

Premiere_Cours_Suite_Partie1

January 30, 2020

1 Première Cours Suites Partie 1

Sur la page <https://repl.it/@fredericjunier/PremiereSuitesPartie1> vous pourrez aussi tester les codes ci-dessous.

1.1 Activité 1 : châteaux de cartes

Pour tout entier naturel $n \geq 1$, on a : * le nombre de cartes au niveau n vérifie $u_n = 3n$ * le nombre total de cartes pour construire n niveaux vérifie $v_{n+1} = u_{n+1} + v_n$

```
In [1]: def chateau(n):  
    u = 0  
    v = 0  
    for k in range(1, n + 1):  
        u = u + 3 #u = k * 3  
        v = v + u  
    return v
```

```
In [2]: [chateau(n) for n in range(0, 8)]
```

```
Out[2]: [0, 3, 9, 18, 30, 45, 63, 84]
```

1.2 Activité 2 Modèle d'évolution d'une population

On a $u_0 = 27500$ étudiants en Septembre 2016.

En notant u_n le nombre d'étudiants en Septembre 2016 + n , en juin 2016 + $n + 1$, après une perte de 150 étudiants, on a $u_n - 150$ étudiants. Puis on a une augmentation de cet effectif, à la rentrée de Septembre, c'est-à-dire $1,04(u_n - 150) = 1,04u_n - 156$ étudiants en Septembre 2016 + $n + 1$.

Pour tout entier $n \geq 0$, on a donc : $u_{n+1} = 1,04u_n - 156$.

La capacité maximale de l'établissement est de 33000. D'après l'algorithme de seuil ci-dessous, à la rentrée de Septembre 2022, la capacité maximale d'accueil sera dépassée.

```
In [7]: def seuil():  
    n = 0  
    u = 27500  
    while u <= 33000:  
        n = n + 1  
        u = 1.04 * u - 156  
    return n
```

1.3 Capacité 2

- Question 1 a) : $a_2 \approx 14400$ et $a_8 \approx 17200$
- Question 1 b) : Le montant de l'APA en 2013 était de $a_7 = 16744$
- Question 2) a) : $a_{10} = a_9 \times 1,05 = 18070 \times 1,05 = 18973,5$
- Question 2) b) : [feuille de calcul en ligne](#)

n	
9	
10	
11	
12	
13	
14	
a(n)	
18070	
18973.5	
19922.175	
20918.28375	
21964.1979375	
23062.407834375	

1.4 Capacité 3

Soit la suite définie par : $u_0 = 4$ et, pour tout entier naturel n , $u_{n+1} = -\frac{1}{2}u_n + 2$.

- Question 1) a) : $u_1 = -\frac{1}{2}u_0 + 2 = 0$ et $u_2 = -\frac{1}{2}0 + 2 = 2$
- Question 1) b) : calcul de u_n avec une fonction Python

```
def suiteU_capacite3_question1(n):  
    u = 4  
    for k in range(n):  
        u = -0.5 * u + 2  
    return u
```

```
In [2]: def suiteU_capacite3_question1(n):  
        u = 4  
        for k in range(n):  
            u = -0.5 * u + 2  
        return u  
  
        #calcul de u(14)  
        print(suiteU_capacite3_question1(14))
```

1.33349609375

Soit la suite définie par $u_0 = 2$ et pour tout entier naturel n , par $u_{n+1} = u_n + n^2 + 1$.

- Question 1) a) : $u_1 = u_0 + 0 = 0$ et $u_2 = -\frac{1}{2} \times 0 + 2 = 2$
- Question 1) b) : calcul de u_n avec une fonction Python

```
def suiteU_capacite3_question2(n):
    u = 2
    for k in range(n):
        u = u + k ** 2 + 1
    return u
```

```
In [3]: def suiteU_capacite3_question2(n):
        u = 2
        for k in range(n):
            u = u + k ** 2 + 1
        return u

        print(suiteU_capacite3_question2(10))
```

297

1.5 Capacité 4

- Question 1) : soit la suite définie pour tout entier $n \geq 1$ par $v_n = \frac{2^n+1}{2+(-1)^n 2^{n+1}}$.

$$\begin{aligned} - v_1 &= \frac{2^1+1}{2+(-1)^1 2^{1+1}} = -\frac{3}{4} \\ - v_2 &= \frac{2^2+1}{2+(-1)^2 2^{2+1}} = \frac{5}{10} \\ - v_3 &= \frac{2^3+1}{2+(-1)^3 2^{3+1}} = -\frac{9}{14} \end{aligned}$$

- Question 2) : soit la suite définie par $u_0 = 0$ et pour tout entier $n \geq 1$, $u_n = u_{n-1} + 2n - 1$

$$\begin{aligned} - u_1 &= u_0 + 2 \times 1 - 1 = 1 \\ - u_2 &= u_1 + 2 \times 2 - 1 = 4 \\ - u_3 &= u_2 + 2 \times 3 - 1 = 9 \end{aligned}$$

