Correction de la partie III : IA

Question 30: Proposer un mode de communication pour que les différents appareils électriques de l'installation puissent transmettre leurs modes de fonctionnement au contrôleur. Aucun détail technique n'est attendu pour cette question

On peut utiliser un réseau Wi-Fi ou un réseau CPL (Courant Porteur de Ligne).

Question 31: Préciser les avantages et inconvénients du choix d'un réseau de neurones pour modéliser la commande

Un réseau de neurones permet de traiter un grand nombre de données et de résoudre des problèmes complexes par l'association de « neurones » relativement simples à modéliser numériquement.
Le principal inconvénient est que le réseau de neurones se comporte comme une « boîte noire » dans laquelle le cerveau humain a des diffultés à comprendre pourquoi et comment les pondérations et choix numériques ont été réalisés. Il présente toutefois l'avantage d'être peu gourmant en place mémoire par rapport à une méthode KNN par exemple.
Compte tenu du nombre de données proposées dans l'exemple de ce sujet, il semble que d'autres méthodes plus adaptées auraient pu être utilisées, car la table des données est très petite, ce qui se conformera dans la suite.

Question 32: Préciser, en justifiant la réponse, en quoi le choix d'une méthode d'apprentissage supervisé est pertinent dans ce problème

Les données sont « étiquetées », autrement dit, pour chaque quadruplet d'entrée, on sait lui associer une valeur bien définie à l'avance : couples entrée/sortie connus

Le perceptron

Question 33: Après avoir calculé la dérivée de f, proposer les fonctions f et f_prime en langage Python

$$f'(x) = \frac{(1+e^{-x})'}{(1+e^{-x})^2} = -\frac{-e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$$

exp	`import numpy as np`
$f(x) = \frac{1}{1+e^{-x}}$	`def f(x):` `return 1/(1+np.exp(-x))`

$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$	`def f_prime(x):`
	`return np.exp(-x)/(1+np.exp(-x))**2`
	`# f(x)*(1-f(x))`

Phase d'inférence

Question 34: Donner les dimensions des arrays W_2, W_3, B_2 et B_3

W_2	$f(W_1X+B_1)$ est la sortie de la première couche : 4 valeurs 4x1 $f(W_2f(W_1X+B_1)+B_2)$ est la sortie de la seconde couche : 4 valeurs 4x1 $W_2f(W_1X+B_1)$ doit donc renvoyer un vecteur de 4 valeurs 4x1	4x4
W_3	b_3 est un réel donc $W_3f(W_2f(W_1X+B_1)+B_2)$ est un réel $f(W_2f(W_1X+B_1)+B_2)$ est de dimension 4x1	1x4
B_2	$W_2f(W_1X+B_1)$ de même dimension que B_2	4x1
b_3	Biais du dernier neurone	1x1

Question 35: Donner l'expression de la matrice W_1

$$W_1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix}$$

Question 36: Proposer la fonction Python inference_couche(X,W,B) renvoyant le vecteur des sorties de la couche A

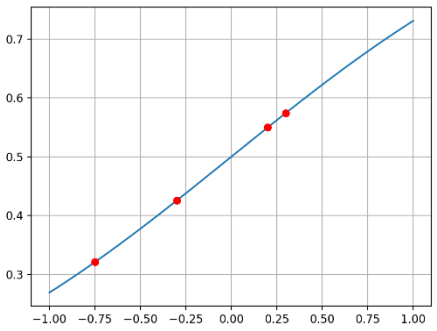
```
def inference_couche(X,W,B):
    return f(np.dot(W,X)+B)
```

