TP03

**Consignes impératives !**

A partir de ce TP, les affichages (sorties) et noms de fonctions doivent respecter exactement l’énoncé. Si une fonction vous est fournie, il ne faut pas modifier son nom ! Si elles ne sont pas fournies mais explicitées dans l’énoncé, créez-les **avec le même nom, et les paramètres dans le même ordre** ! Tout changement de nom ne sera simplement pas pris en compte par mon script de correction et donc le point ne sera pas accordé. Vous pouvez toujours créer des fonctions supplémentaires.

L’objectif de ce TP est de vous faire pratiquer l’utilisation des fonctions et affiner vos connaissances des boucles for et while.

1. **Jeu du pendu**

**Autorisé : Boucles, if elif, find(), print(), input()**

Vous devez programmer un jeu du pendu. Pour rappel, le jeu du pendu est un jeu ou l’objectif est de découvrir un mot caché en choisissant une lettre. La longueur du mot caché est connue car il est affiché sous cette forme :

Pour le mot « Bonjour » : \_ \_ \_ \_ \_ \_ \_

Si le mot caché contient cette lettre, l’indice est mis à jour avec cette lettre.

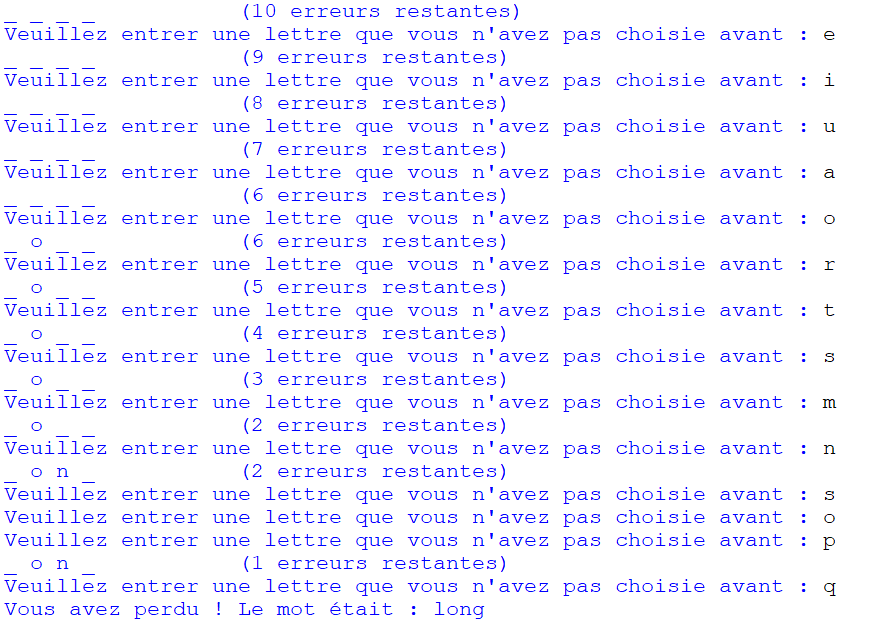
Choix de la lettre « o » : \_ o \_ \_ o \_ \_

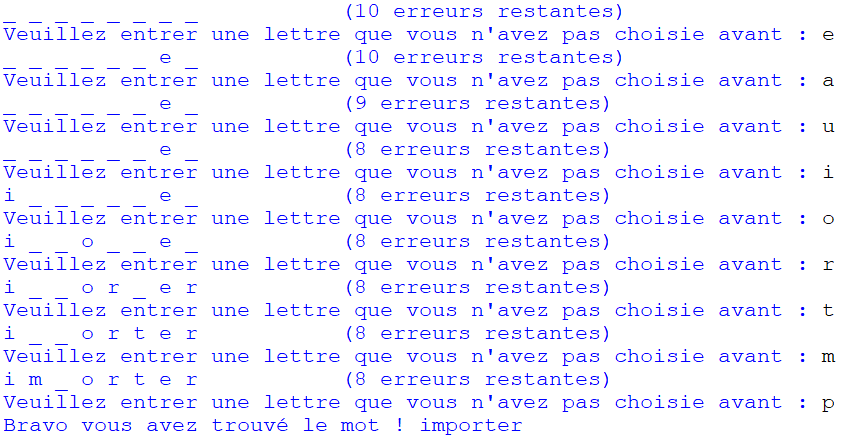
Sinon le joueur perd un essai. Si le joueur atteint un nombre prédéfini d’essais manqués, le joueur perds la partie. S’il trouve le mot avant, il gagne la partie. Le joueur possède **10 essais** et le mot caché est sélectionné aléatoirement dans une liste de mots. Le mot caché se trouve dans la constante MOT mise à votre disposition.

