

Combat naval

Enregistrez le **projet** dans un nouveau dossier **cnaval-login/** où **login** est votre identifiant de connexion, le fichier projet se nommera **cnaval.py**.

Vous disposez d'une grille de combat naval sur laquelle vous devez disposer quatre bateaux : un porte-avion, un croiseur, un destroyer et un sous-marin. Chaque bateau a une taille bien spécifique :

- Le **porte-avion** (porte-aéronefs) occupe 5 cases sur le plateau de jeu.
- Le **croiseur** cuirassé occupe 4 cases sur le plateau de jeu.
- Le **destroyer** (contre-torpilleur) occupe 3 cases sur le plateau de jeu.
- Le **sous-marin** occupe 2 cases sur le plateau de jeu.

	A	B	C	D	E	F	G	H	I	J
0										
1	x	x	x	x	x					
2										
3							x			
4					x		x			
5					x				x	
6			x	x	x	x			x	
7									x	
8										
9						x	x	x	x	x

Placement des navires

Tous les navires de guerre respectent les mêmes règles de placement, à savoir :

- Les navires ne peuvent pas se toucher (contre-exemple : E5 + E6).
- Les navires ne peuvent pas se trouver le long d'un bord du plateau de jeu (contre-exemple : F9 - J9).

Développement

1. Afficher la grille du combat naval.
2. Pour chaque bateau, demander à l'utilisateur :
 1. Quel type de bateau il désire placer : porte-avion, croiseur, destroyer ou sous-marin.
 2. La coordonnée du point à partir duquel le bateau sera placé, p.ex. **A4**
 3. L'orientation du bateau : horizontale (**H**) ou verticale (**V**).
3. Vérifier que les données entrées par l'utilisateur sont valides, qu'elles respectent bien toutes les règles de placement des navires.
4. Si les données ne sont pas correctes, réitérer la demande à l'utilisateur.

5. Si les données sont correctes, ajouter le navire sur le plateau et ré-afficher la grille. Le bateau sera représenté par la lettre 'x' pour chaque case qu'il occupe.

```

A B C D E F G H I J
0
1
2
3
4
5
6
7
8
9
Placement du porte-avion
  Où placez-vous votre porte-avion [XY]: A3
  Quelle orientation pour votre porte-avion [H/V]: H

```

Pour une meilleure visibilité, affichez les cases vides à l'aide d'un point « . ».

L'utilisation de **fonctions** pour la construction de votre code est fortement recommandée dans le sens où cela le clarifiera davantage. Vous créerez :

- **creation_grille(longueur, largeur)** : retourne une grille vierge de taille **longueur** x **largeur**.
- **affiche_grille(longueur, largeur, placements=None)** : affiche la grille de taille **longueur** x **largeur** en préfixant chaque ligne et chaque colonne du numéro ou de la lettre correspondant. Si le paramètre **placements** est donné (s'il ne vaut pas **None**), la fonction devra également afficher les bateaux dans la grille.

Astuce : pour gérer plus efficacement le placement des bateaux, transformez le dictionnaire **placements** en une liste contenant uniquement toutes les coordonnées des bateaux.

- **placement_bateau(long, larg, placements, position, orientation, taille)** : retourne le dictionnaire **placements** modifiés avec les coordonnées du nouveau bateau. L'ordre et l'appartenance de ces coordonnées n'a pas d'importance puisqu'il s'agit simplement de les afficher dans la grille sous forme d'un « x ». Les paramètres **long** et **larg** permettent de connaître la dimension de la grille.
 - **verif_orientation(long, larg, placements, position, orientation, taille)** : retourne **True** si la position et l'orientation du nouveau bateau sont valides, retourne **False** sinon. Les paramètres **long** et **larg** permettent de connaître la dimension de la grille.
 - **verif_collision(placements, position, orientation, taille)** : retourne **True** si la position et l'orientation du nouveau bateau le font entrer en collision avec un autre, retourne **False** sinon.

La fonction **creation_grille()** est appelée une seule fois, en début de jeu.

La fonction **placement_bateau()** sera appelée autant de fois qu'il y a de bateaux dans le jeu.

Toutes les fonctions de gestion du placement des bateaux seront enregistrées dans le fichier **libcnaval.py**.

Aucune demande d'entrée utilisateur (comprenez, appel à la fonction `input()`) ne sera effectué dans une fonction. Ces appels seront toujours présents dans le programme principal.

Structure des données

Vous disposez d'un dictionnaire identifiant chaque navire de guerre et sa taille :

```
navires = {1: ("porte-avion", 5), 2: ("croiseur", 4),  
           3: ("destroyer", 3), 4: ("sous-marin", 2)}
```

Vous devrez créer un autre dictionnaire contenant pour chaque bateau ses coordonnées (sous forme de tuple) sur le plateau de jeu. Par exemple, la grille ci-dessus, le dictionnaire devra contenir :

```
placements = {1: ('A1', 'B1', 'C1', 'D1', 'E1'),  
              2: ('C6', 'D6', 'E6', 'F6'),  
              3: ('I5', 'I6', 'I7'),  
              4: ('G3', 'G4')}
```

Trucs et astuces

Pour les dictionnaires, la méthode `items()` retourne une liste de tuples (clé, valeur) à partir du dictionnaire (*Apprendre à programmer avec Python 3*, p.156).

P.Ex. :

```
pc = {"cpu" : "2.6Ghz", "ram" : "8Go", "ssd" : "25To"}  
pcitems = pc.items() # -> [("cpu", "2.6Ghz"), ("ram", "8Go"), ("ssd", "25To")]  
for cle, val in pcitems :  
    print("{} : {}".format(cle, val))
```

Pour les chaînes de caractères, la méthode `index()` retourne l'index du premier caractère recherché dans la chaîne (*Apprendre à programmer avec Python 3*, p.138).

P.Ex. :

```
ch = "destroyer"  
idx = ch.index('t') # idx vaut 3
```