Question 37: En vous aidant de la figure 18, calculer $A_1 = f(Y_1)$ dans ce cas

$$A_1 = f(Y_1) = f(W_1X+B_1)$$

$$W_1X+B_1 = \begin{bmatrix} 0,3 & 0,2 & 0,25 & -0,04 \\ 0,4 & -0,3 & 0,6 & -0,3 \\ -0,4 & 0,3 & -0,6 & -0,4 \\ -1 & -0,75 & -0,1 & -0,5 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0,2 \\ -0,3 \\ 0,3 \\ -0,75 \end{bmatrix}$$

$$A_1 = f(Y_1) = \begin{bmatrix} \frac{1}{1+e^{-0,2}} \\ \frac{1}{1+e^{0,3}} \\ \frac{1}{1+e^{-0,3}} \\ \frac{1}{1+e^{0,75}} \end{bmatrix} = \begin{bmatrix} 0,54 \\ 0,42 \\ 0,57 \\ 0,32 \end{bmatrix}$$



On peut proposer la fonction Python inference(X) renvoyant le flottant Δ associé aux 2 couches

```
def inference(X):
    A1 = inference_couche(X,W1,B1)
    A2 = inference_couche(A1,W2,B2)
    A3 = inference_couche(A2,W3,B3)
    Delta = A3[0,0]
    return Delta
```

Rétropropagation

Question 38: Montrer que $\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) x_j$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial \big(f(y_i)\big)}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial f(y_i)}{\partial y_i}\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial (y_i)}{\partial w_{ij}} f'(y_i)$$

Car

$$\frac{\partial f(y_i)}{\partial y_i} = f'(y_i)$$

Or :

$$y_i = \sum_{k=1}^{4} (w_{ik} x_k) + b_i \Rightarrow \frac{\partial (y_i)}{\partial w_{ij}} = x_j$$

Donc :

$$\boxed{\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) x_j}$$

La formule de la chaine du sujet de concours peut être utilisée, mais n'a pas grand intérêt. En effet, chaque w_{ij} n'influençant qu'un a_i. Autrement dit, tous les termes sont nuls sauf un :

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{k=1}^{4}\sum_{l=1}^{4} \frac{\partial \mathcal{L}}{\partial a_k}\frac{\partial a_k}{\partial w_{kl}} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}}$$

On peut aussi donner l'expression de $\frac{\partial \mathcal{L}_l}{\partial a_i}$ en fonction de a_i et t_l

$$\frac{\partial \mathcal{L}_l}{\partial a_i} = \frac{\partial (\Delta_l - t_l)^2}{\partial a_i}$$

A la dernière couche, on a : $\Delta_l = a_i$ (une seule sortie), soit :

$$\frac{\partial \mathcal{L}_l}{\partial a_i} = \frac{\partial (a_i - t_l)^2}{\partial a_i} = 2 * \frac{\partial (a_i - t_l)}{\partial a_i}(a_i - t_l) = 2(a_i - t_l)$$

$$\boxed{\frac{\partial \mathcal{L}_l}{\partial a_i} = 2(a_i - t_l)}$$

Puis montrer que $\frac{\partial \mathcal{L}}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i)$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial y_i}\frac{\partial y_i}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial f(y_i)}{\partial y_i}\frac{\partial y_i}{\partial b_i}$$

$$y_i = \sum_{k=1}^{4}(w_{ik}x_k) + b_i \Rightarrow \frac{\partial (y_i)}{\partial b_i} = 1$$

$$\frac{\partial f(y_i)}{\partial y_i} = f'(y_i)$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i)}$$

Remarque : chaque biais n'affecte qu'un seul neurone, il n'y a donc pas besoin de somme comme pour après avec la règle de la chaine.

On peut également montrer que l'erreur se calcule ainsi : $\frac{\partial \mathcal{L}}{\partial x_k} = \sum_{i=1}^{4} \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) w_{ik}$

$$\frac{\partial \mathcal{L}}{\partial x_k} = \sum_{i=1}^{4}\frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial a_i}{\partial x_k} = \sum_{i=1}^{4}\frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial \big(f(y_i)\big)}{\partial x_k} = \sum_{i=1}^{4}\frac{\partial \mathcal{L}}{\partial a_i}\frac{\partial (y_i)}{\partial x_k} f'(y_i)$$

Or :

$$y_i = \sum_{l=1}^{4}(w_{il}x_l) + b_i \Rightarrow \frac{\partial (y_i)}{\partial x_k} = w_{ik}$$

