Informatique Tronc Commun - Révisions - Série 1 - Récursivité

Code de partage : e17b-1788964

A faire pour lundi 18 septembre 2023 19h maximum

Exercice 1 : Fonction d'Ackermann

Définir une **fonction ackermann** qui implémente la fonction d'Ackermann définie par :

$$A(m, n) = n+1 \quad \text{si } m = 0$$

$$A(m, n) = A(m-1, 1) \quad \text{si } m > 0 \text{ et } n = 0$$

$$A(m, n) = A(m-1, A(m, n-1)) \quad \text{si } m > 0 \text{ et } n > 0$$

Entrée[4]:

```
def ackermann(m : int, n: int )-> int :
    """
        Entrées : m, n deux entiers positifs
        Sortie : A(m,n)
    """
    # A compléter

# tests
print ( ackermann(2,3) )   # affiche 9
print ( ackermann(0,3) )   # affiche 4
print ( ackermann(3,0) )   # affiche 5
```

Exercice 2: Exponentiation rapide

Définir une **fonction puissance** en utilisant le principe de l'algorithme d'exponentiation rapide.
Pour rappel : Exponentiation rapide (https://fr.wikipedia.org/wiki/Exponentiation_rapide)

Entrée[5]:

```
def puissance( x : float , p :int ) -> int :
    """ Entrées : x, flottant
                : p, entier
        Sortie : réel qui correspond à la puissance de x par p en utilisan
    """
    # A compléter

# Tests
print ( puissance ( 2.5 , 1 ) ) # affiche 2.5
print ( puissance ( 2.5 , 3 ) ) # affiche 15.625
print ( puissance ( 2.5 , 2 ) ) # affiche 6.25
```

Exercice 3 : Palindrome

Un mot est un palindrome si on peut le lire dans les deux sens de gauche à droite et de droite à gauche. KAYAK est par exemple un palindrome.
Ecrire une **fonction estPalindromeRec récursive** permettant de vérifier si un mot est

palindrome.
Ecrire une **fonction estPalindromeIter itérative** permettant de vérifier si un mot est palindrome.

Entrée[6]:
```
def estPalindromeRec(ch):
    """
        estPalindromeRec(ch: str)-> bool (approche recursive)
        Entree : ch, chaine dont on veut savoir si c'est un palindrome
        Sortie : booléen à True si ch est un palindrome, à False sinon
    """
    # A compléter

def estPalindromeIter(ch):
    """
        estPalindromeRec(ch: str)-> bool (approche itérative)
        Entree : ch, chaine dont on veut savoir si c'est un palindrome
        Sortie : booléen à True si ch est un palindrome, à False sinon
    """
    # A compléter

# Tests
ch = "KAYAK"
print(estPalindromeRec(ch))    # affiche True
print( estPalindromeIter(ch) ) # affiche True
ch = "KAYOAK"
print(estPalindromeRec(ch))    # affiche False
print(estPalindromeIter(ch))   # affiche False
```

Exercice 4 : Conversion binaire

Définir une **fonction récursive convertirBinaire** qui prend en paramètres un entier relatif et le nombre de bits sur lequel sont codés les nombres en binaire et le convertit en binaire (complément à deux) sous la forme d'une chaîne de caractères représentant l'écriture binaire du nombre.

Entrée[7]:
```
def convertirBinaire(n, nbBits) :
    """
        Entrées : n entier relatif, nombre que l'on veut tranformer en bin
                : nbBits, entier, nombre de bits de l'écriture
        Sortie : chaine de caracteres correspondant à l'écriture binaire d
    """
    # A compléter

# Tests
print( convertirBinaire(2, 8) )  # affiche 00000010
print( convertirBinaire(-2, 8) ) # affiche 11111110
print( convertirBinaire(2, 16))  # affiche 0000000000000010
```

Exercice 5 : Calcul approché de √2

Définir les deux **fonctions u et v, mutuellement récursives,** qui calculent les nième termes des suites récurrentes croisées suivantes
u0 = 1
v0 = 2
un+1 = 2.un.vn / (un+vn)

Entrée[ ]:
```
def u(n) :
    # A compléter

def v(n) :
    # A compléter

print( u(2) ) # affiche 1.411764705882353
print( v(2) ) # affiche 1.4166666666666665
```

Exercice 6 : Tours de Hanoï appliquées à un entrepôt

Un entrepot se fait livrer des cartons chaque jour dans la zone de dépôt *zoneDepot*. Les cartons sont empilés du plus lourd, en dessous, au plus léger, au dessus. La pile doit être déplacée par un robot dans la zone de traitement *zoneArrivee* .
Le robot dispose d'une zone de transit *Transit* pour effectuer le déplacement. Le robot peut prendre uniquement le carton au sommet de la pile et ne doit jamais empiler un carton sur un carton plus léger que lui.

Définir une **fonction deplacementsHanoi** qui affiche la séquence d'instructions sous forme de chaine de caractères à envoyer au robot pour réaliser le déplacement de la pile en fonction du nombre de cartons livrés.
Le robot comprend des instructions de la forme Z1 -> Z2 où Z1 est la zone de départ du déplacement et Z2 la zone d'arrivée.

Tours de Hanoi Wikipedia (https://fr.wikipedia.org/wiki/Tours_de_Hano%C3%AF)

Entrée[ ]:
```
def deplacementsHanoi(n, zoneDepot, Transit, ZoneArrivee) :
    """
        Entrées : n, entier, nombre de cartons arrivés en ZoneDepart
        Sortie : chaine de caractères correspondant à la séquence d'instru
        les cartons de zoneDepot vers ZoneArrivee en passant par Transit
    """
    # A compléter

print( "test 1\n", instructions(1,"zoneDepot","Transit","ZoneArrivee") )
print("---")
print( "test 2\n",instructions(3,"zoneDepot","Transit","ZoneArrivee") )
print("---")
print("test 3\n", instructions(7,"zoneDepot","Transit","ZoneArrivee") )



""" Affichages attendus
test 1
zoneDepot -> ZoneArrivee

---
test 2
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
```

```
---
test 3
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
```

```
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
ZoneArrivee -> Transit
ZoneArrivee -> zoneDepot
Transit -> zoneDepot
```

```
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
zoneDepot -> Transit
ZoneArrivee -> Transit
zoneDepot -> ZoneArrivee
Transit -> zoneDepot
Transit -> ZoneArrivee
zoneDepot -> ZoneArrivee
"""
```