UAA4 Mini projets de programmation impérative en Python

1 Compléments Python

Le programme peut tirer un nombre au hasard grâce aux fonctions de la librairie random. Il faut donc rajouter leur import avec l'instruction suivante :

```
from random import *
```

Tirer un entier entre 0 et 100 par exemple se fait avec la fonction **randint**.
Teste cette fonction avec le code suivant (while(1) crée une boucle infinie, tu peux arrêter son exécution en tapant **CTRL+C**e)

```
while(1):
    print(randint(0,100))
```

2 Nombre mystère

Prérequis : input(), if … else, while (tp8 boucles conditionnées)

Règles du jeu :

Un joueur choisit un nombre inférieur à 100 au hasard.
Un autre doit découvrir le nombre tiré en un minimum d'essai sachant qu'on lui répond pour chaque essai si le nombre tiré est plus grand ou plus petit qu'il propose.

Consignes

Ecris un programme qui te permette de jouer contre l'ordinateur.
Il faudra afficher en fin de partie le nombre d'essais nécessaires pour trouver le nombre.
Pour arrêter la boucle lorsque le nombre est trouvé, soit tu peux utiliser une boucle while() , soit tu peux utiliser l'instruction break au sein d'une boucle for (la boucle est stoppée et le programme passe à l'instruction suivante après la boucle)

Résultats

```
Quel est le nombre ?
50
c'est moins !
Quel est le nombre ?
25
C'est plus !
Quel est le nombre ?
37
C'est plus !
```

```
Quel est le nombre ?
43
Bravo, t'as trouvé le nombre mystère! en  4  coups
```

3 Voyageurs dans le tram

Prérequis : input(), if … else, for

Consignes

Ecris un programme qui calcule le nombre de voyageurs présents dans un tram tout au long de son parcours.
Ton programme doit lire le nombre d'arrêts.
Pour chaque arrêt :

- ✓ Ton programme lit le nombre de voyageurs qui montent dans le tram et le nombre qui en descendent
- ✓ Il calcule le nombre de voyageurs présents dans le tram

A fin du parcours, il affiche le nombre de voyageurs qui ont pris le tram

Améliorations

Le tram a une capacité maximale de 100 personnes, au-delà, les voyageurs sont refusés.

Résultats

```
Combien de fois s'arrête le tram ?
2
Arrêt no:  1
Combien de voyageurs veulent monter?
15
Combien de voyageurs descendent?
5
10  voyageurs sont dans le tram


Arrêt no:  2
Combien de voyageurs veulent monter?
110
Combien de voyageurs descendent?
5
100  voyageurs sont dans le tram


110  voyageurs ont pris le tram sur la ligne
```

4 Résolution des équations du 1er et 2ème degré

Prérequis : input(), if … else

Consignes

Ecris un programme permettant de résoudre une équation à coefficients réels de la forme $ax^2 + bx + c = 0$ (a, b et c seront entrés au clavier, vois les formules dans le pdf du drive).
Attention à bien prendre en compte toutes les valeurs que peuvent prendre a, b, c (notamment quand elles sont nulles)

Résultats

```
Résolution d'une équation a x + b = 0
Entre a : 2
Entre b : 12
La solution est x = -6


Résolution d'une équation a x^2 + b x + c = 0
Entre a : 1
Entre b : 1
Entre c : -6
Les solutions x1 = 2 et x2 = -3
```

5 Mini formulaire

Prérequis : input(), if … else, while

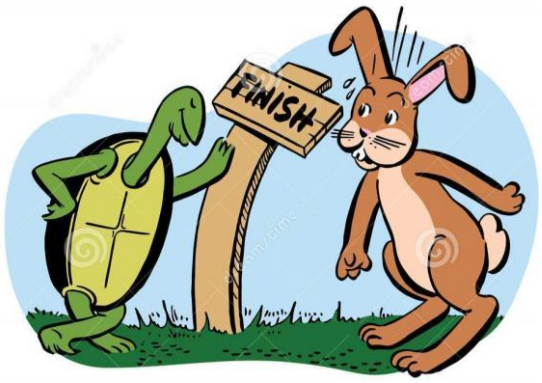
Consignes

Crée un menu te permettant de calculer au moins 3 formules de 4èmes (de tes cours de math ou sciences, vois la liste des formules proposées dans l'annexe en fin de ce document) ; pour chaque formule, un sous menu doit proposer à l'utilisateur de choisir la variable à calculer.