Donc :

$$\boxed{\frac{\partial \mathcal{L}}{\partial x_k} = \sum_{i=1}^{4}\frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) w_{ik}}$$

Question 39: Etablir les relations matricielles donnant l'expression de E_X, $\frac{\partial \mathcal{L}}{\partial W}$ et $\frac{\partial \mathcal{L}}{\partial B}$ en fonction de W, E_A, X, Y et f'

$$\frac{\partial \mathcal{L}}{\partial X} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial x_1} \\ \frac{\partial \mathcal{L}}{\partial x_2} \\ \frac{\partial \mathcal{L}}{\partial x_3} \\ \frac{\partial \mathcal{L}}{\partial x_4} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1)w_{11} + \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2)w_{21} + \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3)w_{31} + \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4)w_{41} \\ \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1)w_{12} + \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2)w_{22} + \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3)w_{32} + \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4)w_{42} \\ \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1)w_{13} + \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2)w_{23} + \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3)w_{33} + \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4)w_{43} \\ \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1)w_{14} + \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2)w_{24} + \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3)w_{34} + \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4)w_{44} \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial A} * f'(Y) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \\ \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) \\ \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) \\ \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial X} = \begin{bmatrix} w_{11} & w_{21} & w_{31} & w_{41} \\ w_{12} & w_{22} & w_{32} & w_{42} \\ w_{13} & w_{23} & w_{33} & w_{43} \\ w_{14} & w_{24} & w_{34} & w_{44} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \\ \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) \\ \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) \\ \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) \end{bmatrix}$$

$$E_X = \frac{\partial \mathcal{L}}{\partial X} = W^T . [E_A * f'(Y)]$$

Il y a un petit problème de mise en forme de cette formule dans le sujet de concours.

$$\frac{\partial \mathcal{L}}{\partial W} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{11}} & \frac{\partial \mathcal{L}}{\partial w_{12}} & \frac{\partial \mathcal{L}}{\partial w_{13}} & \frac{\partial \mathcal{L}}{\partial w_{14}} \\ \frac{\partial \mathcal{L}}{\partial w_{21}} & \frac{\partial \mathcal{L}}{\partial w_{22}} & \frac{\partial \mathcal{L}}{\partial w_{23}} & \frac{\partial \mathcal{L}}{\partial w_{24}} \\ \frac{\partial \mathcal{L}}{\partial w_{31}} & \frac{\partial \mathcal{L}}{\partial w_{32}} & \frac{\partial \mathcal{L}}{\partial w_{33}} & \frac{\partial \mathcal{L}}{\partial w_{34}} \\ \frac{\partial \mathcal{L}}{\partial w_{41}} & \frac{\partial \mathcal{L}}{\partial w_{42}} & \frac{\partial \mathcal{L}}{\partial w_{43}} & \frac{\partial \mathcal{L}}{\partial w_{44}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) x_1 & \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) x_2 & \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) x_3 & \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) x_4 \\ \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) x_1 & \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) x_2 & \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) x_3 & \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) x_4 \\ \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) x_1 & \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) x_2 & \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) x_3 & \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) x_3 \\ \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) x_1 & \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) x_2 & \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) x_3 & \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) x_4 \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial W} = [E_A * f'(Y)] . X^T$$

$$\frac{\partial \mathcal{L}}{\partial B} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial b_1} \\ \frac{\partial \mathcal{L}}{\partial b_2} \\ \frac{\partial \mathcal{L}}{\partial b_3} \\ \frac{\partial \mathcal{L}}{\partial b_4} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \\ \frac{\partial \mathcal{L}}{\partial a_2} f'(y_2) \\ \frac{\partial \mathcal{L}}{\partial a_3} f'(y_3) \\ \frac{\partial \mathcal{L}}{\partial a_4} f'(y_4) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} \\ \frac{\partial \mathcal{L}}{\partial a_2} \\ \frac{\partial \mathcal{L}}{\partial a_3} \\ \frac{\partial \mathcal{L}}{\partial a_4} \end{bmatrix} * \begin{bmatrix} f'(y_1) \\ f'(y_2) \\ f'(y_3) \\ f'(y_4) \end{bmatrix}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial B} = E_A * f'(Y)}$$