Calculs de tous les termes entre u_0 et u_n avec une fonction Python

```
def suiteU_capacite4(n):
    u = 0
    print(u)
    for k in range(1, n + 1):
        u = u + 2 * k - 1
        print(u)
```

On peut conjecturer que pour tout entier $n \geq 0$, on a $u_n = n^2$.

```
In [4]: def suiteU_capacite4(n):
        u = 0
        print(u)
```

```

        for k in range(1, n + 1):
            u = u + 2 * k - 1
            print(u)

suiteU_capacite4(20)

```

```

0
1
4
9
16
25
36
49
64
81
100
121
144
169
196
225
256
289
324
361
400

```

Out[4]: 400

1.6 Capacité 5 : manipuler les listes en Python

```

In [9]: ###Question 1
        L1 = [852, 843, 954]
        print(L1[1])
        #843
        print(L1[0])
        #852
        #print(L[3])
        #provoque une erreur

        ###Question 2
        L2 = [k * 2 - 1 for k in range(3)]

        ###Question 3
        from math import sin

```

```

L3 = []
for k in range(1, 50):
    if sin(k) >= 0:
        L3.append(k)
#équivalent à
L32 = [k for k in range(1, 50) if sin(k) >= 0]
print(L3 == L32)
##True

### Question 4
L4 = list(range(2, 5))
L4.pop()
L4.append(14)
L4.pop(1)
L4.pop(1)
L4.append(16)
print(L4)

```

```

843
852
True
[2, 16]

```

1.7 Suite de syracuse

```

In [1]: def syracuse(u , n):
        """Retourne la liste des premiers termes
        de la suite de syracuse de premier terme u"""
        L = [u]
        for k in range(n):
            if u % 2 == 0:
                u = u // 2
            else:
                u = 3 * u + 1
            L.append(u)
        return L

```

```

In [4]: syracuse(634 , 40)

```

```

Out[4]: [634,
        317,
        952,
        476,
        238,
        119,
        358,
        179,
        538,

```

```

269,
808,
404,
202,
101,
304,
152,
76,
38,
19,
58,
29,
88,
44,
22,
11,
34,
17,
52,
26,
13,
40,
20,
10,
5,
16,
8,
4,
2,
1,
4,
2]

```

```

In [7]: for k in range(10, 21):
        print(syracuse(k, 21))

```

```

[10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4]
[12, 6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4]
[15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4]
[16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2]
[17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1]
[18, 9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4]
[19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4]
[20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2]

```

```
In [8]: def tempsVol(u):
        """Retourne le plus petit indice du terme de la suite de syracuse
        de premier terme u, qui est égal à 1"""
        i = 0
        while u != 1:
            if u % 2 == 0:
                u = u // 2
            else:
                u = 3 * u + 1
            i = i + 1
        return i
```

```
In [9]: tempsVol(634)
```

```
Out[9]: 38
```

1.8 Capacité 6 : Suite définie par des motifs géométriques ou combinatoires

- Question 1) : On a $t_1 = 1$ et pour tout entier $n \geq 2$, on a $t_n = t_{n-1} + n$.
- Question 2) : On admet que pour tout entier $n \geq 1$, on a $t_n + t_{n-1} = n^2$. On en déduit que $t_n + t_n - n = n^2$ c'est-à-dire $t_n = \frac{n(n+1)}{2}$. Notons que pour tout entier $n \geq 1$, $t_n = 1 + 2 + \dots + n$ donc $1 + 2 + \dots + n = \frac{n(n+1)}{2}$.
- Question 3) a) : On a $u_1 = 0$ et pour tout entier $n \geq 1$, on a $u_{n+1} = u_n + n$ ou encore pour tout entier $n \geq 2$, on a $u_n = u_{n-1} + n - 1$.
- Question 3) b) : On peut remarquer que $u_n = 1 + 2 + \dots + n - 1$. D'après la question précédente, on a, pour tout entier $n \geq 1$, $u_n = \frac{(n-1)n}{2}$. On peut aussi noter que $u_1 = 0 = t_1 - 1$ puis $u_2 = u_1 - 1 + 1 = t_1$ puis $u_3 = t_1 + 2 = t_2$ puis $u_4 = t_2 + 3 = t_3$. En terminale, on pourra démontrer par récurrence que pour tout entier $n \geq 2$, on a $u_n = t_{n-1} = \frac{(n-1)n}{2}$.
- Question 4) : On peut assimiler le nombre total de poignées de mains échangées dans une assemblée de n personnes qui se saluent toutes deux à deux, par la valeur de u_n définie dans la question précédente. En effet, on peut considérer que les personnes arrivent successivement dans la salle et que tout nouvel arrivant salue toutes les personnes déjà présentes.

