Sujet d'évaluation – Algorithmes

Exercice 1

Écrivez une fonction, `persistance`, qui prend un paramètre positif `num` et renvoie sa persistance multiplicative, qui est le nombre de fois que vous devez multiplier les chiffres de `num` jusqu'à atteindre un seul chiffre.

```
Exemple:
39 --> 3 (car 3*9 = 27, 2*7 = 14, 1*4 = 4 et 4 n'a qu'un seul chiffre)
999 --> 4 (car 9*9*9 = 729, 7*2*9 = 126, 1*2*6 = 12, et enfin 1*2 = 2)
4 --> 0 (car 4 est déjà un nombre à un chiffre)
```

Exercice 2

La racine numérique est la somme récursive de tous les chiffres d'un nombre.

Étant donné n , prenons la somme des chiffres de n . Si cette valeur comporte plus d'un chiffre, continuez à réduire de cette manière jusqu'à ce qu'un nombre à un chiffre soit produit. L'entrée sera un entier non négatif.

```
16 --> 1 + 6 = 7
942 --> 9 + 4 + 2 = 15 --> 1 + 5 = 6
132189 --> 1 + 3 + 2 + 1 + 8 + 9 = 24 --> 2 + 4 = 6
493193 --> 4 + 9 + 3 + 1 + 9 + 3 = 29 --> 2 + 9 = 11 --> 1 + 1 = 2
```

Exercice 3

Construisez une tour en forme de pyramide, sous la forme d'un tableau/liste de chaînes, étant donné un nombre entier positif d'étages. Une tour est représentée par le caractère "*".

Par exemple, une tour de 3 étages ressemble à ceci :

```
[
    "  *  ",
    " *** ",
    "*****"
]
```

6 etages:

```
[
"     *     ",
```

```
"    ***    ",
"   *****   ",
"  *******  ",
" ********* ",
"***********"
]
```

Sujet d'évaluation – Programmation Orientée Objet en Python

Contexte

Vous êtes sollicité par une DSI pour concevoir un outil de **modélisation du cycle de vie des machines et services d'un système d'information**, répartis dans différents datacenters. Ce modèle permettra, à terme, d'être exploité par une future API de supervision.

L'objectif est de construire un ensemble de classes permettant de :

- représenter les serveurs (physiques ou virtuels),
- suivre les services qui y sont installés,
- structurer les datacenters,
- tracer les maintenances effectuées,
- gérer les techniciens affectés à ces maintenances.

Objectifs pédagogiques

L'évaluation porte sur :

- La capacité à modéliser des objets réels,
- L'usage correct de l'héritage et de la composition,
- L'utilisation pertinente de méthodes spéciales (`__str__`, `__eq__`, `__len__`, etc.),
- La qualité de l'organisation du code.

Fonctionnalités attendues

Classe `Service`

- Représente une application déployée sur un serveur.
- Attributs : nom, port, protocole, critique (booléen).
- Peut être comparé à un autre service (nom + port).

Classe `Serveur`

- Attributs communs : nom, IP, OS, date de mise en service.
- Possède une liste de services.
- Fournit des méthodes pour ajouter/retirer un service.

- Implémente des méthodes spéciales pertinentes.

Classe `ServeurPhysique` et `ServeurVirtuel`

- `ServeurPhysique` : rack, consommation en kW, garantie (booléen).
- `ServeurVirtuel` : hyperviseur, allocation (ex: "4 vCPU / 8 Go RAM").

Classe `Technicien`

- Nom, spécialité (Linux, virtualisation, réseau...), et identifiant.
- Peut être affecté à une ou plusieurs maintenances.

Classe `Maintenance`

- Concerne un serveur donné.
- Attributs : identifiant, date, type de maintenance (préventive, corrective), technicien responsable.
- Peut être liée à un serveur.
- Fournit une méthode permettant d'afficher un résumé clair.

Classe `Datacenter`

- Contient une liste de serveurs.
- Peut lister tous les services critiques hébergés.
- Peut rechercher les serveurs maintenus par un technicien donné.
- Peut calculer le nombre total de services hébergés.

Contraintes

- Le code doit être structuré et lisible.
- L'héritage doit être justifié (pas d'héritage systématique).
- La composition doit être utilisée à bon escient.