LES SUITES NUMÉRIQUES M02

EXERCICE N°1 Suite et relation de récurrence : 1er contact

VOIR LE CORRIGÉ

On donne la suite u définie par : $\begin{cases} u_0 = -2 \\ \forall n \in \mathbb{N} \text{ , } u_{n+1} = 3u_n+1 \end{cases}$.

1) Identifier la fonction f du cours.

2) Déterminer, *si possible*, u_1 , u_2 , u_8 et u_{1000} .

EXERCICE N°2 Suite et relation de récurrence : 2ème contact

VOIR LE CORRIGÉ

On donne la suite v définie par : $\begin{cases} v_0 = 2 \\ \forall n \in \mathbb{N} \text{ , } v_{n+1} = {v_n}^2-3 \end{cases}$.

1) Identifier la fonction f du cours.

2) Déterminer v_1 , v_2 et v_{15} .

EXERCICE N°3 Suite définie par un algorithme (Python)

VOIR LE CORRIGÉ

On donne la suite $(w_n)_{n \in \mathbb{N}}$ définie par : $w_0 = 2$
Pour un terme w_n ,

w_{n+1} s'obtient de la façon suivante :
- Multiplier w_n par lui même.
- Enlever 3 au résultat.

1) Écrire une fonction «premiers_termes_de_w» en Python qui prend comme argument un entier n et qui renvoie une liste contenant les valeurs des $n+1$ premiers termes de la suite.

2) Écrire une fonction « w » en Python qui prend comme argument un entier n et qui renvoie la valeur de w_{n+1} . (*On pourra utiliser la question 1*)

EXERCICE N°4 Triangle de Sierpinski (un peu de culture)

Juste une petite vidéo pour votre culture personnelle
https://www.youtube.com/shorts/kMA8bdkErII

EXERCICE N°1 ***Relation de récurrence : premier contact (Le corrigé)*** RETOUR À L'EXERCICE

On donne la suite u définie pour par : . $\begin{cases} u_0 = -2 \\ \forall n \in \mathbb{N} , u_{n+1} = 3u_n+1 \end{cases}$

1) Identifier la fonction f du cours.

$f : x \mapsto 3x+1$

2) Déterminer, *si possible*, . u_1 , u_2 , u_8 et u_{1000}

▪ $u_1 = 3\times u_0+1 = 3\times(-2)+1$

$\boxed{u_1 = -5}$

▪ $u_2 = 3\times u_1+1 = 3\times(-5)+1$

$\boxed{u_2 = -14}$

▪ $u_8 = 3\times u_7+1 = \ldots$ heu ça va faire beaucoup de calculs !

On va utiliser la calculatrice.

▪ À l'aide la calculatrice

$\boxed{u_8 = -9842}$

▪ Il n'est pas (encore) possible (à ce stade du cours) de calculer u_{1000} dans un temps raisonnable.

C'était plus facile quand la suite était définie de manière explicite !

EXERCICE N°2 ***Suite et relation de récurrence : 2ème contact (Le corrigé)*** RETOUR À L'EXERCICE

On donne la suite v définie par : . $\begin{cases} v_0 = 2 \\ \forall n \in \mathbb{N} \ , \ v_{n+1} = {v_n}^2 - 3 \end{cases}$

1) Identifier la fonction f du cours.

$f \ : \ x \mapsto x^2 - 3$

2) Déterminer v_1 , v_2 et v_{15} .

- $v_1 = {v_0}^2 - 3$
$= 2^2 - 3$
$\boxed{v_1 = 1}$

- $v_2 = {v_1}^2 - 3$
$= 1^2 - 3$
$\boxed{v_2 = -2}$

Pour v_{15} on utilise la calculatrice

- À l'aide la calculatrice
$\boxed{v_{15} = 1}$

On aurait pu penser, en voyant le carré, que les valeurs allaient « exploser ».
Les suites définies par des relations de récurrence sont souvent difficiles à appréhender, il faudra faire attention.

EXERCICE N°3 Suite définie par un algorithme (Python) (Le corrigé)

RETOUR À L'EXERCICE

On donne la suite $(w_n)_{n \in \mathbb{N}}$ définie par : $w_0 = 2$
Pour un terme w_n ,
w_{n+1} s'obtient de la façon suivante :
- Multiplier w_n par lui même.
- Enlever 3 au résultat.

1) Écrire une fonction «premiers_termes_de_w» en Python qui prend comme argument un entier n et qui renvoie une liste contenant les valeurs des $n+1$ premiers termes de la suite.

```python
def premiers_termes_de_w(n):
    w = [2] #On déclare une liste avec un seul element w0
    for k in range(n):
        resultat = w[k]*w[k]  #1ere instruction de l'algorithme
        resultat = resultat - 3 #2eme instruction de l'algorithme
        w.append(resultat) # On "ajoute" le résultat à la liste w
    return w    #On renvoie la liste contenant les n+1 premiers termes

def premiers_termes_de_w_bis(n):
    w = [2] #On déclare une liste avec un seul element w0
    for k in range(n):
        w.append(w[k]*w[k]-3) #on ajoute directement à la liste le résultat des deux instructions
    return w    #On renvoie le dernier terme de la liste contenant les n+1 premiers termes

def premiers_termes_de_w_ter(n):
    w = [2] #On déclare une liste avec un seul element w0
    for _ in range(n):   #l'indice n'est pas utilisé donc on préfère "_" plutôt qu'une lettre
        w.append(w[-1] * w[-1] - 3) # w[-1] est le dernier élélment (actuel) de la liste
    return w
```

On donne ici trois réponses possibles, il y a en bien sûr bien d'autres
Les commentaires (après les #) ne sont pas à recopier dans l'éditeur (sauf si vous aimez perdre du temps;))

2) Écrire une fonction « w » en Python qui prend comme argument un entier n et qui renvoie la valeur de w_{n+1} . (*On pourra utiliser la question 1*)

```python
def premiers_termes_de_w(n):
    w = [2] #On déclare une liste avec un seul element w0
    for k in range(n):
        resultat = w[k]*w[k]  #1ere instruction de l'algorithme
        resultat = resultat - 3 #2eme instruction de l'algorithme
        w.append(resultat) # On "ajoute" le résultat à la liste w
    return w[-1]    #On renvoie le dernier terme de la liste contenant les n+1 premiers termes

def premiers_termes_de_w_bis(n):
    w = [2] #On déclare une liste avec un seul element w0
    for k in range(n):
        w.append(w[k]*w[k]-3) #on ajoute directement à la liste le résultat des deux instructions
    return w[-1]    #On renvoie le dernier terme de la liste contenant les n+1 premiers termes

def premiers_termes_de_w_ter(n):
    w = [2] #On déclare une liste avec un seul element w0
    for _ in range(n):   #l'indice n'est pas utilisé donc on préfère "_" plutôt qu'une lettre
        w.append(w[-1] * w[-1] - 3) # w[-1] est le dernier élélment (actuel) de la liste
    return w[-1]
```

On donne ici trois réponses possibles, il y a en bien sûr bien d'autres