Pour ce faire vous devrez séparer votre code en plusieurs fonctions.

* 1. La fonction affichage\_mot\_cache qui prends en paramètres le mot caché, les lettres déjà tirées et le nombre d’erreurs restant
     1. Cette fonction affichera le mot caché et les lettres déjà trouvées sous la forme vu ci-dessus
     2. Elle affichera également le nombre d’erreurs restantes (voir exemple de sortie)
  2. La fonction tirage\_lettre qui prend en paramètres les lettres déjà tirées. Celle-ci retournera les lettres déjà tirées avec la nouvelle lettre venant d’être choisie
     1. Cette fonction demandera à l’utilisateur de saisir une lettre et reposera la question tant que la lettre entrée a déjà été choisie
  3. La fonction verification\_mot qui prend en paramètres le mot et les lettres déjà tirées. Celle-ci retournera un booléen si le mot a été trouvé ou non
     1. Cette fonction vérifiera si le mot a effectivement été trouvé.
  4. La fonction diminuer\_erreurs qui prend en paramètre le nombre d’erreurs restants et retourne le nombre d’erreurs restant diminué de 1.

Exemple de sortie :





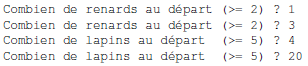
1. **Proies et prédateurs**

**Autorisé : Boucles, if elif, print(), input(), round(), abs()**

Le but de cet exercice est de simuler l’évolution au cours du temps d’une population de prédateurs (des renards) et de proies (des lapins).

La première partie du programme permet la saisie des populations initiales. Cette saisie est redemandée tant qu’elle ne respecte pas la règle.

Exemple de déroulement



Pour cela il vous faudra créer deux fonctions :

* initialisation\_lapins sans paramètres et qui retournera un int représentant le nombre de lapins saisi par l’utilisateur
* initialisation\_renards sans paramètres et qui retournera un int représentant le nombre de renards saisi par l’utilisateur

La seconde partie consiste à simuler l’évolution des populations de lapins et de renards, chaque mois pendant 50 mois.

Pour cela il vous faudra créer une fonction simulation prenant en paramètres la durée, le nombre de lapins et le nombre de renards.

Le taux de croissance des lapins est de 40%, le taux d’attaque est de 1% et le taux de croissance des renards est de 0.8%. Le taux de mortalité des renards est de 10%.

La croissance des lapins est définie par :

nb\_lapins\*(1.0 + taux\_croissance\_lapins - taux\_attaque \* nb\_renards)

Note : nb\_renards est le nombre de renards du mois précédant le calcul

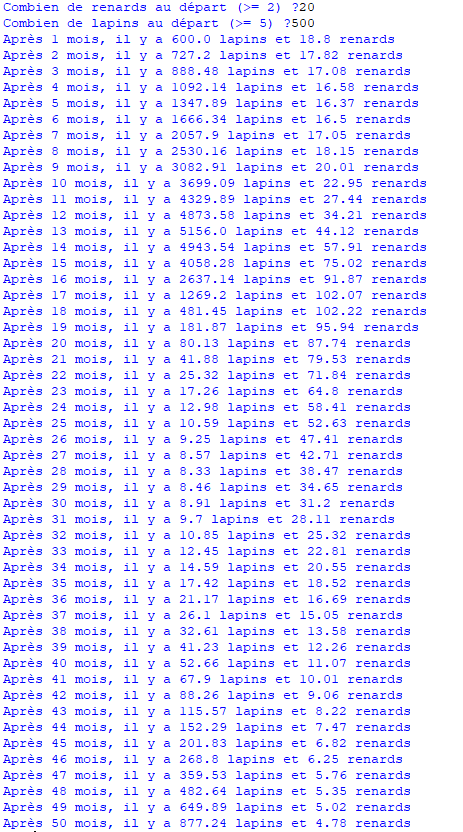
La croissance des renards est définie par :

nb\_renards \*= (1.0 + taux\_attaque \* nb\_lapins \* taux\_croissance\_renards – taux\_mortalite)

