

Interrogation d'Informatique n°1

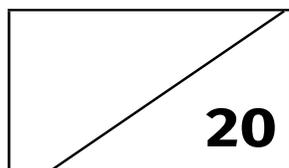
Jeudi 09/11/2023
Durée : 30 minutes

Consignes

- En Mathématiques, les énoncés du cours doivent être complètement écrits : un cadre, des hypothèses et une conclusion.
- En Informatique : les scripts doivent être correctement indentés, en mettant en valeur l'indentation à l'aide d'une barre verticale.
- La note finale tiendra compte, directement ou indirectement, de la qualité de la rédaction et de la présentation.
- Le crayon à papier ne sera pas corrigé. ● L'usage de la calculatrice est interdit.

Nom :

Prénom :

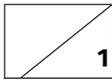


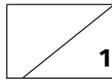
Exercice 1 | Des crochets en veux-tu en voilà

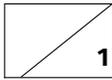
[— / 7]

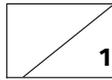
On considère la liste $L = [6, 7.0, [1, 2, 3], "bcpst", 8]$. On rappelle un exemple d'utilisation de la méthode `append` sur les listes :

```
>>> M = [1, 2, 3]
>>> M.append("abc")
>>> M
[1, 2, 3, 'abc']
```

1.  Que renvoie `L[1]` ? 

2.  Que renvoie `L[-1]` ? 

3.  Que renvoie `len(L)` ? 

4.  Que renvoie `L[2][1]` ? 

5.  Que renvoie `L[1:] [1][1]` ? *On expliquera soigneusement le résultat.* 

Comment, avec une commande Python et à l'aide du recours à L, pouvez-vous :

6.  Accéder au "b" de "bcpst" ? 

7.  Compléter la liste L pour qu'elle devienne la liste ci-après ? 

`[6, 7.0, [1, 2, 3], "bcpst", 8, 12]`

Exercice 2 | Une suite définie par une somme

[_____ / 7]

On considère la suite (v_n) définie pour tout $n \in \mathbb{N}^*$ par : $v_n = \sum_{k=n}^{2n} \frac{1}{k}$.

Par exemple, $v_1 = \sum_{k=1}^2 \frac{1}{k} = 1 + \frac{1}{2} = \frac{3}{2}$.

1.  Calculer $v_2 = \sum_{k=2}^4 \frac{1}{k}$. (Ecrire le résultat sous forme de fraction irréductible)



2.  Écrire une fonction d'en-tête `suite(n)` qui prend en argument un entier n et retourne la valeur de v_n .



3. On admet que la suite (v_n) converge vers $\ln(2)$, autrement dit, v_n est aussi proche de $\ln(2)$ que l'on veut pourvu que n soit assez grand. On rappelle que la fonction logarithme népérien peut être obtenue à l'aide du module `math` et s'appelle `log`.

3.1)  Indiquer une commande permettant d'importer la fonction `log` du module `math`.



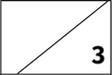
3.2)  Compléter la fonction `approx(eps)` qui prend en argument un nombre $\varepsilon > 0$ et retourne le premier entier n tel que $|v_n - \ln(2)| \leq \varepsilon$ (on

rappelle que la fonction `abs` de Python permet d'obtenir la valeur absolue d'un nombre réel, et il n'est pas nécessaire d'importer cette fonction. On supposera la fonction `log` chargée).

```
def approx(eps):
    n = 0
    while _____ :
        _____
    return n
```

Exercice 3 | Nombre d'occurrences dans une liste

[_____ / 6]

1.  Compléter le programme suivant pour que la fonction `occurrences(L, e)` compte le nombre de fois où l'élément e apparaît dans une liste L . Par exemple, `occurrences([6, 8, 4, 0, 4, 3], 4)` renvoie `2`, tandis que `occurrences([6, 8, 4, 0, 4, 3], 7)` renvoie `0`.

```
def occurrences(L, e):
    C = 0 # Compteur du nombre d'occurrences
    for x in L:
        if _____ :
            _____
    return C
```

2.  En déduire une fonction d'en-tête `tout_est_present(M, L)` qui retourne `True` si tous les éléments de M sont présents dans L au moins une fois, et `False` dans le cas contraire. Par exemple, `tout_est_present([1, 2], [1, 2, 3])` retournera `True`, alors que `tout_est_present([1, 2, 3], [1, 2])` retournera `False` puisque `3` n'est pas présent dans `[1, 2]`.



