Domaine : Programme complet - Question n° 1

SuperSauteur a super besoin de super beaucoup de temps pour super-sauter super loin et super vite.

Le jour de son premier anniversaire il pouvait super-sauter en une seconde à : 1^2 centimètre.

Le jour de son second anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2$ centimètres.

Le jour de son troisième anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2 + 3^2$ centimètres.

Le jour de son quatrième anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2 + 3^2 + 4^2$ centimètres.

etc.

On souhaite écrire une fonction `calculer_longueur_saut` qui prend en paramètre un entier positif n et retourne le nombre entier correspondant à la longueur de saut de SuperSauteur le jour de son n ième anniversaire.

1. Donner le prototype de cette fonction.

   Le nom de la fonction est calculer_longueur_saut.

   Elle admet un paramètre n qui est un entier positif.

   Elle retourne un nombre entier positif.

2. Faut-il utiliser une boucle bornée ou une boucle non-bornée ? Justifier la réponse.

   On doit "boucler" de 1 à n (n étant l'age de SuperSauteur).

   On connait donc le nombre de fois que la boucle s'exécutera.

   Il faut donc utiliser une boucle bornée.

3. Noémie a étudié les variables dont elle pense avoir besoin pour coder cette fonction en langage Python. Elle a noté sur son brouillon :

   - `n` : âge de SuperSauteur
   - `anniversaire` va parcourir 1, 2, 3, ... n
   - `longueur_saut = longueur_saut + anniversaire ** 2`

Son enseignant lui dit : "c'est un très bon début, il n' y a plus qu'à formaliser tout ça"

En utilisant les mêmes noms de variables que Noémie, écrire le corps de la fonction *calculer_longueur_saut(n)*

```
def calculer_longueur_saut(n):
    longueur_saut = 0
    for anniversaire in range(1,n+1):
        longueur_saut = longueur_saut + anniversaire**2
    return longueur_saut
```

4. Ecrire le code d'une seconde fonction *calculer_age_mur_du_son* qui ne prend aucun paramètre mais retourne l'âge à partir duquel SuperSauteur dépassera la vitesse du son lors de ses sauts (c'est-à dire fera des sauts d'une longueur supérieure à 34 000 centimètres en une seconde)

```
def calculer_age_mur_du_son():
    age_mur_son = 0
    anniversaire = 1
    while age_mur_son < 34000:
        age_mur_son = calculer_longueur_saut(anniversaire)
        anniversaire += 1
    return anniversaire - 1

print("SuperSauteur dépassera la vitesse du mur du son a",calcu
```

Domaine : Programme complet - Question n° 2

Depuis ses huit ans, SuperAvare économise des kopecks (monnaie locale).

Le jour de ses huit ans il avait 8^3=512 kopecks dans sa tirelire.

Le jour de ses neuf ans il avait $512+9^3=1241$ kopecks dans sa tirelire.

Le jour de ses dix ans il avait $1241+10^3=2241$ kopecks dans sa tirelire.

Le jour de ses onze ans il avait $2241+11^3=3572$ kopecks dans sa tirelire.

etc.

On souhaite écrire une fonction `calculer_tirelire` qui prend en paramètre un entier positif n et retourne le nombre entier correspondant au montant de la tirelire de SuperAvare le jour de son n ième anniversaire.

1. Donner le prototype de la fonction calculer_tirelire

   Le nom de la fonction est calculer_tirelire.

   Elle admet un paramètre qui est un entier positif.

   Elle retourne un entier positif.

2. Faut-il utiliser une boucle bornée ou une boucle non-bornée ? Justifier la réponse.

   On doit "boucler" de 1 à n (n étant l'age de SuperSauteur).

   On connait donc le nombre de fois que la boucle s'exécutera.

   Il faut donc utiliser une boucle bornée.

3. Lucie a étudié les variables dont elle pense avoir besoin pour coder cette fonction en langage python. Elle a noté sur son brouillon : - n = âge de SuperAvare - anniversaire = 8, 9, 10, 11, …n - somme_tirelire = somme_tirelire + anniversaire**3 Son enseignante lui a dit : "c'est un très bon début, en plus c'est assez clair".

