Programmation fonctionnelle 2/2

Exercices

Exercice 1 :

Les fonctions `map` et `filter` peuvent être écrites en python à l'aide de la syntaxe plus moderne des listes en compréhension(voir cours de première).

Exemple :

In [ ]:
```
#map
def f(x):
    return x**2

list(map(f,range(9)))
```

In [ ]:
```
#en compréhension
[f(x) for x in range(9)]
```

1. Réécrire le code ci-dessous avec une liste en compréhension pour obtenir la même résultat.

In [ ]:
```
 list(map(lambda x: 2*x, range(10)))
```

In [ ]:
```
#en compréhension

#
```

2. Même question

In [ ]:
```
list (filter(lambda x: x>5, range(10)))
```

In [ ]:
```
#en compréhension

#
```

Exercice 2:

Ecrire une fonction `trouve(p,tab)` , qui prend en paramètres une fonction `p` ("p" comme propriété) et un tableau `tab` , qui renvoie le premier élément `x` de `tab` tel que `p(x)` est vraie. Si aucun élément de `tab` ne satisfait `p` , on renvoie `None` .

On pourra s'appuyer sur les fonctions proposées ci-dessous pour les tests.

In [ ]:
```
#propriétés
def pair(n):
    return n%2==0

def positif(n):
    return n>0
```

In [ ]:
```
#réponse
def trouve(p, tab):




    pass
```

In [ ]:
```
#tests
T=[-1,0,1]
print(trouve(pair, T))
print(trouve(positif, T))
```

Exercice 3 :

Ecrire une fonction `double(f)` qui reçoit une fonction `f` en argument et renvoie une fonction qui applique deux fois de suite la fonction `f` .

On pourra s'appuyer sur les fonctions proposées ci-dessous pour les tests.

In [ ]:
```
#fonctions
def factorielle(n):
    if n<=1:
        return 1
    else:
        return n*factorielle(n-1)

def exp(x):
    return 2**x

def separe(chaine):
    res=''
    s='.'
    for car in chaine:
        if car !=s:
            res=res+car+s
        else:
            res=res+car
    return res
```

In [ ]:
```
#réponse
def double(f):



    pass
```

In [ ]:
```
#tests
print(double(factorielle)(3))
print(double(exp)(3))
print(double(separe)('nsi'))
```

Exercice 4 :

On considère la liste d'URLS suivante:

In [ ]:
```
urls = [
'http://www.je-suis.fr',
'https://www.google.com' ,
'www.je-suis' ,
'http://www.abc.co',
'http://inexistant.com' ,
]
```

1. Compléter la fonction `checkurl(url)` qui prend en paramètre une chaîne de caractères et qui renvoie `True` si cette chaîne est une url correcte et `False` sinon(on pourra commencer par séparer l'url en trois parties à l'aide de laméthode `.split()`).

In [ ]:
```
def checkurl(url):
    sep=url.split('.')




    pass
```

2. A l'aide de la fonction `filter` ou d'une liste en compréhension, afficher la liste des url valides de la liste `urls` .

In [ ]:
```

```

Exercice 5:

On considère la liste d'adresses IP suivante:

In [ ]:
```
ips=['127.0.0.1',
     '255.255.255.0',
     '127.256.41.2',
     '10.14.145.2',
     '1.10.10']
```

1. Compléter la fonction `checkip(ip)` qui prend en paramètre une chaîne de caractères et qui renvoie `True` si cette chaîne représenta une adresse IP correcte et `False` sinon(on pourra commencer par séparer l'ip en quatre parties à l'aide de laméthode `.split()`).

In [ ]:
```
def checkip(ip):




    pass
```

2. A l'aide de la fonction `filter` ou d'une liste en compréhension, afficher la liste des IP valides de la liste `ips` .

