


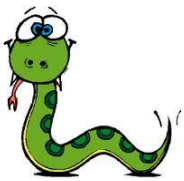


Nom :	Note :
Une seule réponse juste par question sauf signalé. Un point par réponse juste. Pas de point négatif.	Classe :

Récursivité Question de cours

	Q1	Réponse
On souhaite réaliser une implémentation de la fonction factorielle en utilisant la programmation récursive. Cocher la forme correcte : $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$		A B C D

A
$$n\,! = \begin{cases} 1 & \text{si } n = 0 \\ (n+1)\cdot(n-1)\,! & \text{sinon} \end{cases}$$

B
$$n\,! = \begin{cases} 1 & \text{si } n = 0 \\ (n-1)\,! & \text{sinon} \end{cases}$$

C
$$n\,! = \begin{cases} 1 & \text{si } n = 0 \\ n\cdot(n+1)\,! & \text{sinon} \end{cases}$$

D
$$n\,! = \begin{cases} 1 & \text{si } n = 0 \\ n\cdot(n-1)\,! & \text{sinon} \end{cases}$$

	Q2	Réponse
Que signifie programmation dynamique : A Le programme s'adapte à toute situation B N'est possible qu'avec Python C Le programme est optimisé en temps et en mémoire D Ne veut rien dire		A B C D

	Q3	Réponse
On souhaite implémenter une suite arithmétique $S_n = 1 + 2 + 3 \ldots\ldots + n$ en programmation récursive. Que vaut le cas de base ? A : 1 si n ≠ 1 B : 0 si n = 1 C : 0 si n = 0 D : 1 si n ≠ 0		A B C D

	Q4	Réponse
On souhaite implémenter une suite arithmétique $S_n = 1 + 2 + 3 \ldots\ldots + n$ en programmation récursive. Que vaut le cas récursif ? A : S_{n-1} si n≠ 1 B : $n+S_{n-1}$ si n ≠ 0 C : $n+S_{n-1}$ si n = 1 D : S_{n-1} si n = 1		A B C D

	Q5	Réponse

Avec la fonction suivante comment est évalué f(8)
 	
A : Le calcul provoque une erreur
B : 14 C : 256 D : 2
	`def f(n):`
	`  if n == 0:`
	`  return 1`
	`  else:`
	`  return 2 * f(n-1)`
		A
		B
		C
		D

! Deux réponses correctes pour cette question	Q6	Réponse
Diviser pour régner est un paradigme de programmation : A : principalement utilisé avec un style de programmation itératif B : principalement utilisé avec un style de programmation récursif C : consiste à diviser un problème en sous-problèmes dépendants les uns des autres D : consiste à diviser un problème en sous-problèmes indépendants les uns des autres		A B C D

	Q7	Réponse
Une complexité décrite par une relation de récurrence de la forme : C(n) = C(n-1) + n et C(1) = 1 est en : A : **Q**(n) B : **Q**(2^n) C : **Q**(n^2) D : **Q**(nlogn)		A B C D

	Q8	Réponse
La multiplication dite du 'paysans russe' ne nécessite pas la connaissance des tables de multiplication, seulement l'addition et la division par deux. Deux entiers sont multipliés de la manière suivante : on divise a par 2 tant que c'est possible, en doublant b, sinon on décrémente a et on ajoute b au résultat. Quel est le code correct ?		A B C D

A

```
def multi(a, b):
    if a == 0:
        return 0
    if a % 2 == 0:
        return multi( a//2, b)
    return b + multi(a-1, b)
```

B

```
def multi(a, b):
    if a == 0:
        return 0
    if a % 2 == 0:
        return multi( a//2, b*2)
    return b + multi(a-1, b)
```

C

```
def multi(a, b):
    if a == 0:
        return 0
    if a % 2 == 0:
        return multi( a//2, b*2)
    return b + multi(a, b)
```

D

```
def multi(a, b):
    if a == 0:
        return 0
    if a % 2 == 0:
        return multi( a//2, b*2)
    return b + multi(a-1, b-a)
```

	Q9	Réponse
		A
		B
		C
		D

La suite de Syracuse est une suite qui est définie de la manière suivante : $u_0 = N$ puis $u_{n+1} = u_n/2$ si u_n est pair et $u_{n+1} = 3 \cdot u_n + 1$ si u_n est impair.
On donne N=10 quelle est la suite obtenue ?