   - `n` : âge de SuperAvare
   - `anniversaire` va parcourir 8, 9, 10, 11 ... n
   - `somme_tirelire = somme_tirelire + anniversaire ** 3`

   Son enseignante lui dit : "c'est un très bon début, il n' y a plus qu'à formaliser tout ça"

   En utilisant les mêmes noms de variables que Lucie, écrire le corps de la fonction `calculer_tirelire(n)`

```
def calculer_tirelire(n):
  tirelire = 0
  for age in range(8,n+1):
    tirelire = tirelire + age**3
  return tirelire
```

4. écrire le code d'une seconde fonction *age_millionaire()* qui ne prend aucun paramètre mais retourne l'âge à partir duquel SuperAvare sera kopeck-millionnaire.

```
def calculer_age_millionnaire():
  age_millionnaire = 8
  while calculer_tirelire(age_millionnaire) < 1000000:
    age_millionnaire = age_millionnaire + 1
  return age_millionnaire


print("SuperAvare sera kopeck-milionnaire a",calculer_age_milli
```

Domaine : Boucle while - Question n° 3

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
r = 0
while b != 0:
    r = a % b
    a = b
```

```
b = r
```

1. lorsque a = 10 et b = 3

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
10	3	1
3	1	0
1	0	0

   La valeur de a à la fin de l'exécution du programme est 1.

2. lorsque a = 10 et b = 15

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
10	15	10
15	10	5
10	5	0
5	0	0

   La valeur de a à la fin de l'exécution du programme est 5.

3. Que permet de calculer ce programme ?

   Ce programme permet de calculer le PGCD (Plus Grand Commun Diviseur) des nombres a et b.

   L'algorithme utilisé est basé sur le lemme d'Euclide.

Domaine : Boucle while - Question n° 4

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
a = 5
b = 3
```

```
a = b * a
b = a - 2
print("a =", a)
print("b =", b)
```

1. lorsque a = 6 et b = 15

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
6	15	6
15	6	3
6	3	0
3	0	0

   La valeur de a à la fin de l'exécution du programme est 3.

2. lorsque a = 6 et b = 5

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
6	5	1
5	1	0
1	0	0

   La valeur de a à la fin de l'exécution du programme est 1.

3. Que permet de calculer ce programme ?

   Ce programme permet de calculer le PGCD (Plus Grand Commun Diviseur) des nombres a et b.

   L'algorithme utilisé est basé sur le lemme d'Euclide.

Domaine : Boucle while - Question n° 5

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
a = 5
b = 3
a = b * a
b = a - 2
print("a =", a)
print("b =", b)
```

1. lorsque a = 15 et b = 21

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
15	21	15
15	6	3
6	3	0
3	0	0

   La valeur de a à la fin de l'exécution du programme est 3.

2. lorsque a = 15 et b = 7

   On peut utiliser le tableau d'exécution suivant :

a	b	r
.	.	0
15	7	1
7	1	0
1	0	0

   La valeur de a à la fin de l'exécution du programme est 1.

3. Que permet de calculer ce programme ?

   Ce programme permet de calculer le PGCD (Plus Grand Commun Diviseur) des nombres a et b.

   L'algorithme utilisé est basé sur le lemme d'Euclide.

Domaine : Corriger un programme - Question n° 6

On veut calculer la somme des inverses des n premiers entiers.

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def somme_inv_entiers(n)
    for i in range(n):
        s = s + 1/i
    return i
```

Voici la liste des erreurs dans ce programme :

- il manque le caractère deux points à la fin de la première ligne.
- Le `range(n)` est incorrect : en effet, pour `i = 0` il y aura une erreur division par 0, à la ligne 3 : `s = s + 1/i`.

  il faut indiquer : `range(1,n+1)`
- A la dernière ligne, il faut retourner `s` et non `i`.

Voici le programme corrigé :

```
def somme_inv_entiers(n):
    for i in range(1,n+1):
        s = s + 1/i
    return s
```

Domaine : Corriger un programme - Question n° 7

On veut calculer le produit des entiers de 1 à n (n est étant un entier supérieur ou égal à 1).

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def produit_entiers(n):
    p = 0
    for i in range(n)
        p = p * n
    return n
```

Voici la liste des erreurs dans ce programme :

- il manque le caractère deux points à la fin de la ligne avec l'instruction *for*.
- Il faut initialiser la variable *p* à 1 et non à 0. (sinon le produit calculé à la sortie de la boucle sera toujours égal à 0.)

  il faut indiquer : *range(1,n+1)* et non *range(n)* (sinon le produit calculé à la sortie de la boucle sera toujours égal à 0)
- A la dernière ligne, il faut retourner *p* et non *n*.

Voici le programme corrigé :

```
def produit_entiers(n):
    p = 1
    for i in range(1,n+1):
        p = p * i
    return p
```

Domaine : Corriger un programme - Question n° 8

On veut écrire une fonction qui détermine si un entier est un nombre premier.

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def est_premier():
    if nombre <= 1:
        return False
    diviseur = 2
    while diviseur < nombre
        if nombre % diviseur = 0:
            return True
    return False
```

Voici la liste des erreurs dans ce programme :

- il manque le caractère deux points à la fin de la ligne avec l'instruction `while`.
- il manque le paramètre `nombre` à définir dans l'entête de la fonction : `def est_premier(nombre):`

  le test `if nombre % diviseur = 0` est incorrect, il faut utiliser un double = : `if nombre % diviseur = 0`
- dans la séquence :

```
if nombre % diviseur == 0:
  return True
```