Note : nb\_lapins est le nombre de lapins du mois précédant le calcul

Si le nombre de lapins ou de renards descend dans les négatifs, vous devez afficher 0.0 à la place.

Exemple de sortie :



1. **Proies et prédateurs V2.0**

**Autorisé : Boucles, if elif, print(), input(), round(), abs()**

**Reprendre le code du tp03\_ex02.py et l’adapter dans le fichier tp03\_ex03.py**

Dans une première partie nous allons simplifier l’affichage. Au lieu d’afficher chaque mois la progression des populations, nous allons afficher uniquement l’étape finale (au 50ème mois).

Nous allons ajouter quelques messages selon l’évolution des populations au cours de ces 50 mois passés.

Si durant la simulation le nombre de renards ou de lapins descend en dessous de 5, il doit y avoir un message indiquant :

Les ... ont été en voie d’extinction

(avec ... soit renards soit lapins).

Si durant la simulation le nombre de renards ou de lapins descend en dessous de 5 mais remonte au-dessus de 5 par la suite, le message précédent s’affiche (puisqu’on est descendu en dessous de 5) mais est suivi de :

mais la population est remontée ! Ouf !

Si la population de renards ou de lapins passe en dessous de 2, nous supposons que l’espèce ne peut plus se reproduire. Dans ce cas de figure, le nombre de renards ou de lapins doit alors être mis à zéro (durant la simulation) et le message suivant affiché (à la fin de la simulation) :

et les ... ont disparus :-(

(avec ... soit renards soit lapins).

Si durant la simulation, aucun des événements décrits ci-dessous n’est arrivé, affichez à la fin de la simulation

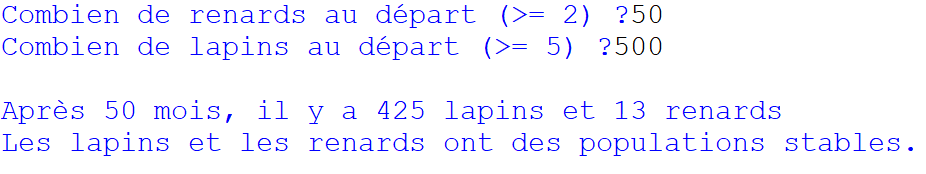
Les lapins et les renards ont des populations stables.

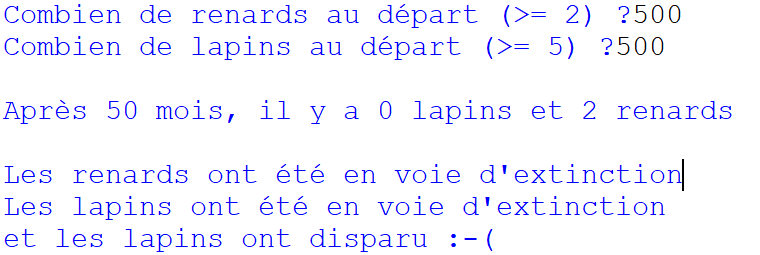
Finalement, modifiez votre code de sorte que l’on arrête aussi la simulation si le nombre de renards et le nombre de lapins sont tous les deux plus petits que 2 (plus de reproductions possible).

Essayez de réfléchir dans quel cas il est judicieux d’utiliser une fonction ou non. Votre programme doit conserver les 3 fonctions décrites au point 2) !