6 Le lièvre et la tortue

Prérequis : input(), if … else, while

Règles du jeu :



Un lièvre et une tortue font une course selon les règles suivantes :
A chaque tour, on lance un dé non truqué à 6 faces. SI le 6 sort, le lièvre gagne la partie, sinon, la tortue avance d'une case. La tortue gagne quand elle a avancé 6 fois.

Consignes

Ecris un programme permettant de prévoir qui a la plus grande chance de gagner ; pour cela cela, il devra simuler 1000 parties et afficher la fréquence de victoire de chacun.
Fais ensuite varier le nombre de lancers des dés afin de vérifier quand les résultats s'inversent.

7 Jeu de pierre, papier, ciseaux

Prérequis : input(), if … else, while (tp8 boucles conditionnées)

Règles du jeu :

Deux joueurs se montrent simultanément leur main qui symbolisera une pierre (poing fermé), un papier (main tendue) ou des ciseaux (l'index et le majeur forment un V).
La pierre bat les ciseaux, les ciseaux battent le papier et le papier bat la pierre. Si les deux joueurs jouent le même symbole, il y a égalité.

Consignes

Ecris un programme qui te permette de jouer contre l'ordinateur en 5 manches.
Le choix de la pierre, du papier ou des ciseaux se fera respectivement par 1, 2, 3.

Améliorations

Demande le choix 1, 2, 3 jusqu'à ce que la réponse tapée soit correcte.
Rajoute la possibilité de jouer soit contre l'ordinateur, soit à 2 joueurs.
Le jeu doit continuer jusqu'à ce que le joueur tappe -1

Résultats

```
Comment veux-tu jouer?
1: contre l'ordinateur | 2: à 2 joueurs)
1
Que choisis-tu ?
1 : pierre, 2 : papier, 3 : ciseaux ?
5
Que choisis-tu ?
1 : pierre, 2 : papier, 3 : ciseaux ?
1
Le programme a choisi:  2 | Tu as choisi:  1
Le programme a gagné le point
Score du programme:  0 | Ton score:  1
…
Que choisis-tu ?
1 : pierre, 2 : papier, 3 : ciseaux ?
2
Le programme a choisi:  3 | Tu as choisi:  2
Le programme a gagné le point
Score du programme:  5 | Ton score:  5
Le programme a gagné la partie
```

8 Jeu de nim

Prérequis : input(), if … else, while

Règles du jeu :

20 bâtonnets en bois sont alignés les uns à côté des autres au centre de la table.
A tour de rôle chacun des 2 concurrents va devoir en retirer 1, 2 ou 3 à l'endroit de leur choix, le but étant de laisser le dernier bâtonnet à son adversaire. Il leur faudra donc calculer et ne pas retirer de bâtonnet(s) au hasard.
Celui qui se retrouve avec le dernier bâtonnet a perdu.

Consignes

- ✓ Ecris un programme à 2 joueurs
- ✓ Ecris une 2ème version ou un joueur peut jouer contre l'ordinateur. Pour cela, implémente l'algorithme proposé par Fort Boyard qui devrait rendre ton programme imbattable :

  https://www.fan-fortboyard.fr/pages/emission/conseil/batonnets.html

```
||||||||||||||||||||
Joueur n° 1
Combien de batonnets retires - tu ? 3


|||||||||||||||||| ...
Joueur n° 2
Combien de batonnets retires - tu ? 2


|||||||||||||||| .....
Joueur n° 1
Combien de batonnets retires - tu ? 3
…
||| ..................
Joueur n° 2
Combien de batonnets retires - tu ? 2
Le joueur n° 2 a gagné !
```

9 Jeu de dé : le Zanzibar

Prérequis : input(), if … else, for, while

Règles du jeu :