A : 10, 5, 15, 8, 4, 2, 1 B : 10, 5, 16, 8, 3, 2, 1
C : 10, 5, 16, 8, 4, 2, 1 D : 10, 5, 16, 8, 4, 3, 5, 1

	Q10	Réponse
		A
		B
		C
		D

Un algorithme récursif de calcul du pgcd de deux entiers a et b est rappelé ci-dessous :
Quel est le code correct ?

```
fonction pgcd(a,b)
si b=0 alors
    retour a
sinon
    retour pgcd( b, a % b)
```

```
# Code A
def pgcd(a,b):
    if b == 0:
        return a
    else:
        return pgcd(b,a//b)
```

```
# Code B
def pgcd(a,b):
    if b == 0:
        return a
    else:
        return pgcd(b,b%a)
```

```
# Code C
def pgcd(a,b):
    if b == 0:
        return a
    else:
        return pgcd(a,a%b)
```

```
# Code D
def pgcd(a,b):
    if b == 0:
        return a
    else:
        return pgcd(b,a%b)
```

	Q11	Réponse
		A
		B
		C
		D

La suite de Padovan est donnée ci-dessous :

$$P_0 = P_1 = P_2 = 1 \text{ et } \forall n \in \mathbb{N} \quad P_{n+3} = P_{n+1} + P_n.$$

On donne N=9 quelle est la suite obtenue ?

A : 1, 1, 1, 2, 3, 4, 6, 8, 10 B : 1, 1, 1, 2, 2, 3, 4, 5, 7
C : 1, 1, 1, 2, 3, 4, 7, 9, 11 D : 1, 1, 1, 2, 2, 3, 5, 7, 11

	Q12	Réponse
		A
		B
		C
		D

La suite de Padovan est donnée ci-dessous :

$$P_0 = P_1 = P_2 = 1 \text{ et } \forall n \in \mathbb{N} \quad P_{n+3} = P_{n+1} + P_n.$$

Complétez le code ci-contre :

A : padovan(x-3) + padovan(x-2)
B : padovan(x+1) + padovan(x)
C : padovan(x-3) + padovan(x-1)
D : padovan(x+1) + padovan(x-2)

```
def padovan(x):
    if x==0 or x==1 or x== 2:
        return 1
    else:
        return .................
```

	Q13	Réponse
		A
		B
		C
		D

Nous étudions l'algorithme de calcul du pgcd par Euclide (environ 300 ans avant notre ère). L'algorithme de calcul du pgcd de deux entiers a et b est rappelé ci-contre : Quel est le cas de base ?

```
fonction pgcd(a,b)
si b = 0 alors
    return a
fin si
si a > b alors
    a = a - b
    return pgcd(a,b)
sinon
    b = b - a
    return pgcd(b,a)
fin si
```

```
# A
if b == 0:
    return a
```

```
# B
if b == a:
    return 1
```

```
# C
if b == 1:
    return a
```

```
# D
if b != 0:
    return a
```

	Q14	Réponse
		A
		B
		C
		D

Voilà une implémentation récursive proposée pour la recherche d'une valeur v dans un tableau t.
Cette valeur devant se trouvée entre les indices i inclus et len(v) exclu.
Quelle est l'affirmation correcte ?

```
def appartient(v, t, i):
    if i == len(t):
        return False
    elif t[i] == v:
        return True
    else:
        appartient(v, t, i + 1)
```

A : Le code est correct
B : Le code n'est pas récursif
C : Il manque un return dans le bloc else : `return appartient(v, t, i + 1)`
D : Le code provoque une erreur d'exécution.

