
Récurtivité

Exercices

Exercice 1

Étant données les fonctions Python suivantes.

def f(x) :

```
    z = x * x
    return z - x
```

def g(y) :

```
    x = y + 1
    t = f(x)
    return x + y + t
```

Décrire la configuration de la pile pour l'appel à g(4), au moment (A) ou l'appel interne f(x) s'apprête à renvoyer sa valeur et (B) où la fonction g s'apprête à renvoyer son résultat.

(A)

reg	...	
ret		
y	4	
x	5	g(4)
t		
reg	...	
ret	?	?(?)
x	?	
?	?	

(B)

reg	...	
ret	?	
y	?	
x	?	?(?)
t	?	

Exercice 2

Implémenter ces fonctions de manière récursive.

```
def compte_a_rebours(n: int) -> None:
    """Affiche n, n - 1, n - 2, ..., 2, 1, Partez !"""

def n_entiers(n: int) -> None:
    """Affiche les n premiers entiers naturels de 0 à n - 1"""
```

Exercice 3

Implémenter ces fonctions de manière récursive.

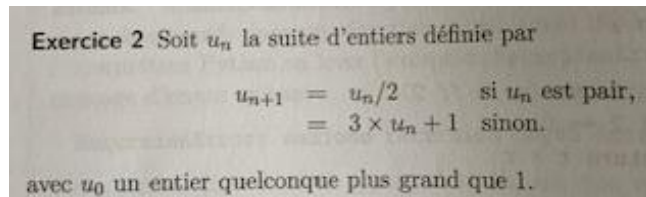
a)

```
# Triangle croissant
def triangle_croissant(taille):
    """
    triangle_croissant(4) renvoie :
    #
    ##
    ###
    ####
    """
    pass
```

b)

```
# Triangle décroissant
def triangle_decroissant(taille):
    """
    triangle_decroissant(4) renvoie :
    ####
    ###
    ##
    #
    """
    pass
```

Exercice 4



Écrire une fonction récursive Syracuse(u_n) qui affiche les valeurs successives de la suite U_n tant que U_n est plus grand que 1.

Exercice 5

On considère la suite suivante définie par la relation de récurrence suivante, où a et b sont des réels quelconques :

$$u_n = \begin{cases} a \in \mathbb{R} & \text{si } n = 0 \\ b \in \mathbb{R} & \text{si } n = 1 \\ 3u_{n-1} + 2u_{n-2} + 5 & \forall n \geq 2 \end{cases}$$

Écrire une fonction récursive serie(n, a, b) qui renvoie le n -ème terme de cette suite par des valeurs a et b données en paramètres.

Exercice 6

Écrire une fonction récursive pgcd(a, b) qui renvoie le PGCD de deux entiers a et b .

En arithmétique élémentaire, le **plus grand commun diviseur** ou **PGCD de deux nombres entiers** non nuls est le plus grand entier qui les **divise** tous les deux. Par exemple, le PGCD de 20 et de 30 est 10, puisque leurs **diviseurs** communs sont 1, 2, 5 et 10.

Exercice 7

```
def somme(tableau: list) -> int:
    """Calculer la somme des valeurs d'un tableau"""
```

Exercice 8

```
# Maximum d'une liste
def maximum(tableau):
    "Déterminer la valeur maximale dans un tableau"
    pass
```

```
# Maximum d'une liste
def maximum(tableau):
    "Déterminer la valeur maximale dans un tableau"
    if len(tableau) == ... :
        return tableau[0]
    else:
        premier = ...
        maxi_fin = ...
        if premier > ...:
            return premier
        else:
            ...
```

Exercice 9

```
# Convertir un nombre en str vers le int correspondant
def entier(chaine):
    "Convertit un nombre en chaîne de caractères en entier"
    pass
```

Exercice 10

Écrire une fonction récursive appartient(v, t, i) prenant en paramètre une valeur v, un tableau t et un entier i et renvoyant True si v apparaît dans t entre l'indice i (inclus) et len(t) (exclu), et False sinon. On supposera que i est toujours compris entre 0 et len(t).

Exercice 11

```
def nombre_chiffres(n: int) -> int:
    """
    Renvoie le nombre de chiffres nécessaires à l'écriture de n en base 10
    n est un entier positif
    On rappelle que  $536 // 10 = 53$ 
    nombres_chiffres(509) -> 3
    """

def somme_chiffres(n: int) -> int:
    """
    Renvoie la somme des chiffres composant n
    somme_chiffres(509) -> 14
    """

def listes_imbriquees(n: int) -> list:
    """
    Renvoie n listes imbriquées
    listes_imbriquees(1) -> []
    listes_imbriquees(2) -> [[]]
    listes_imbriquees(3) -> [[[]]]
    """

def profondeur(liste: list) -> int:
    """
    liste est une liste telle que renvoyée par la fonction listes_imbriquées
    La fonction profondeur renvoie le nombre de listes utilisées
    profondeur([]) -> 1
    profondeur([[]]) -> 2
    profondeur([[[]]]) -> 3
    """
```

Exercice 12

Les palindromes

On appelle palindrome un mot qui se lit dans les deux sens comme « été » ou « radar ».
Écrire une fonction récursive palindrome qui teste si un mot est un palindrome.

- Entrée : Un mot (type str).
- Sortie : Un booléen égal à True si le mot est un palindrome, False sinon.

Principe

Cas de base :

- Si le mot est la chaîne vide, c'est un palindrome ;
- Si le mot ne contient qu'une seule lettre, c'est un palindrome.

Dans les autres cas :

- le mot est un palindrome si et seulement si la première et la dernière lettre sont égales et le mot tronqué de ses premières et dernières lettres est un palindrome.