Le Zanzibar est un des plus vieux jeu de dé et est à l'origine du 421.
A chaque tour, le joueur peut lancer 3 fois les dés pour obtenir le meilleur Zanzi, c'est-à-dire un brelan (3 valeurs identiques). Il peut reprendre les 3 dés, 2 dés ou un seul à sa convenance. Il peut décider d'arrêter dès le premier jet de dés, ou le deuxième. Les autres joueurs devront alors essayer de faire un meilleur Zanzi sur le même nombre de jet de dés.
Ordre de valeurs des Zanzis :

- 3 x As : 100points
- 3 x 6 : 60 points
- 3 x 5 : 5 points
- 3 x 4 : 4 points
- 3 x 3 : 3 points
- 3 x 2 : 2 points

Pour gagner une partie de Zanzibar, il faut être le premier joueur à atteindre le score fixé dès le début entre les joueurs (200, 500, 1000 …)

Consignes

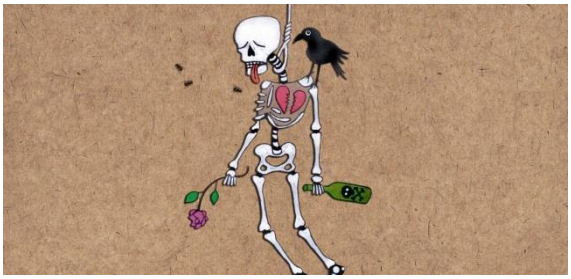
Crée une 1ère version de programme qui simule le lancer de dés et permet de jouer seul
Crée une 2ème version qui permet de jouer à 2
Crée une 3ème version qui permet à un joueur de jouer contre le programme.

```
Lancer n° 1 : 1 1 6
Veux-tu relancer le 1er dé ? (o/n) n
Veux-tu relancer le 2e dé ? (o/n) n
Veux-tu relancer le 3e dé ? (o/n) o
Lancer n° 2 : 1 1 2
Veux-tu relancer le 1er dé ? (o/n) n
Veux-tu relancer le 2e dé ? (o/n) n
Veux-tu relancer le 3e dé ? (o/n) o
Lancer n° 3 : 1 1 1
tu as marqué 100 points. Tu as donc au total 100 points
```

10 Jeu du pendu

Prérequis : input(), if … else, while et chaînes de caractères (1ère version), listes à 1 dimension (2ème version) (tp9 Chaînes et tp10 Listes), lecture de fichier (3ème version)

Règles du jeu :



Le Pendu est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent.

Consignes

- ✓ Ecris un programme en 3 étapes :

  Dans une 1ère version, un joueur choisit un mot de 8 lettres maximum, un autre joueur tente de trouver les lettres composant le mot.

  Dans une 2ème version, l'ordinateur choisit un mot au hasard dans une liste fournie.
  Dans une 3ème version, l'ordinateur choisit un mot dans un fichier (un dictionnaire).

- ✓ À chaque coup, le joueur saisit une lettre.

- ✓ Si la lettre figure dans le mot, l'ordinateur affiche le mot avec les lettres déjà trouvées. Celles qui ne le sont pas encore sont remplacées par des barres (_).

- ✓ Le joueur a 6 chances. Au delà, il a perdu.

Résultats

```
Jeu du pendu: tu as 6 vies pour trouver le mot

_ _ _ _ _ _

Choisis une lettre

w

Raté !

_ _ _ _ _ _

Encore  6  lettres à trouver ! Il te reste:  5  vies

Choisis une lettre

j

Bravo !

j _ _ _ _ _
```

```
Encore  5  lettres à trouver ! Il te reste:  5  vies
Choisis une lettre
j
Bravo !
... mais tu as déjà choisi cette lettre
j _ _ _ _ _
...
...
j a m _ o n
Encore  1  lettres à trouver ! Il te reste:  5  vies
Choisis une lettre
b
Bravo !
j a m b o n
Bravo, tu as gagné !
```

11 Jeu de dé : Le Yams

Prérequis : input(), if … else, for, while, listes à une dimension

Règles du jeu :

Le Yams se joue avec 5 dés et se finit une fois toutes les cases de la fiche de score remplies. Chaque joueur joue tout à tour et dispose de 3 lancers à chaque coup. L'objectif étant de réaliser des combinaisons qui rapportent des points. Le joueur a le choix de reprendre tous ou une partie des dés à chaque lancé, selon son gré, pour tenter d'obtenir la combinaison voulue. A chaque tour, le joueur doit obligatoirement inscrire son score dans une des cases de la grille.

