# Exercices sur les tests

## Exercice n°1

Ecrire un programme (`Queloc.py`) qui demande à un utilisateur de donner un heure (heure, minute seconde). Le programme convertira et affichera cet horaire en nombre de secondes. L'affichage ne sera fait que si `TempsS` est inférieur ou égal à 86399.

## Exercice n°2

Ecrire une fonction (`C_Div.py`) qui prendra 2 arguments d'entrée, qui teste si le premier argument est divisible par le second. Elle retournera un entier qui sera positionné à 1 si le nombre est effectivement divisible, -1 sinon.

## Exercice n°3

Reprendre la fonction `Racine_V2.py` pour prendre en compte le fait que le discriminant peut être négatif.

La fonction retournera toujours 2 arguments V1 et V2 comme suit :

- V1 et V2 seront les racines réelles si $\Delta \geq 0$
- V1 sera le partie réelle et V2 la partie imaginaire si $\Delta < 0$

## Exercice N°3

Ecrire la fonction `Bingo` qui utilisera deux nombres, `N` et `d` avec `N` strictement inférieurs à 50 et d strictement inférieurs à 10. La fonction retournera 1 si la $d^{ième}$ décimal de `ln(N)` est égale à `d`, elle retournera -1 sinon. Si le nombre `d` est supérieur à 9 ou si le nombre `N` est supérieur à 50, alors la fonction retournera 0.

Exemple : ln(3) = 1.0986122886 donc `Bingo(3,8)` retournera 1 alors que `Bingo(3,2)` retournera -1. `Bingo(52,8)` ou `Bingo(2,18)` retournera 0

Conseil : rappelez-vous que la fonction `math.trunc` permet de récupérer la partie entière d'un nombre réel.

Ecrire ensuite le programme qui demandera deux nombres à l'utilisateur, appellera `Bingo` avec ces deux nombres et affichera « Gagné » si `Bingo` retourne 1, « Perdu » si `Bingo` retourne -1, « Impossible » si `Bingo` retourne 0.

## Exercice n°4

Ecrire un programme qui demande deux caractères à l'utilisateur et l'informe ensuite s'ils sont dans l'ordre de l'alphabet. Attention les caractères peuvent être en minuscules ou en majuscules !

La table ascii est un tableau de correspondance entre un caractère et une valeur. Pour les lettres de l'alphabet on

| 'A' | 'B' | … | 'Z' | … | 'a' | 'b' | … | 'z' |
|---|---|---|---|---|---|---|---|---|
| 65 | 66 | … | 90 | … | 97 | 98 | … | 122 |

*NB : La fonction ord() permet de récupérer la valeur du code ASCII d'une lettre. On pourra donc construire les différents test en s'aidant de ce tableau et de cette fonction. A noter aussi que l'on peut parfaitement comparer deux caractères (ex : Car1 < Car2 , Car1 > 'Z', Car2 == '4',…)*

- Ecrire une première solution en s'interdisant l'utilisation des conditions composées (ET,OU) mais uniquement des imbrications de si…sinon
- Ecrire une seconde solution utilisant des conditions composées (ET,OU)

## Exercice n°5

Ecrire une fonction qui calcule la durée d'un vol. La fonction prendra 4 arguments d'entrée, l'heure et les minutes au départ et l'heure et les minutes à l'arrivée. La fonction retournera la durée par 2 arguments : le nombre d'heures et le nombre de minutes.

On vérifiera que les données fournies sont valables. Si ce n'est pas le cas un message d'erreur sera affiché et les arguments de retour recevront -1.

On supposera que la durée du vol est toujours inférieure à 24h00 mais que l'arrivée puisse être le lendemain.

## Exercice n°6

Codez une fonction qui prend en argument d'entrée le numéro d'un jour, d'un mois et d'une année à l'utilisateur, et qui retourne s'il s'agit ou non d'une date valide (1 si la date est valide, 0 sinon).

Il n'est sans doute pas inutile de rappeler rapidement que le mois de février compte 28 jours, sauf si l'année est bissextile, auquel cas il en compte 29. L'année est bissextile si elle est divisible par quatre. Toutefois, les années divisibles par 100 ne sont pas bissextiles, mais les années divisibles par 400 le sont.

