le fichier .py à rendre exécutable.

Deux répertoires se créent alors : dist et build. **Dans dist se trouve un répertoire du nom de votre fichier py dans lequel se trouve l'exécutable.**

Et voilà ! Votre fichier python exécutable windows est prêt !

## Pour aller plus loin :

➔ Modifier ce que bon vous semble dans notre programme (couleurs, texte, tailles....), ajouter un score, ...