Il peut arriver lors d'un tour que le résultat ne convienne pas au joueur et qu'il se dise qu'il pourrait faire un plus grand score sur un autre tour. Il peut alors choisir de barrer une autre case à la place. Bien entendu, il ne pourra plus faire cette combinaison par la suite.

Le gagnant d'une partie de Yams est le joueur qui comptabilisera le plus de points à la fin des 10 coups.

Pour ce projet, plusieurs versions sont possibles, suivant votre niveau en programmation Python. Plus le niveau est élevé, meilleure est la note, bien sûr. Mais l'objectif est d'arriver à créer un premier programme qui fonctionne, ne soyez pas trop ambitieux, il est préférable de produire un programme simple mais qui fonctionne. Je vous conseille donc de vous lancer dans le projet « débutant » puis de l'améliorer en passant au niveau avancé 1 puis 2 etc…

Consignes

Niveau 1 : Tirage unique

Crée un programme qui simule le lancer de 5 dés, stocke les nombres tirés dans une liste de 5 éléments, le trie, puis affiche le tirage obtenu dans une liste de 5 éléments appelée « Tirage ».

Deux fois de suite, le programme demande combien de dés le joueur veut relancer, puis demande les uns après les autres les dés à relancer. Il simule le nouveau tirage et l'affiche.

Niveau 2 : Jeu entier à 1 joueur

Un jeu entier pour un joueur avec l'affichage du score à la fin.

Pour cela, on utilisera 3 listes :

- ✓ La première liste : « tirage » qui contient le tirage (éventuellement modifiée deux fois). Par exemple [1,1,2,3,4]. Elle est recréée à partir d'une liste vide à chaque tirage.

- ✓ Une deuxième liste appelée « repartion » qui contiendra 6 éléments : le nombre de 1, de 2, de 3 , de 4, de 5 et de 6. Pour l'exemple précédent ce sera [2,1,1,1,0,0]. Elle est recréée à partir d'une liste vide à chaque tirage. Elle permet de rapidement calculer le score.

- ✓ Une troisième liste appelée « score »est utilisée tout au long du jeu et mise à jour à chaque tirage.

Pour chaque tirage, le programme propose au joueur les « cases » du score non encore remplies et lui demande de choisir. Le programme calcule le score (par exemple si le joueur choisit carré, il vérifie q'il y a un carré et attribut le total des 4 dés à la case carré.
Après les 10 tirages, le score est calculé et affiché.

Joueurs	**Joueur1**	**Joueur2**
Total de 1		
Total de 2		
Total de 3		
Total de 4		
Total de 5		
Total de 6		
Total		
Si total > 63 alors bonus de 35 points		
Total 1		
Brelan (Total des 3 dés)		
Carré (total des 4 dés)		
Full (25 pts)		
Petite suite (30 pts)		
Grande suite (40 pts)		
Yams (50 pts)		
Chance (total des 5 dés)		
Total 2		
Total		

12 Autres jeux de dé

https://www.regles-de-jeux.com/jeux-de-des/

12.1 Le 421

12.2 Le 5000

12.3 Les 50 points

12.4 Dés bloqués

12.5 Le 36

12.6 Bunco

13 Master mind

Prérequis : input(), if … else, for, while,(listes à une dimension)

Règles du jeu :

Le mastermind est un jeu de société qui se joue à deux. Les joueurs disposent de pions colorés (nous considérerons les couleurs violet, jaune, rouge, orange, rose, bleu foncé, bleu clair, vert et le vide). Un joueur choisit un code secret de quatre pions parmi ces couleurs. Le second joueur va essayer de trouver le code. Il a droit à 10 essais. À chaque proposition qu'il fait, le premier joueur lui donne deux indications~: le nombre de pions de la bonne couleur qui sont bien placés (nbbp), et le nombre de pions de la bonne couleur qui sont mal placés (nbmp).