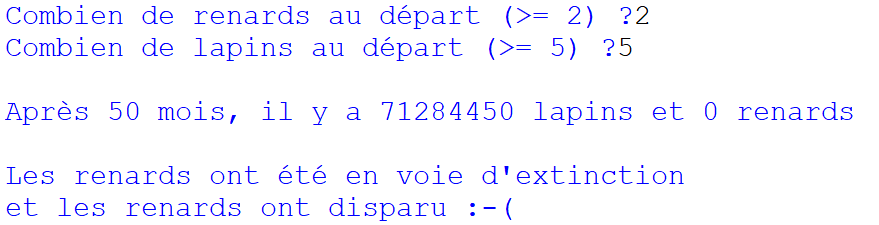
Exemple de sortie

1)

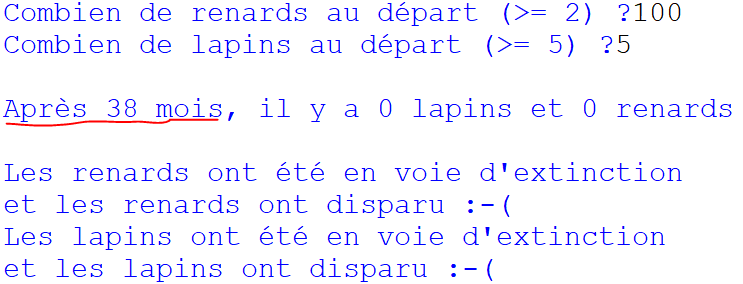


2)

3)



4)



1. **Date de Pâques**

**Autorisé : Boucles, if elif, print(), input()**

Le but de cet exercice est de déterminer la date du dimanche de Pâques (chrétiennes grégoriennes). On demande à l’utilisateur d’entrer une année et le programme affiche la date de Pâques de l’année correspondante.

Pour cela on vous demande d’écrire trois fonctions :

1. Une fonction demander\_annee qui ne prends pas de paramètres et retourne un entier. Cette fonction doit :
   1. Demander une année à l’utilisateur (entre 1583 et 4000). Tant que l’année saisie n’est pas conforme, cette fonction devra reposer la question
   2. Retourner l’année (correcte) saisie
2. Une fonction affiche\_date qui prend deux entiers en paramètres : une année et un chiffre entre 22 et 56. Ce chiffre représente la date de pâques possible comprise entre le 22 mars et le 25 avril ( 22 mars jusqu’au 31+25 avril). Cette fonction doit :
   1. Afficher le message « Date de Pâques en » suivi de l’année reçue en premier paramètre, suivi de « : », et de la date de Pâques (Voir Exemple de sortie)
   2. Il faut prendre en considération le fait d’afficher le bon mois (mars ou avril) et le bon jour (Le 55 avril n’existant pas).
3. Une fonction date\_paques qui reçoit une année en paramètre et retourne un entier entre 22 et 56. Il s’agit de [l’algorithme de Gauss](https://fr.wikipedia.org/wiki/Calcul_de_la_date_de_P%C3%A2ques_selon_la_m%C3%A9thode_de_Gauss). **Toutes les divisions ci-dessous sont des divisions entières (à l’exception des modulos).**
   1. Le siecle il suffit de diviser l’année par 100
   2. Une valeur p qui vaut 13 plus 8 fois le siecle, le tout divisé par 25
   3. Une valeur q, division du siecle par 4
   4. Une valeur m valant 15 - p + siecle - q, le tout modulo 30
   5. Une valeur n valant 4 + siecle - q modulo 7
   6. Une valeur d qui vaut m plus 19 fois « l’année modulo 19 », le tout modulo 30
   7. Une valeur e qu’il serait trop compliqué de décrire en français et que nous vous donnons directement :   
      (2\*(annee % 4) + 4 \*(annee % 7) + 6 \* d + n) % 7
   8. Le jour (ou presque, voir ci-dessous) : e plus d plus 22.

La valeur du jour doit cependant encore être corrigée dans certains cas particuliers :

Si e vaut 6 et que d vaut 29 ou 28 et que 11\*(m+1) modulo 30 est inférieur à 19, alors il faut soustraire 7 au jour.

C’est cette valeur (jour) que devra renvoyer la fonction date\_paques.

Exemple de sortie



