Devoir surveillé # 1 -

	Sujet A	Sujet B	Sujet C	Notation
I.1	21	27	3	2
I.2	2639	263	1517	2
II.1	6.78503	4.44720	8.57836	2
II.2	32.5494	8.5941	128.5253	2
II.3	2695	535	13374	2
1 point si résultat cohérent avec la limite trouvée				
III.1	207	51	293	2
1 point si résultat proche de ce qui est attendu				
III.2	293	51	293	2
1 point si résultat proche de ce qui est attendu				
IV.1u_{10}	0.50063	0.33371	0.33346	2
IV.1ℓ	0.50063	0.33371	0.33346	1
IV.2u_{10}	$3.22392.10^{68}$	$2.69004.10^{15}$	$2.18335.10^{142}$	1
IV.2ℓ	∞	∞	∞	1
IV.3	13	9	17	3
1 point si résultat proche de ce qui est attendu				

Le corrigé n'est écrit que pour le sujet A. Pour les autres sujets, il suffit d'adapter les codes fournis.

Les solutions sont fournies sous forme de fonction mais il n'est pas nécessaire d'en utiliser pour répondre aux questions du sujet.

I - Dénombrement et divisibilité

1 Pour ce code, il faut mettre en place un compteur qui va permettre de dénombrer le nombre de fois qu'un nombre `nombre` est divisible par 3, 4, 5 et 7. Le test de divisibilité se fait à l'aide de la commande %. Il suffit de faire ce test de divisibilité au sein d'une boucle `for` variant de `1000` à `10000`.

```
1 def diviseur(a,b,c,d):
2         compteur=0 # Initialisation du compteur
3         for nombre in range(1000,10001): # nombre va varier de 1000 à 10000
4                 # Test de divisibilité par a,b,c et d
5                 if nombre%a==0 and nombre%b==0 and nombre%c==0 and nombre%d==0:
6                         compteur+=1 # Si le test est vrai, on incrémente le compteur.
7         return compteur
8 print("Le nombre de diviseurs est :" , diviseur(3,4,5,7))
```

2 Pour trouver le plus grand diviseur de 13195, il faut parcourir une boucle variant avec un `nombre_test` de 2 à 13194. Initialement, le plus grand diviseur vaut 1. À chaque itération, on va tester si le nombre 13195 est divisible par `nombre_test`. Si c'est le cas, on enregistre le résultat. Le dernier résultat enregistré donnera le diviseur le plus grand.

```
1 def plus_grand_diviseur(nombre):
2         diviseur_le_plus_grand=1 # Initialisation
3         for nombre_test in range(2,nombre): # Parcours de 2 à nombre - 1
4                 # Test de divisibilité
```

```
                if nombre%nombre_test==0:
                        diviseur_le_plus_grand=nombre_test # Enregistrement du résultat
        return diviseur_le_plus_grand
print("Le plus grand diviseur de 13195 est  :" , plus_grand_diviseur(13195))
```

II - Calcul d'une somme

1 Il s'agit ici de refaire, ce qui a été vu en TD sur le calcul d'une somme. Il faut d'abord initialiser une variable `somme`. On incrémente celle-ci à chaque itération `k` de la boucle `for` par `exp(-sqrt(i)/4)`. Le nombre d'itération est ici de $n+1$. En effet, le calcul de la somme (dans l'énoncé) varie pour k allant de 0 à n.

```
def somme_exo2(n):
    somme=0 # Initialisation de la variable sum
    for k in range(n+1): # Pour k variant de 0 à n
        somme=somme+exp(-sqrt(k)/4) # on incrémente la somme à chaque itération
    return somme
print("La valeur de u10 est :",somme_exo2(10))
```

2 Pas de difficultés particulières pour cette question, si la précédente a été correctement faite. Il suffit de calculer u_{100000} par exemple pour avoir une estimation de la limite ℓ. (Il faut prendre un nombre n assez grand).

```
print("La limite de un est :",somme_exo2(100000))
```

3 Pour cette question, on va utiliser une boucle `while`. Cette boucle `while` va également effectuer le calcul de $\sum_k \exp\left(-\sqrt{k}/4\right)$, mais ce calcul se stoppera lorsque l'on aura atteint le critère d'arrêt $\ell - \texttt{sum} < 10^{-3}$.

```
def val_limite_n(l):
        k=0 # Initialisation de la valeur k
        somme=0 # Initialisation de sum
        while l-somme>=10**(-3): # Tant que le critère d'arrêt n'est pas atteint
                somme=somme+exp(-sqrt(k)/4) # On incrémente la valeur de la somme...
                k=k+1 #... puis celle de k
        return k-1 # On retourne k-1
print("La valeur de n0 est :",val_limite_n(32.5494))
```

On retourne `k-1` car lors d'une itération k, on calcule effectivement $\sum_0^k \exp\left(-\sqrt{k}/4\right)$, mais à la fin de la boucle, k vaut $k+1$, ce qui fait qu'entre la valeur de l'indice k en fin de boucle et la valeur de somme calculée, il y a un décalage d'un rang.

III - Somme divergente

La question de cette exercice est semblable à celle faite dans le TD (exercice VII, question 3) et ressemble également à ce qui est fait à la question 3 de l'exercice précédent. Encore une fois, une structure avec une boucle `while` est adaptée.

```
def indice_limite():
    somme=0 # Initialisation de la somme
    k=1 # Initialisation de k, on commence à k=1 car on veut faire le calcul
    # de la somme avec ceux-ci étant impair
    while somme<21500: # Tant que la valeur limite n'est pas atteinte
        somme=somme+k**2*sin(1/k) # on itère le calcul
        k=k+2 # On incrémente de +2 (pour conserver les nombres impairs)
    return k-2 # Attention de bien penser à retirer 2 au résultat final
    #(voir question précédente)
print("La valeur de n0 est :",indice_limite())
```

IV - Étude d'une suite numérique dont la limite varie

1 Dans un premier temps, on définit une fonction permettant de calculer la suite u_n définit de façon récursive. À la vue du sujet, il semble intéressant de renseigner deux paramètres d'entrée à cette fonction : le nombre n utilisé pour calculer u_n et le paramètre de valeur initial u_0.

```
1 def suite(n,u):
2         for i in range(n): # On calcule la valeur de un, puisque le dernier i
3         # calculé vaut n-1
4                 u=u**3/200+1/2 # Ui+1=Ui**3/200+1/2
5         return u # On retourne la valeur de Un
6 print("u10 vaut : ",suite(10,1))
```

Pour le calcul de la limite, il suffit de prendre une valeur de n élevée :

```
1 print("La limite vaut : ",suite(100000,1))
```

2 Il faut refaire ce qui a été fait précédemment mais en changeant la valeur de u_0.

```
1 print("u7 vaut : ",suite(7,15))
2 print("La limite vaut : ",suite(100000,15))
```

On constate que pour le calcul de la limite amène une erreur : OverflowError : (34, 'Result too large'), ce qui signifie que la valeur du calcul est trop grande. La limite tend vers $+\infty$.

3 Pour cette question, le plus simple est de tout simplement tester les valeurs limites pour u_0 variant de 1 à 15. Quand une erreur apparait, c'est que la valeur initiale était trop importante.

```
1 for i in range(1,16):
2     print("La limite est :",suite(100000,i),"La valeur de i est :",i)
```

L'erreur apparait pour $i = 14$. La réponse attendue est donc 13.