1. 1ère version de code

Consignes

- ✓ Ecris un programme qui te permette de jouer contre l'ordinateur.
- ✓ Ecris une **1ère version** où le code secret est représenté par des entiers.

  Nous considèrerons qu'il y a 9 couleurs possibles représentées par les entiers de 0 à 8 et que le code secret a une longueur de 4.

Résultat

```
Tour 1
Quelle est ta proposition ?
1 2 5 6
Tu as 0 pions bien placé(s) et 0 pions mal placé(s)
Tour 2
Quelle est ta proposition ?
1 4 5 6
Tu as 0 pions bien placé(s) et 1 pions mal placé(s)
Tour 3
```

```
Quelle est ta proposition ?
4 3 4 8
Tu as 2 pions bien placé(s) et 2 pions mal placé(s)
Tour 4
Quelle est ta proposition ?
4 3 8 4
Tu as 4 pions bien placé(s) et 0 pions mal placé(s)
Bravo, tu as gagné !
Le code secret était: 4 3 8 4
```

Etapes de jeu

- ✓ générer un code secret
- ✓ initialiser le numéro du tour à 1
- ✓ répéter au maximum 10 fois (on pourra s'arrêter plus tôt si le joueur a gagné)
  - o afficher le numéro du tour
  - o demander la proposition du joueur
  - o calculer le nombre de pions bien placés et le nombre de pions mal placés
  - o afficher ces nombres
  - o si le nombre de pions bien placés n'est pas égal à la taille du code secret, augmenter le numéro du tours et recommencer
- ✓ afficher un message (gagné/perdu) et afficher le code secret.

Aide

- ✓ Tu peux lire 4 entiers sur une ligne en utilisant la fonction **split** : elle permet de découper une chaîne de charactères en plusieurs éléments (les caractères étant délimités par des espace). Il faut ensuite les convertir en tier

code

```
p1, p2, p3, p4 = input("entre  4 entiers séparés par un espace: ").split()
p1, p2, p3, p4 = int(p1), int(p2), int(p3), int(p4)
```

- ✓ Tu peux tester si un entier est égal à un des 4 entiers en utilisant l'instruction if … in

code

```
if p1 in (1, 2, 3, 4):
    print(p1, " est égal à 1 ou 2 ou 3 ou 4")
```

2. 2ème version de code

✓ Ecris une **2ème version** où tu fais correspondre à chaque nombre une couleur.

Une fois entré ton choix, le programme re-affiche sur la même ligne ton choix suivi du numéro de tours, des nombres nbbp et nbmp

Résultat

```
Tour 1 - Quelle est ta proposition ?
3 4 5 8
   0 7 6 1          1 0 0
Tour 2 - Quelle est ta proposition ?
3 4 5 0
   0 7 6 1          2 0 1
Tour 3 - Quelle est ta proposition ?
0 3 4 5
   0 7 6 1          3 1 0
Tour 4 - Quelle est ta proposition ?
0 7 6 1
   0 7 6 1          4 4 0
Bravo, tu as gagné !
Le code secret était:  0 7 6 1
```

Aide : installation de la libraire colored dans un terminal

```
pip install colored
```

Tirage au sort de 4 nombres et affichage avec leur couleur associée

```
from colored import Fore, Back, Style

# Fore: couleur du texte, Back: couleur de l'arrière-plan, Style : mettre en gras

couleur=[Back.red, Back.green, Back.yellow, Back.blue, Back.magenta, Back.cyan, Back.deep_pink_4a,
Back.dark_orange_3a, Back.grey_0]


code1, code2, code3, code4 = randint(0,8), randint(0,8), randint(0,8), randint(0,8)

print(Style.reset, Fore.white, Style.BOLD, couleur[code1], code1, couleur[code2], code2,
couleur[code3], code3, couleur[code4], code4, Style.reset)
```

3. 3ème version de code

Après avoir suivi les **tp9 Chaînes et tp10 Listes**, optimise ton code en définissant les entiers des pions avec une liste

1 Jeu de l'Awale

Prérequis : input(), if … else, for, while, listes à une dimension

2 Jeu d'XO

Prérequis : input(), if … else, for, while, listes à 1 ou 2 dimensions

Règles du jeu :

Deux joueurs, appelés "X" et "O" marquent à tour de rôle les espaces dans une grille 3×3. Le joueur qui a réussi à placer trois marques respectives dans une rangée horizontale, verticale ou diagonale gagne la partie

3 Pygame zero

Fais un jeu dans lequel tu peux déplacer un personnage avec les flèches afin d'éviter des obstacles qui traverse l'écran.