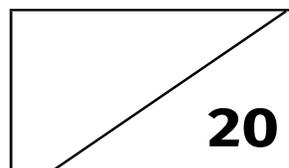
Interrogation d'Informatique n°1

Jeudi 09/11/2023
Durée : 30 minutes

Consignes

- En Mathématiques, les énoncés du cours doivent être complètement écrits : un cadre, des hypothèses et une conclusion.
- En Informatique : les scripts doivent être correctement indentés, en mettant en valeur l'indentation à l'aide d'une barre verticale.
- La note finale tiendra compte, directement ou indirectement, de la qualité de la rédaction et de la présentation.
- Le crayon à papier ne sera pas corrigé. ● L'usage de la calculatrice est interdit.

Nom :		Prénom :	
-------	--	----------	--



Exercice 1 | Des crochets en veux-tu en voilà

[— / 7]

On considère la liste $L = [5, 6.0, [1, 2, 3], "bcpst", 9]$. On rappelle un exemple d'utilisation de la méthode `append` sur les listes :

```
>>> M = [1, 2, 3]
>>> M.append("abc")
>>> M
[1, 2, 3, 'abc']
```

1. Que renvoie `L[1]` ?

2. Que renvoie `L[-1]` ?

3. Que renvoie `len(L)` ?

4. Que renvoie `L[2][1]` ?

5. Que renvoie `L[2:][1][1]` ? *On expliquera soigneusement le résultat.*

Comment, avec une commande Python et à l'aide du recours à L, pouvez-vous :

6. Accéder au "p" de "bcpst" ?

7. Compléter la liste L pour qu'elle devienne la liste ci-après ?

`[5, 6.0, [1, 2, 3], "bcpst", 9, 11]`



Exercice 2 | Une suite définie par une somme

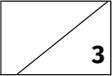
[_____ / 7]

On considère la suite (u_n) définie pour tout $n \in \mathbb{N}^*$ par : $u_n = \sum_{k=n}^{3n} \frac{1}{k}$.

Par exemple, $u_2 = \sum_{k=2}^6 \frac{1}{k} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6}$.

1.  Calculer $u_1 = \sum_{k=1}^3 \frac{1}{k}$. (Ecrire le résultat sous forme de fraction irréductible)



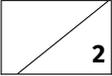
2.  Écrire une fonction d'en-tête `suite(n)` qui prend en argument un entier n et retourne la valeur de u_n .



3. On admet que la suite (u_n) converge vers $\ln(3)$, autrement dit, u_n est aussi proche de $\ln(3)$ que l'on veut pourvu que n soit assez grand. On rappelle que la fonction logarithme népérien peut être obtenue à l'aide du module `math` et s'appelle `log`.

- 3.1)  Indiquer une commande permettant d'importer la fonction `log` du module `math`.



- 3.2)  Compléter la fonction `approx(eps)` qui prend en argument un nombre $\varepsilon > 0$ et retourne le premier entier n tel que $|u_n - \ln(3)| \leq \varepsilon$ (on

rappelle que la fonction `abs` de Python permet d'obtenir la valeur absolue d'un nombre réel, et il n'est pas nécessaire d'importer cette fonction. On supposera la fonction `log` chargée).

```
def approx(eps):
    n = 0
    while _____ :
        _____
    return n
```

Exercice 3 | Nombre d'occurrences dans une liste

[_____ / 6]

1.  Compléter le programme suivant pour que la fonction `compte(L, e)` compte le nombre de fois où l'élément e apparaît dans une liste L .
Par exemple, `compte([6, 8, 4, 0, 4, 3], 4)` renvoie `2`, tandis que `compte([6, 8, 4, 0, 4, 3], 7)` renvoie `0`.

```
def compte(L, e):
    C = 0 # Compteur du nombre d'occurrences
    for x in L:
        if _____ :
            _____
    return C
```

2.  En déduire une fonction d'en-tête `tout_est_present(M, L)` qui retourne `True` si tous les éléments de M sont présents dans L au moins une fois, et `False` dans le cas contraire.
Par exemple, `tout_est_present([1, 2], [1, 2, 3])` retournera `True`, alors que `tout_est_present([1, 2, 3], [1, 2])` retournera `False` puisque `3` n'est pas présent dans `[1, 2]`.

