Programmation avancée, partie python,
Upssitech SRI 2e année – 2018/2019
Durée : 1h30

Avertissement : **la compréhension des questions fait partie du sujet**
Le barême est indicatif et susceptible d'ajustements.

Exercice 1 : Echauffement (6pts)

Les questions sont ici indépendantes.

1. Ecrire une fonction dict_paires qui prend un dictionnaire dont les valeurs sont elles mêmes des dictionnaires, et qui renvoie un nouveau dictionnaire dont les clés sont des paires combinant les clés des 2 niveaux (cf exemple plus bas).

2. Ecrire une fonction dict_split qui fait l'opération exactement inverse: à partir d'un dictionnaire indexé par des paires, faire un dictionnaire indexé par le 1er élément de la paire et dont les valeurs sont des dictionnaires indexés par toutes les valeurs associées à cet élément dans une paire.

Par exemple:

```
d = { 1: {"a":2,"b":4},
      2: {"c":4},
      5: {"b":4,"d":2,"e":7},
    }
dict_paires(d) -> {(1, 'b'): 4, (1, 'a'): 2, (5, 'e'): 7, (5, 'd'): 2, (2, 'c'): 4, (5, 'b'): 4}
```

Et dict_split(dict_paires(d)) renverrait un dictionnaire identique à d.

Exercice 2 : Objets (5 pts)

On veut définir une classe similaire à une liste, et qui pour chaque instance va compter combien de fois on accède à un élément de la liste. Pour cela définissez une classe qui doit être initialisé avec une valeur itérable, qui stocke les valeurs de cet itérable dans un attribut de l'instance, et qui comporte les deux méthodes suivantes

1. `__get_item__` pour accéder à un élément de la liste stockée.

2. `__repr__` pour retourner une chaine représentant la liste stockée, pour affichage.

Cela donnerait ce comportement:

```
cl = ListCompte(range(10))
print(cl)
print(cl.counter)
cl[4]+cl[2]
print(cl.counter)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
0
2
```

Exercice 3 : Calcul vectoriel (6pts)

Les questions suivantes doivent être résolues avec les fonctionalités de la bibliothèque numpy **sans boucle ou récursion** ou bien la solution vaudra 0.
On rappelle les fonctions suivantes, qu'on considère déjà importées sous ce nom:

- mean(vecteur): fait la moyenne des éléments du vecteur. Si le vecteur est booléen, considère que True=1, False=0
- arange(n): renvoie un vecteur des valeurs entières de 0 à n-1.
- zeros(n): renvoie un vecteur de taille n avec toutes les valeurs à 0.

1. écrire une fonction qui approxime la valeur de pi en utilisant la méthode suivante: tirer au hasard n valeurs de points (x,y) de coordonnées entre 0 et 1, la proportion de points qui sont dans le cercle de rayon 1 (c'est-à-dire tels que $x^2 + y^2 \leq 1$) est une approximation de $\pi/4$. n est un paramètre de la fonction.

   Vous pouvez, par exemple, générer des vecteurs de x et de y séparément avec la fonction de numpy random.uniform(size=n), qu'on suppose importée sous le nom "uniform" directement.

2. en considérant un polynôme représenté par un numpy.array de taille 6 maximum, par exemple $x + 3x^2 + 2x^4$ est représenté par `p = numpy.array([0,1,3,0,2,0])`, écrire une fonction qui calcule la représentation de la dérivée d'un polynôme. Dans le cas de p, cela donnerait le vecteur [1. 6. 0. 8. 0. 0.], représentant $1 + 6.0x + 8.0x^3$.

Ex. 4 : Fonctions d'ordre supérieur (3pts)

1. Ecrire une fonction filter qui prend comme paramètre une fonction f et qui renvoie son résultat s'il est positif, ou zéro sinon (on suppose que le résultat de f est numérique).

   Exemple d'utilisation :

```
def bof(y):
    return y

print(filter(bof(-2)))   -> 0
print(filter(bof(10)))   -> 10
```

2. Que faut-il changer pour en faire un décorateur, avec un argument en plus qui correspond au seuil auquel on coupe le résultat (tout résultat inférieur au seuil est ramené au seuil, comme avec zéro dans la question précédente).

   Exemple d'utilisation :

```
@filter(10)
def bof(y):
    return y

print(bof(5)) -> 10
print(bof(15)) -> 15
```