  Si `nombre % diviseur == 0`, alors `diviseur` divise `nombre` et donc `nombre` n'est pas premier, il faut donc retourner `False` et non `True`..
- A la dernière ligne, si aucun diviseur n' a été trouvé, le nombre est premier et il faut retourner `True` et non `False`

Voici le programme corrigé :

```
def est_premier(nombre):
    if nombre <= 1:
        return False
    diviseur = 2
    while diviseur < nombre:
        if nombre % diviseur == 0:
            return False
```

```
            diviseur += 1
        return True
```

Domaine : Tableau exécution - Question n° 9

Soit le programme Python suivant :

```
cpt = 0
for i in range(4):
    for j in range(i):
        if (i + j) % 2 == 0:
            cpt = cpt + 1
```

Remplir le tableau d'exécution suivant (en utilisant le nombre de lignes nécessaires)

i	j	cpt
.	.	0
0	.	0
1	0	0
2	0	1
2	1	1
3	0	1
3	1	2
3	2	2

Domaine : Tableau exécution - Question n° 10

Soit le programme Python suivant :

```
cpt = 0
for i in range(2,5):
    for j in range(i-1):
```

```
        if (i + j) % 2 == 1:
            cpt = cpt + 1
```

Remplir le tableau d'exécution suivant (en utilisant le nombre de lignes nécessaires)

i	j	cpt
.	.	0
2	0	0
3	0	1
3	1	1
4	0	1
4	1	2
4	2	2

Domaine : Linux - Question n° 11

"Linux ou GNU/Linux est une famille de systèmes d'exploitation open source de type Unix fondée sur le noyau Linux, créé en 1991 par Linus Torvalds. De nombreuses distributions Linux ont depuis vu le jour et constituent un important vecteur de popularisation du mouvement du logiciel libre."

Source : Wikipédia, extrait de l'article consacré à GNU/Linux.

"Windows est au départ une interface graphique unifiée produite par Microsoft, qui est devenue ensuite une gamme de systèmes d'exploitation à part entière, principalement destinés aux ordinateurs compatibles PC. Windows est un système d'exploitation propriétaire."

Source : Wikipédia, extrait de l'article consacré à Windows.

1. Expliquer succinctement les différences entre les logiciels libres et les logiciels propriétaires.

1. Accessibilité du code source

- **Logiciel libre:** Le code source est ouvert et accessible à tous. Les utilisateurs ont la liberté de l'étudier, de le modifier et de le redistribuer.
- **Logiciel propriétaire:** Le code source est fermé et secret. Seuls les propriétaires du logiciel ont le droit de le modifier ou de le distribuer.

2. Liberté d'utilisation et de modification

- **Logiciel libre:** Les utilisateurs ont la liberté d'utiliser le logiciel pour n'importe quel usage, de le copier, de le modifier et de le redistribuer, souvent sous certaines conditions (licences libres).
- **Logiciel propriétaire:** Les utilisateurs sont soumis à des restrictions d'utilisation définies par le propriétaire du logiciel. Ils n'ont généralement pas le droit de modifier le logiciel ou de le redistribuer.

