

TP SUR LES FONCTIONS

Exercice 1 :

1. Sans taper les programmes suivants, indiquer quel sera le résultat affiché.

a)

```
def fonction1(prenom):  
    reponse="Je m'appelle "+prenom+" :"  
    return reponse
```

```
print(fonction1("Pierre"))
```

Je m'appelle Pierre

c)

```
def fonction3(i,j):  
    difference=i-j  
    return difference
```

```
print(fonction3(4))
```

un ton

b)

```
def fonction2(prenom):  
    return "hello world !"  
    reponse="Je m'appelle "+prenom+"."  
    return reponse
```

```
print(fonction2("Pierre"))
```

hello world !

2. Faites vérifier vos résultats par votre professeur.

Exercice 2 :

Ecrire une fonction *prixTTC* ayant comme paramètres un prix HT et un taux de TVA et qui renvoie le prix TTC. Tester cette fonction.

On rappelle que $\text{prix TTC} = \text{prix HT} * (1 + \text{taux}/100)$

Exercice 3 :

Un étudiant se voit attribuer deux notes *a* et *b* qui sont des nombres réels. S'il est en option informatique, sa note finale sera $\frac{a + 2 \times b}{3}$, sinon sa note finale sera $\frac{a + b}{2}$.

1. Ecrire une fonction *note_finale* qui calcule et retourne la note finale en prenant en paramètres les deux notes *a* et *b* et l'option. Tester cette fonction.
2. Ecrire une fonction *affichage_resultat* qui a comme paramètres les deux notes *a* et *b* et l'option et affiche « reçu » ou « collé » suivant que le résultat est supérieur ou égal à 10 ou non. Cette fonction utilisera la fonction précédente. Tester.

Exercice 4 :

1. Ecrire une fonction *estPremier* qui a comme paramètre un entier naturel *n* strictement positif et qui renvoie un booléen indiquant si *n* est premier.

Indications :

Un nombre est premier s'il a exactement deux diviseurs distincts (par exemple, 2 est premier, mais pas 4 ; 1 n'est pas premier car il a un seul diviseur).

L'idée est donc de tester tous les entiers de 1 à *n* et de vérifier si ce sont des diviseurs de *n* (on rappelle que *k* est un diviseur de *n* si le reste de la division euclidienne de *n* par *k* est nul ; en Python cela se traduit par *if n%k==0*). Lorsqu'on trouve un diviseur, alors on incrémente un compteur. A la fin, si le compteur vaut 2, alors *n* est premier, sinon, il n'est pas premier.

2. Ecrire une fonction *crible* qui prend comme paramètre un entier *M* strictement positif et qui affiche tous les nombres premiers inférieurs ou égaux à *M*. Elle devra utiliser la fonction *estPremier*. La tester avec 100, puis 200 (il y a 25 nombres premiers inférieurs ou égaux à 100).

Exercice 5 :

Dans cet exercice, on veut programmer le jeu « Pierre, feuille, ciseaux ».

Principe du jeu :

Chaque joueur a trois choix possibles : « Pierre », « feuille » ou « ciseaux ».

A chaque partie, les joueurs choisissent simultanément un des trois coups possibles en le symbolisant avec la main.

Le vainqueur (lorsqu'il y en a un) est déterminé par les règles suivantes :

- la pierre bat les ciseaux ;
- les ciseaux battent la feuille ;
- la feuille bat la pierre.

Programmation :

1. Ecrire une fonction *affichage* qui prend comme paramètres une variable *nom* et une variable *choix*.

Cette fonction devra afficher « Vous avez choisi : » si la variable *nom* contient "j", sinon elle devra afficher « L'ordinateur a choisi : ».

Elle devra ensuite afficher le choix : « Pierre », « feuille » ou « ciseaux ».

La tester avec différentes valeurs. Vous devriez obtenir un affichage du type :

```
Vous avez choisi :      L'ordinateur a choisi :
pierre                 . . . ou feuille
```

2. Ecrire une fonction *unePartie* ne prenant aucun paramètre. Cette fonction devra demander au joueur son choix, faire appel à la fonction *affichage* afin d'afficher le choix du joueur, générer un entier aléatoire compris entre 1 et 3 simulant le choix de l'ordinateur, afficher le choix de l'ordinateur puis afficher s'il y a égalité ou le nom du vainqueur.


```
Que choisissez-vous ? 1 pour pierre, 2
pour feuille, 3 pour ciseaux.1
Vous avez choisi :
pierre
L'ordinateur a choisi :
feuille
L'ordinateur a gagné la partie !
```

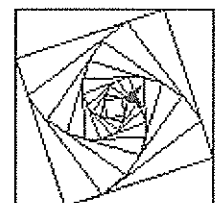
3. Ecrire une fonction *plusieursParties* ne prenant aucun paramètre. Cette fonction devra faire appel à la fonction *unePartie* et demander au joueur s'il souhaite rejouer tant que le joueur a répondu « oui » à la dernière question.
4. On veut améliorer la dernière fonction en rajoutant le score du joueur et de l'ordinateur.
 - a) Faites en sorte que la fonction *unePartie* retourne "j" si c'est le joueur qui gagne, "o", si c'est l'ordinateur qui gagne et "e" s'il y a égalité.
 - b) Modifier la fonction *plusieursParties* de façon qu'elle calcule et affiche les scores.