*NB: en python l'opérateur // retourne le quotient de la division entière et l'opérateur % retourne le reste.*

## Exercice n°7 : fonctions discontinues.

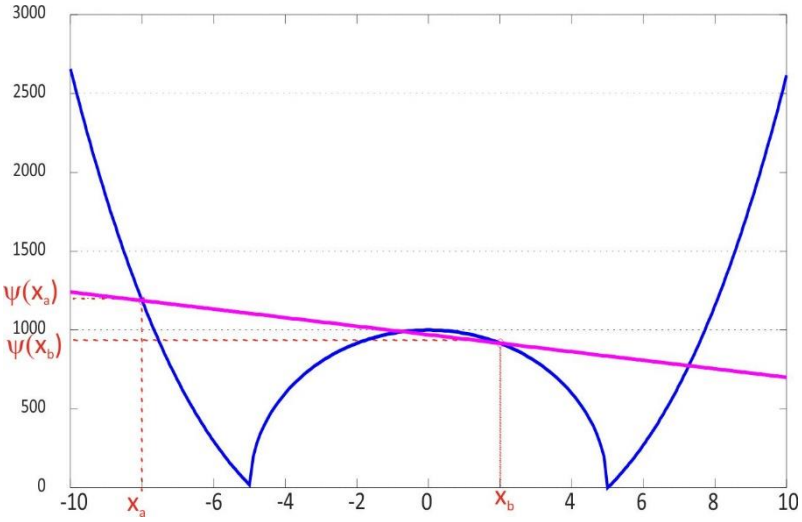
On considère la fonction $\Psi$ discontinues décrite par le jeu d'équations suivant :

$$\begin{cases} y = -3x^3 - 2x - 365 & si \ \ x \in ]-\infty, -5] \\ y = 200 * \sqrt{25 - x^2} & si \ \ x \in ]-5 \ \ 5[ \\ y = 3x^3 - 2x - 365 & si \ x \in [5 \ \ \infty[ \end{cases}$$

Le tracé en bleu dans la figure ci à-côté est la représentation graphique de cette fonction sur l'intervalle [-10 10].

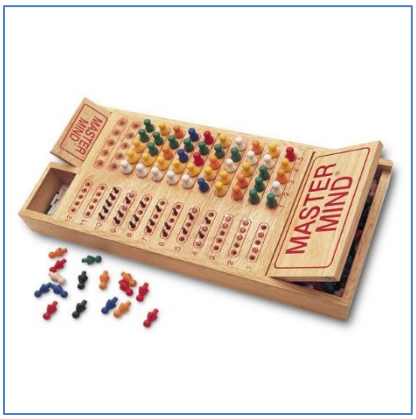
Ecrire ma fonction **`Discon`** qui à partir de x calcule la valeur y.

Ecrire un script (**`DroitaDiscon`**) qui demande 2 valeurs d'abscisse ($x_a \ et \ x_b$) à l'utilisateur et qui calcule les valeurs des coefficients *a* et *b* de la droite (en rouge sur le tracé) $y = ax + b$ qui relient les points $(x_a, \Psi(x_a))$ et $(x_b, \Psi(x_b))$.




Lors de la saisie de ces deux points, le programme vérifiera que $x_a$ est toujours strictement inférieur à $x_b$. Si ce n'est pas le cas, **le calcul ne se fera pas** et un message d'erreur sera dispensé.

## Exercice n°8 : Mastermind light




On souhaite faire une version légère du *mastermind*. Dans cette version du jeu, le joueur doit deviner un nombre compris entre 1111 et 4444. L'ordinateur tire au hasard le nombre à découvrir. Le joueur dispose ensuite de 5 tours pour trouver la combinaison. A chaque tour il donne une valeur que l'ordinateur analyse. Les informations que retourne l'ordinateur sont :

- le nombre de chiffres qui sont corrects, quelle que soit leur place.
- le nombre de chiffres qui sont placés.

*Exemple : Le nombre à découvrir est 1234, si le joueur donne 1321 la réponse de l'ordinateur sera 3 chiffres corrects, 1 chiffre correctement placé.*

Ecrire la fonction `Zazard` qui ne prend aucun argument d'entrée et qui retourne un nombre dont tous les chiffres sont compris entre 1 et 4. *On utilisera la fonction* `random()` *du module* `random` *qui retourne un nombre réel* $x \in [0,1[$. *Ainsi* `3*random.random()` *retourne un nombre réel* $x \in [0,3[$.

Ecrire la fonction `Compare` qui prendra 2 arguments d'entrée (deux nombres X et Y) et qui retournera deux arguments de sortie `ident` et `place`. L'argument de sortie `ident` sera le nombre de chiffres identiques et l'argument de sortie `place` sera le nombre de chiffres placés.

*La façon la plus simple de coder cette fonction consiste à isoler dans des variables l'ensemble des chiffres (8 au total) composant ces 2 nombres. On pourra également utiliser la somme de comparaison. Par exemple val = ((T == 2) + (T < 5) + (T > 0 ) ) val recevra 3 si T vaut 2, 2 si T vaut 1 , 0 si T=-1,….*

Ecrire le script `Master.py` qui dans un premier temps appellera la fonction `Zazard` puis qui appellera cinq fois de suite au maximum la fonction `Comp`. Le script `Master` gérera les affichages envoyés à l'utilisateur. A la fin du jeu, on affichera si le joueur a gagné (et en combien de coup) ou perdu.

> Si le joueur saisit une valeur aberrante (par exemple 114 ou 1477) aucune analyse ne sera lancée mais cela comptera comme un tour pour le joueur.