⇨ Voir le document dans le drive.

4 Annexe - Exemples de formules de Sciences et Math

Tu peux choisir d'autres formules sous réserve d'accord du professeur.

4.1 Chimie

1. Mole-molécule

$$x = n N_A$$

Où x = nombre d'atomes, de molécules ou d'ions
n = nombre de mole
N_A = constante d'Avogadro

2. Concentration molaire

$$C = \text{Error!}$$

3. Dilution :

$$C_i V_i = C_f V_f$$

Où C_i est la concentration de la solution initiale
C_f est la concentration de la solution finale
V_i est le volume de la solution initiale
V_f est le volume de la solution finale

4.2 Physique

1. MRU-Position

$$x(t) = xo + vo\ t$$

Où x(t) est la position à l'instant t
xo est la position initiale (t=0sec)
vo est la vitesse initiale
t est le temps

2. MRUA-Vitesse

$$v(t) = vo + a \cdot t$$

Où v(t) est la vitesse à l'instant t
vo est la vitesse initiale
a est l'accélération (constante)
t est le temps

3. MRUA - Position

$$x(t) = xo + vo\ t + a$$

Où x(t) est position à l'instant t
xo est la position initiale
a est l'accélération (constante)
t est le temps

4. Energie potentielle

$E_p = m\ g\ h$

Où E_p est l'énergie potentielle
m est la masse
g est la constante de gravitation
h est la hauteur

5. Energie cinétique

$E_c =$ Error! $m\ v^2$

Où E_c est l'énergie cinétique
m est la masse
v est la vitesse

6. Energie mécanique et puissance

$E_m = E_p + E_k$

$P = \frac{E_m}{t}$

Où E_m est l'énergie mécanique
E_p est l'énergie potentielle
Ec est l'énergie cinétique
P est la puissance
T est le temps

4.3 Mathématiques

1. Equation du second degré : $f(x) = ax^2 + bx + c = 0$

- ✓ Si a = 0, l'équation $bx + c = 0$ admet une racine:

  $x_1 = -c/b$

  Tableau de signe : la fonction possède le même signe que a pour des valeurs de x à droite de la racine, et de signe opposé à gauche de la racine.

On calcule $\Delta = b^2 - 4ac$

- ✓ Si $\Delta > 0$, l'équation admet deux racines distinctes :

  $$x_1, x_2 = \frac{-b \pm \sqrt{\Delta}}{2a}$$

  Factoriation : $\boxed{ax^2 + bx + c = a(x - x_1)(x - x_2)}$

- ✓ Si $\Delta = 0$, l'équation admet une racine double:

  $$x_1 = x_2 = -\frac{b}{2a}$$

  Factorisation : $\boxed{ax^2 + bx + c = a(x - x_1)^2}$

- ✓ Si $\Delta < 0$, l'équation n'admet pas de racine réelle ; on ne peut pas le factoriser.

- ✓ Tableau de signe (à insérer dans les 3 cas du signe ci-dessus en fonction du signe de Δ) : la fonction possède le même signe que ***a*** sauf pour les valeurs de *x* comprises entre ses racines lorsque celles-ci existent.

2. Formules de statistiques :

⇨ voir le tp sur les tableaux dans le drive.

- ✓ Moyenne : $m = \frac{1}{N} \sum_{i=1}^{Nc} n_i x_i$
- ✓ Variance : $V = \frac{1}{N} \sum_{i=1}^{Nc} (x_i - m)^2$
- ✓ Ecart-type : $\sigma = \sqrt{V}$