Exercice 6 :

L'objectif de cet exercice est de tracer la figure ci-contre.

1. A l'aide du module turtle, écrire une fonction *carré* qui prend en paramètre la longueur du côté du carré, la couleur du trait, l'angle selon lequel la tortue doit tracer le carré et qui trace le carré correspondant (on utilisera une boucle car on répète 4 fois les mêmes

instructions). Par exemple, *carré(50,"red",20)* affichera .



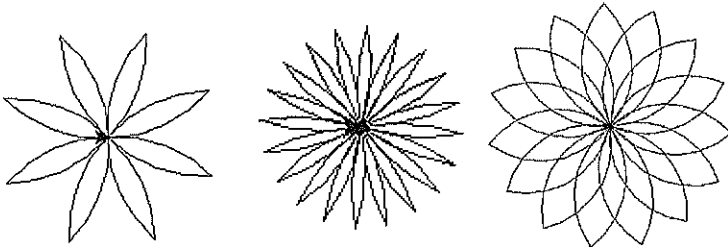
2. Ecrire une fonction *figure* permettant de réaliser la figure attendue. Cette fonction prendra en paramètre la longueur du côté du carré extérieur et la couleur du trait et elle utilisera la fonction *carré*.

Indications :

- La tortue doit tracer un carré, puis on la déplace et elle trace le 2^e carré, puis on la déplace et elle trace le 3^e carré... Les carrés seront tracés tant que le côté du carré est supérieur à une valeur choisie (par exemple 10).
- Dans chaque triangle rectangle, la longueur du petit côté vaut 1/4 de la longueur du côté du carré. La longueur du deuxième côté de l'angle droit vaut donc 3/4 de la longueur du côté du carré et la longueur de l'hypoténuse vaut $\sqrt{\frac{5}{8}}$ de la longueur du côté du carré (d'après le théorème de Pythagore).
- La mesure de l'angle aigu le plus petit est $\arctan(\frac{1}{3})$ soit environ 18,435° (d'après la trigonométrie).
- Pour pouvoir utiliser la racine carrée en Python, il faut importer le module math (from math import*), puis utiliser la fonction sqrt (sqrt(x) renvoie la racine carrée de x).

Exercice 7 :

L'objectif de cet exercice est de tracer des fleurs comme ci-dessous à l'aide du module turtle :



1. La fonction **circle(rayon,angle)** permet de tracer un arc de cercle de rayon donné, la longueur de l'arc de cercle correspondant à la valeur de l'angle donné en degré. Par exemple, `circle(100,180)` trace un demi-cercle de rayon 100. Tester cette fonction avec différentes valeurs pour l'angle afin de bien comprendre son utilisation.
2. Ecrire une fonction *petale* prenant en paramètre un rayon et un angle et qui dessine un pétale de la fleur.

Indications :

La tortue doit répéter deux fois les deux instructions suivantes :

- tracer un arc de cercle de rayon et d'angle donnés en paramètres ;
 - tourner à gauche de $(180 - \text{angle})$ degrés..
3. Ecrire une fonction *fleur* prenant en paramètres un nombre de pétales, un rayon et un angle. Cette fonction devra utiliser la fonction *petale* pour tracer une fleur avec le nombre de pétales donné en paramètre. La tortue tournera de $360/\text{nbPetales}$ entre chaque tracé de pétale.

Exercice 8 :

On définit la suite de Syracuse d'un entier N donné, comme étant la suite (u_n) définie par récurrence par : $u_0 = N$ et

$$\text{pour tout entier } n \geq 0, u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$$

Une célèbre conjecture, appelée conjecture de Syracuse, affirme que quelque soit l'entier naturel N , la suite de Syracuse de N finit toujours par atteindre 1 (autrement dit, il existe toujours un entier n tel que $u_n = 1$). Cette conjecture a été vérifiée à l'aide d'ordinateurs pour toutes les valeurs possibles de N comprises entre 1 et $2^{62} - 1$. Cependant, aucune démonstration mathématique n'existe encore à l'heure actuelle.

1. Ecrire une fonction *syracuse(N,n)* qui affiche les termes de u_0 à u_n de la suite de Syracuse de N .
Par exemple avec $N = 6$ et $n = 10$, vous obtiendrez $6 - 3 - 10 - 5 - 16 - 8 - 4 - 2 - 1 - 4 - 2$
Avec $N = 5$ et $n = 8$, vous obtiendrez $5 - 16 - 8 - 4 - 2 - 1 - 4 - 2 - 1$

Indication : u est pair ssi le reste de la division euclidienne de u par 2 vaut 0 (utiliser l'opérateur %)

2. Le temps de vol de la suite de Syracuse est le plus petit indice n tel que $u_n = 1$.
Ecrire une fonction *vol(N)* qui retourne le temps de vol de la suite de Syracuse de N .
Par exemple, le temps de vol si $N = 15$ est 17 et si $N = 127$, il est de 46.

