

## Fonctions récursives – Activités préparatoires.

Les activités préparatoires sont disponibles aux liens suivants : sujet – <https://bit.ly/3zkiJgb> et corrigé – <https://bit.ly/3zl7QKJ>.

### QCM

**Question 1** Que retourne la commande suivante `mystere(4)` ?

```
def mystere(n):
    if n>0 :
        return mystere(n-2)
    else :
        return n==0
```

- 1°) 0.
- 2°) False.
- 3°) True.
- 4°) L'exécution génère une erreur.

**Question 2** Laquelle de ces fonctions retourne True lorsqu'on exécute `f(5)` ?

```
def f1(n):
    if n==0 :
        return True
    else :
        return f(n-2)

def f2(n):
    if n<=0 :
        return True
    else :
        f(n-2)

def f3(n):
    if n<=0 :
        return True
    return f(n-2)

def f4(n):
    if n==0 :
        return True
    f(n-2)
```

- 1°) f1.
- 2°) f2.
- 3°) f3.

4°) f4.

**Question 3** Quel affichage obtient-on en exécutant `affiche(3)` ?

```
def affiche(n):
    print(n)
    if n>=0:
        affiche(n-1)
```

- 1°) 3, 2, 1, 0 (avec des retours à la ligne entre chaque valeurs).
- 2°) 0, 1, 2, 3 (avec des retours à la ligne entre chaque valeurs).
- 3°) 3, 2, 1, 0, -1 (avec des retours à la ligne entre chaque valeurs).
- 4°) 3.

**Question 4** Une seule des fonctions définies ci-dessous retourne 'cccc' à l'appel de `replique(5, 'c')`. Déterminer laquelle.

```
def replique(a,b): # Fonction 1
    if a==1:
        return b
    else :
        return replique( a-1 , b+b)

def replique(a,b): # Fonction 2
    if a==1:
        return b
    elif a%2 == 0:
        return replique( a-2 , b+b)
    else :
        return b + replique( a-2 , b+b)

def replique(a,b): # Fonction 3
    if a==1:
        return b
    elif a%2 == 0:
        return replique( a//2 , b+b)
    else :
        return b + replique( a//2 , b+b)
```

```
def replique(a,b): # Fonction 4
    if a==1:
        return b
    else :
        replique( a-1 , b+b)
```

- 1°) Fonction 1.
- 2°) Fonction 2.
- 3°) Fonction 3.
- 4°) Fonction 4.

**Question 5** Que retourne l'instruction `copy(3, 'A')` ?

```
def copy(n,s):
    if n==0:
        return s
    return copy(n-1, s+s)
```

- 1°) 'AAA'.
- 2°) 'AAAAAA'.
- 3°) 'AAAAAAAA'.
- 4°) '3A'.

**Question 6** Que retourne l'instruction : `mystere(3, '$')` ?

```
def mystere(n,s):
    if n==0:
        return s
    return s + mystere(n-1, s)
```

- 1°) '\$\$\$'.
- 2°) '\$2\$'.
- 3°) '\$\$\$\$'.
- 4°) L'exécution déclenche une erreur.

**Question 7** Que retourne la commande `f(3,4)` ?

```
def f(a,b):
    if a == 0 :
        return b
    return f(a-1, b+1)
```

- 1°) 4.
- 2°) 5.
- 3°) 6.
- 4°) 7.

**Question 8** Que retourne la commande `mystere(3)` ?

```
def mystere(n):
    if n>0 :
        return mystere(n-2)
    else :
        return n==0
```

- 1°) True.
- 2°) False.
- 3°) RecursionError.
- 4°) 0.

**Question 9** On propose de créer une fonction récursive permettant de calculer  $x^n$ . Compléter la fonction proposée.

```
def puissance(x,n):
    if n > 0 :
        return .....
    return 1
```

- 1°) `puissance(x,n-1)`.
- 2°) `x*puissance(x,n-1)`.
- 3°) Quoique l'on écrive, cette fonction ne donnera pas le résultat attendu.
- 4°) `x**(n-1)*puissance(x,n-1)`.

**Question 10** Que renvoi ce programme en console ?

```
def ed(L,M=[]):
    if len(L) == 0 : return M
    a=L.pop()
    if a not in M : M.append(a)
    return ed(L,M)
L=[2, 3, 2, 6, 8, 9, 9, 10, 9, 3, 6, 7, 8, 8, 9]
print(ed(L))
```

- 1°) None.
- 2°) [9, 8, 7, 6, 3, 10, 2].
- 3°) [9, 8, 8, 7, 6, 3, 9, 10, 9, 9, 8, 6, 2, 3, 2].
- 4°) [2, 10, 3, 6, 7, 8, 9].

**Question 11** Que retourne le programme suivant ?

```
def A(x):
    if x <= 1 : return x
    return B(x+1)

def B(x) :
    return A(x-2)+4

print(A(4))
```

- 1°) 13.
- 2°) 1.
- 3°) 12.
- 4°) Une erreur de type : "RecursionError : maximum recursion depth exceeded in comparison".