## Exercice n°9 : j'assure pour l'exam

Ecrire un script permettant de saisir les données nécessaires (sans contrôle de saisie) et de traiter le problème ci-après (avant de se lancer à corps perdu dans cet exercice, on pourra réfléchir un peu et s'apercevoir qu'il est plus simple qu'il n'en a l'air … c'est à dire faire une analyse !)

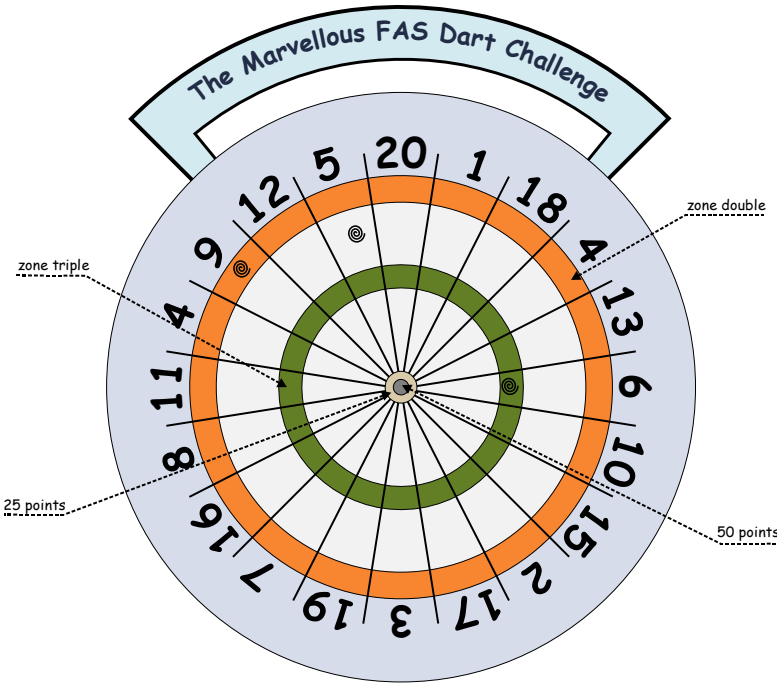
Une compagnie d'assurance automobile propose à ses clients quatre familles de tarifs identifiables par une couleur, du moins au plus onéreux : tarifs bleu, vert, orange et rouge. Le tarif dépend de la situation du conducteur:

- un conducteur de moins de 25 ans et titulaire du permis depuis moins de deux ans, se voit attribuer le tarif rouge, si toutefois il n'a jamais été responsable d'accident. Sinon, la compagnie refuse de l'assurer.
- un conducteur de moins de 25 ans et titulaire du permis depuis plus de deux ans, ou de plus de 25 ans mais titulaire du permis depuis moins de deux ans a le droit au tarif orange s'il n'a jamais provoqué d'accident, au tarif rouge pour un accident, sinon il est refusé.
- un conducteur de plus de 25 ans titulaire du permis depuis plus de deux ans bénéficie du tarif vert s'il n'est à l'origine d'aucun accident et du tarif orange pour un accident, du tarif rouge pour deux accidents, et refusé au-delà

- De plus, pour encourager la fidélité des clients acceptés, la compagnie propose un contrat de la couleur immédiatement la plus avantageuse s'il est entré dans la maison depuis plus d'un an.

## Exercice n°10 : Je suis une flèche…

Dans le jeu de fléchette simple chaque joueur lance 3 fléchettes (on appelle cela une volée) à chaque tour.




Chaque fléchette arrivée dans la partie intérieure de la cible (ci à-côté) marque des points, avec les différent cas suivant :

- dans le petit disque central : 50 points
- dans le grand disque central : 25 points
- dans le reste de la cible, les points correspondent aux points attribués (entre 1 et 20) au secteur atteint. Les points sont doublés si la fléchette est dans le grand anneau “zone double (orange)” qui délimite la zone de jeu.
  Les points sont triplés si la fléchette est dans le petit anneau “zone triple (vert)”
- Une fléchette qui arrive au-delà de l’anneau double (donc la zone grise) ne marque aucun point.

Les points obtenus lors d’une volée est simplement la somme des points acquis par chacune des 3 fléchettes. Dans ce jeu simple on suppose que 2 joueurs s’affrontent. La partie se joue en 5 volées et le vainqueur est celui qui totalise le plus de points sur les 5 lancés.

Le script général (**FlechetteSimple.py**) se construira à partir des lignes suivantes :

```
Joueur1 = input('Quel est le nom du premier jouer : ')
Joueur2 = input('Quel est le nom du second jouer : ')
#Première volée
…
# Deuxième volée
…
# Troisième volée
…
# Quatrième volée
…
# Cinquième volée
…
# Affichage du vainqueur
print('Le vainqueur est donc :')
…
print('Il obtient le mirifique score de …')
print('Alors que son adversaire n''a obtenu que…. ')
```

A vous de proposer une structuration avec une pour plusieurs fonctions pour rendre ce programme lisible et efficace.