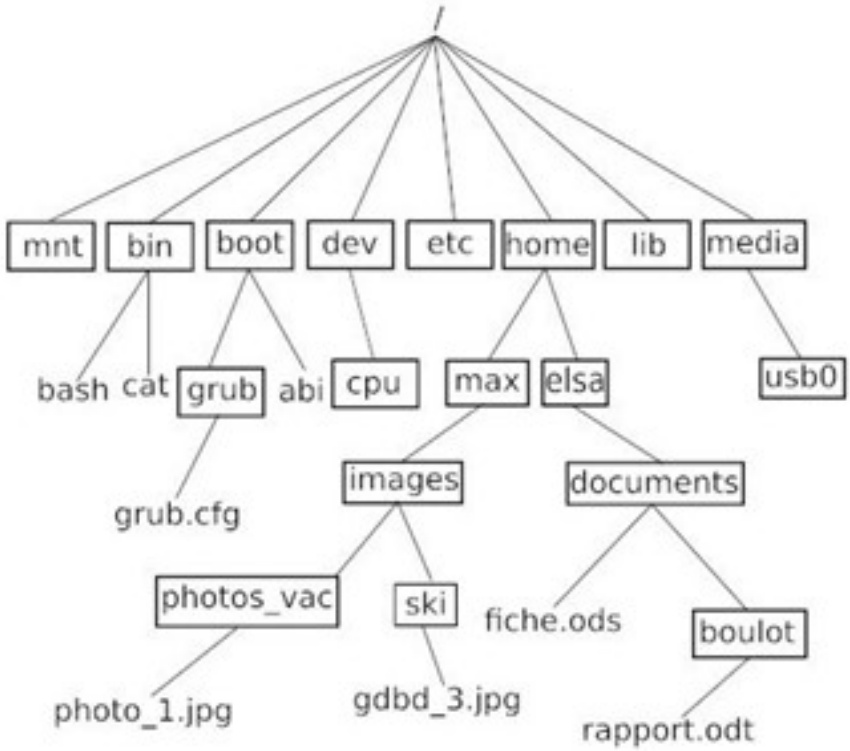
3. Modèle économique

- **Logiciel libre:** Le logiciel libre est souvent gratuit, mais il peut aussi être payant (par exemple, pour des services d'assistance ou des versions spécifiques). Le modèle économique peut reposer sur la communauté, les dons, ou la vente de services associés.
- **Logiciel propriétaire:** Le logiciel propriétaire est généralement payant. Les revenus sont générés par la vente de licences d'utilisation.

2. Expliquer le rôle d'un système d'exploitation.

   Le système d'exploitation est un logiciel essentiel qui gère les ressources matérielles (comme le processeur, la mémoire et le stockage) d'un ordinateur et fournit des services de base pour les autres programmes. Il sert d'intermédiaire entre l'utilisateur et le matériel, permettant ainsi aux applications de fonctionner et à l'utilisateur d'interagir avec l'ordinateur de manière conviviale.

On donne ci-dessous une arborescence de fichiers sur un système GNU/Linux (les noms encadrés représentent des répertoires, les noms non encadrés représentent des fichiers, / correspond à la racine du système de fichiers) :



3. Indiquer le chemin absolu du fichier *rapport.odt*.

Le chemin absolu du fichier *rapport.odt* est :
/home/elsa/documents/boulot/rapport.odt

On suppose que le répertoire courant est le répertoire *elsa*.

4. Indiquer le chemin relatif du fichier *photo_1.jpg*.

Le chemin relatif du fichier *photo_1.jpg* est :
../max/images/photos_vac/photo_1.jpg

L'utilisatrice Elsa ouvre une console (aussi parfois appelée terminal), le répertoire courant étant toujours le répertoire *elsa*.

On fournit ci-dessous un extrait du manuel de la commande UNIX *cp* :

```
NOM
  cp - copie un fichier
UTILISATION
  cp fichier_source fichier_destination
```

5. Déterminer le contenu du répertoire *documents* et du répertoire *boulot* après avoir exécuté la commande suivante dans la console :

```
cp documents/fiche.ods documents/boulot
```

contenu du répertoire *documents* : le fichier *fiche.ods*

contenu du répertoire *boulot* : les fichiers *fiche.ods* et *rapport.odt*

<

Domaine : Tests et spécification de fonctions - Question n° 12

Reprendre le code de ce programme et y ajoutant :

- la description de ce que fait cette fonction à l'aide des *docstrings*
- 2 tests au format *doctest*
- Deux instructions *assert* testant la fonction dont une des deux doit provoquer une erreur.

```
def mystere(chaine,c):
    n = 0
    for l in chaine:
        if l == c:
            n = n + 1
    return n
```

```
import doctest

def mystere(chaine,c):
    """
    Compte le nombre d'occurences du caréactère c dans la chaine de
    :param: chaine : une chaine de caractères
    :param: c: un caractère
    :return n : le nombre de d'occurences du caractere c dans chain
    >>> mystere("abracadabra","a")
    5
    >>> mystere("abracadabra","b")
    2
    """
    n = 0
    for ch in chaine:
        if ch == c:
```

```
            n = n + 1
    return n

assert mystere("abracadabra","a") == 5
assert mystere("abracadabra","b") == 3

doctest.testmod()
```

Domaine : Tests et spécification de fonctions - Question n° 13

Reprendre le code de ce programme et y ajoutant :

- la description de ce que fait cette fonction à l'aide des *docstrings*
- 2 tests au format *doctest*
- Deux instructions *assert* testant la fonction dont une des deux doit provoquer une erreur.

```
def mystere(chaine,c):
    for x in chaine:
        if c == x:
            return True
    return False
```

```
import doctest

def mystere(chaine,c):
    """
    Détermine si le caractère c est présent dans la chaine c
    :param: chaine : une chaine de caractères
    :param: c: un caractère
    :return n : le nombre de d'occurences du caractere c dans chain
    >>> mystere("abracadabra","a")
    True
    >>> mystere("abracadabra","q")
```

```
    False
    """
    for x in chaine:
        if c == x:
            return True
    return False

assert mystere("abracadabra","a") == True
assert mystere("abracadabra","b") == False

doctest.testmod()
```