	Q15	Réponse
		A
		B
		C
		D

On considère une suite u_n, définie par la relation de récurrence suivante :

$$u_n = \begin{cases} 2 & \text{si } n = 0 \\ 3 & \text{si } n = 1 \\ 3u_{n-1} + 2u_{n-2} + 5 & \text{si } n > 1 \end{cases}$$

Calculer les 6 premières valeurs de la suite

A : 2, 3, 8, 41, 57, 63
B : 2, 3, 12, 37, 126, 475
C : 2, 3, 18, 65, 236, 843
D : 2, 3, 18, 65, 240, 963

Attention plusieurs réponses possibles	Q16	Réponse
		A
		B
		C
		D

On code la suite u_n, définie par la relation de récurrence suivante :

$$u_n = \begin{cases} 2 & \text{si } n = 0 \\ 3 & \text{si } n = 1 \\ 3u_{n-1} + 2u_{n-2} + 5 & \text{si } n > 1 \end{cases}$$

Cocher les codes faux ?

```
# Code A
def suite(n):
    if n == 0:
        return 2
    elif n == 1:
        return 3
    else:
        return 3suite(n-1) + 2suite(n-2) + 5

for _ in range(6):
    print(suite(_), end=" ")
```

```
# Code B
def suite(n):
    if n == 0:
        return 2
    elif n = 1:
        return 3
    else:
        return 3*suite(n-1) + 2*suite(n-2) + 5

for _ in range(6):
    print(suite(_), end=" ")
```

```
# Code C
def suite(n):
    if n == 0:
        return 3
    elif n == 1:
        return 2
    else:
        return 3*suite(n-1) + 2*suite(n-2) + 5

for _ in range(6):
    print(suite(_), end=" ")
```

```
# Code D
def suite(n):
    if n == 0:
        return 2
    elif n == 1:
        return 3
    else:
        return 3*suite(n-1) + 2*suite(n-2) + 5

for _ in range(6):
    print(suite(_), end=" ")
```

! Deux réponses correctes pour cette question	Q17	Réponse
Cocher les affirmations exactes concernant la programmation récursive. Dans l'approche récursive …		A B C D

A : Une fonction récursive ne peut pas se programmer de façon itérative
B : La pile d'appel des fonctions n'est pas sollicitée
C : Les cas de bases permettent de calculer directement une solution
D : Les cas récursifs utilisent la fonction pour suivre un cheminement vers le résultat

! Deux réponses correctes pour cette question	Q18	Réponse
Cocher les affirmations exactes concernant la programmation dynamique. La programmation dynamique …		A B C D

A : augmente la quantité de mémoire vive nécessaire.
B : consiste à ne pas calculer plus d'une fois le même calcul.
C : doit toujours être gérée explicitement par le programmeur.
D : la mémoïsation en fait partie

! Trois réponses correctes pour cette question	Q19	Réponse
La programmation récursive et Python. Cocher les affirmations exactes.		A B C D

A : La limite de récursion est fixée par défaut à 1000 appels
B : La limite de récursion ne peut pas être modifiée sous Python.
C : Il existe des langages plus spécialisés dans l'écriture de fonctions récursives que Python.
D : la bibliothèque **functools** de Python contient une fonction **lru_cache** qui réalise automatiquement la mémoïsation. (Techniquement appelé un décorateur)

	Q20	Réponse
Cocher l'affirmation fausse :		A B C D

A : La fonction récursive peut être remplacée par une fonction non récursive
B : Les fonctions récursives sont plus rapides que les fonctions non récursives
C : Les fonctions récursives prennent généralement plus de mémoire que les fonctions non récursives
D : La récursivité rend les programmes plus faciles à comprendre