In [ ]:
```

```

Exercice 6:

Python dispose de deux outils principaux de tris : `.sort()` et `sorted()` . Pour répondre aux questions, il est vivement conseillé de consulter la documentation de python à ce sujet : https://docs.python.org/fr/3/howto/sorting.html (https://docs.python.org/fr/3/howto/sorting.html)

1. Laquelle de ces deux commandes relève du paradigme fonctionnel ? Justifier et donner un exemple de chaque commande

Réponse: C'est la fonction `sorted()` . En effet, cette fonction contruit une nouvelle liste triée à partir de celle passée en argument, alors que la méthode `.sort()` modifie la liste à laquelle le tri s'appplique.

In [ ]:
```
#Exemple pour sorted()



#
```

In [ ]:
```
#Exemple pour .sort()



#
```

Dans la suite, on utilisera la fonction `sorted()` pour répondre aux questions. On considère la liste ci-dessous :

In [ ]:
```
groupe=[('Pierre','M',2007),('Jeanne','F',2010),('Naomie','F',2007),
        ('Agathe','F',2012),('Ikrame','F',2012),('Jules','M',2012)]
```

2. Trier la liste en ordre alphabétique des prénoms.

In [ ]:
```

```

3. Trier la liste en ordre alphabétique décroissant.

In [ ]:
```

```

4. Trier la liste par genre ("M","F").

In [ ]:
```

```

5. Trier la liste en ordre décroissant de date de naissance.

In [ ]:
```

```

6. Trier la liste par genre puis par année de naissance.

In [ ]:
```

```

Exercice 7:

Une année s'est écoulée et la nouvelle édition de la course de module de Tatooine est encore plus captivante. Cette année, la position de chaque concurrent est stockée dans une liste, et c'est cette liste que l'on modifiera au fur et à mesure des questions. Les fonctions écrites ici ne sont donc pas des fonctions pures au sens fonctionnel et ne renvoient rien(on appelle cela des procédures).

In [ ]:
```
tatooine=['Gasgano', 'Teemto', 'Sebulba', 'Anakin']
```

Parmi les moments phares ce cette édition, il y a:

Une panne moteur fait passer le premier concurrent à la dernière position. Le second concurrent accélère et prend la tête de la course. Le dernier concurrent sauve l'honneur et dépasse l'anvant dernier module de la course. Un tir de blaster élimine le module en tête de la course. Dans un spectaculaire retournement de situation, un module qu'on pensait éliminé fait son grand retour à la dernière position.

1. Compléter la fonction `panne_moteur` , en modifiant la liste passée en argument de manière à ce que le premier module passe dernier, le deuxième premier et ainsi de suite.

In [ ]:
```
def panne_moteur(classement):
    pass



panne_moteur(tatooine)
```

In [ ]:
```
assert(tatooine==['Teemto', 'Sebulba', 'Anakin','Gasgano'])
```

2. Compléter la fonction `passe_en_tete`, modifiant la liste passée en argument de manière à ce que le premier module passe deuxième et le deuxième premier.

In [ ]:
```
def passe_en_tete(classement):
    pass



passe_en_tete(tatooine)
```

In [ ]:
```
assert(tatooine==['Sebulba', 'Teemto', 'Anakin','Gasgano'])
```

3. Compléter la fonction `sauve_honneur`, modifiant la liste passée en argument de manière à ce que le dernier module passe avant-dernier et l'avant-dernier dernier.`

In [ ]:
```
def sauve_honneur(classement):
    pass



sauve_honneur(tatooine)
```

In [ ]:
```
assert(tatooine==['Sebulba', 'Teemto','Gasgano', 'Anakin'])
```

4. Compléter la fonction `tir_blaster`, enlevant le premier concurrent de la liste passée en argument.

In [ ]:
```
def tir_blaster(classement):
    pass


tir_blaster(tatooine)
```

In [ ]:
```
assert(tatooine==['Teemto','Gasgano', 'Anakin'])
```

5. Compléter la fonction `retours_inattendus`, faisant passer le dernier concurrent à la première place et ajoutant le concurrent `'Aldar'` à la fin de la liste passée en argument.

In [ ]:
```
def retours_inattendus(classement):
    pass



retours_inattendus(tatooine)
```

In [ ]:
```
assert(tatooine==['Anakin','Teemto','Gasgano', 'Aldar'])
```