On résout donc l'équation $\frac{(n-1)n}{2} = 45 \iff n^2 - n - 90 = 0 \iff$

$$\begin{cases} n = \frac{1+19}{2} = 10 \\ \text{ou } \frac{1-19}{2} = -9 < 0 \text{ impossible} \end{cases}.$$

1.9 Algorithmique 2 : Factorielle de n

```
In [10]: def factorielle(n):
        u = 1
        for k in range(1, n + 1):
            u = u * k
        return u
```

```
In [11]: [factorielle(n) for n in range(10)]
```

```
Out[11]: [1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]
```

1.10 Capacité 7 : Modéliser un phénomène discret à croissance linéaire

- Question 1) : $v_1 = 20$ et $v_2 = v_1 - 0,6 = 19,4$.
- Question 2) : pour tout entier n tel que $1 \leq n \leq 23$, on a $v_{n+1} = v_n - 0,6$.
- Question 3) : au douzième mois, le montant de la mensualité est de $v_{12} = v_1 - 11 \times 0,6 = 20 - 6,6 = 13,4$ euros. Plus généralement, pour tout entier n tel que $1 \leq n \leq 24$, on aura $v_n = v_1 - 0,6(n - 1) = 20,6 - 0,6n$.

1.11 Capacité 8 : Modéliser un phénomène discret à croissance exponentielle

- Question 1) : $C_1 = 820$ euros et $C_2 = C_1 \times 1,025 = 840,5$ euros.
- Question 2) : Pour tout entier naturel n , on a $C_{n+1} = 1,025 \times C_n$ (formule de récurrence) et $C_n = C_0 \times 1,025^n$ (formule explicite)
- Question 3) : Algorithme de seuil en Python, fonction retournant le plus petit entier n tel que $C_n > s$

```
def seuil(s):  
    c = 800  
    n = 0  
    while c <= s:  
        c = 1.025 * c  
        n = n + 1  
    return n
```

```
In [13]: def seuil(s):  
        c = 800  
        n = 0  
        print(n, c)  
        while c <= s:  
            c = 1.025 * c  
            n = n + 1  
            print(n, c)  
        return n  
  
        print(seuil(1000))
```

```
0 800  
1 819.9999999999999  
2 840.4999999999998  
3 861.5124999999997  
4 883.0503124999997  
5 905.1265703124996  
6 927.7547345703119  
7 950.9486029345696  
8 974.7223180079338
```


9 999.0903759581321
10 1024.0676353570852
10

1.12 Capacité 11 Déterminer une relation pour une suite définie par un motif géométrique

- **Question 1** : pour tout entier $n \geq 1$, on a $a_{n+1} = a_n + 2$ avec $a_1 = 1$. La suite (a_n) est donc arithmétique de raison 2 et pour tout entier naturel $n \geq 1$, on a $a_n = a_1 + (n - 1) \times 2 = 1 + 2(n - 1) = 2n - 1$.
- **Question 2** : D'après la formule sur la somme des termes consécutifs d'une suite arithmétique, on a $\sum_{k=1}^n a_k = \frac{a_1 + a_n}{2} \times n = \frac{1 + 2n - 1}{2} \times n = n^2$.
- **Question 3** : On résout l'inéquation $\sum_{k=1}^n a_k \geq 1 \iff n^2 \geq 1000 \iff n \geq \sqrt{1000} \iff n \geq 32$. Le robot doit donc effectuer au moins 32 trajets en ligne droite (le dernier sera incomplet) et 31 virage, ce qui pour une distance de 1 kilomètre soit 10^5 centimètres représente un temps de $\frac{10^5}{20} + 31 \times 2 = 5062$ secondes.

1.13 Capacité 12 Modéliser un phénomène discret à croissance exponentielle, calculer le terme général d'une suite géométrique

- **Question 1** : $u_0 = 60$, $u_1 = 30$ et $u_2 = 15$.
- **Question 2** : Pour tout entier naturel n , on a $u_{n+1} = 0,5u_n$ donc la suite (u_n) est géométrique de raison 0,5 et donc $u_n = 0,5^n u_0 = \frac{60}{2^n}$. On en déduit que $u_5 = \frac{60}{2^5} = 1,875$.
- **Question 3** : Fonction `seuil()` qui retourne le plus petit entier n à partir duquel $u_n < 0,25$. Elle retourne 8 donc au bout de 8 demi-vies soit $8 \times 3 = 24$ jours, l'activité radioactive de cet échantillon est strictement inférieure à 0,25.

```
In [2]: def seuil():  
        u = 60  
        n = 0  
        while u >= 0.25:  
            u = u / 2  
            n = n + 1  
        return n
```

```
In [3]: seuil()
```

```
Out [3]: 8
```