Question 40: Ecrire la fonction Python retropropagation_couche(X,W,B,Ea,alpha) attendue

```
def retropropagation_couche(X,W,B,EA,alpha):
    Y = np.dot(W,X) + B
    fp = f_prime(Y)
    EW = np.dot(EA*fp,X.T)
    W -= alpha*EW # Modification en place
    EB = EA*fp
    B -= alpha*EB # Modification en place
    EX = np.dot(W.T,EA*fp)
    return EX
```

Voici la fonction epoch_ligne(X,Deltaopt,alpha) réalisant la procédure epoch sur une ligne des entrées X connaissant la valeur Deltaopt de la ligne associée

```
def epoch_ligne(X,Deltaopt,alpha):
    A1 = inference_couche(X,W1,B1)
    A2 = inference_couche(A1,W2,B2)
    A3 = inference_couche(A2,W3,B3) # = Delta
    EX3 = 2*(A3-Deltaopt) # Dérivée de l'erreur
    EX2 = retropropagation_couche(A2,W3,B3,EX3,alpha) # 3° couche
    EX1 = retropropagation_couche(A1,W2,B2,EX2,alpha) # 2° couche
    EX0 = retropropagation_couche(X,W1,B1,EX1,alpha) # 1° couche
```

La fonction epoch(LD,alpha) réalisant un epoch sur toutes les données d'apprentissage (on pensera à transformer l'array des données avec reshape)

```
def epoch(LD,alpha):
    N = len(LD)
    for i in range(N):
        X = LD[i,:4].reshape(-1,1) # Vertical
        Deltaopt = LD[i,4]
        epoch_ligne(X,Deltaopt,alpha)
```

La fonction erreur_totale(LD) calculant et renvoyant l'erreur quadratique moyenne totale après un epoch

```
def erreur_totale(LD):
    N = len(LD)
    S = 0
    for i in range(N):
        X = LD[i,:4].reshape(-1,1) # Vertical
```

```
        Deltaopt = LD[i,4]
        Delta = inference(X)
        Err = Deltaopt - Delta
        S += Err**2
    Err = S/N
    return Err
```

La fonction entrainement(LD,Nit,alpha) réalisant Nit epoch et renvoyant les listes Lit du numéro de l'itération et Lerr de l'erreur totale associée

```
def entrainement(LD,Nit,alpha):
    Lit = []
    Lerr = []
    for i in range(Nit):
        Lit.append(i)
        epoch(Donnees_A,alpha)
        err = erreur_totale(Donnees_A)
        Lerr.append(err)
    return Lit,Lerr
```

Pour 100 epoch de différentes valeurs de alpha, on trace l'erreur totale en fonction du nombre d'itérations réalisées

```
from matplotlib import pyplot as plt

def Affiche(fig,X,Y,type):
    plt.figure(fig)
    plt.plot(X,Y,type)
    plt.grid(True)
    plt.show()
    plt.pause(0.01)

alpha = 0.01
Nit = 100
Lit,Lerr = entrainement(Donnees_A,Nit,alpha)
Affiche(4,Lit,Lerr,'-')
```

Analyse des résultats

Question 41: Analyser les résultats. Préciser si ce réseau de neurones est capable de bien prédire la valeur optimale de Δ

4 des 5 valeurs sont correctes, mais pour l'avant dernière ligne, le réseau de neurones donne $\Delta = 0{,}015$ pour une valeur attendue de $0{,}22$, ce qui est loin d'être satisfaisant. Autrement dit, 20% des prédictions sur le set de tests sont loin de la bonne valeur, 80% sont plutôt bonnes. Le réseau de neurones ainsi mis en place n'est donc pas parfait !

Question 42: Pour améliorer le modèle, préciser en justifiant la réponse, s'il est préférable d'augmenter le nombre d'itérations de l'entraînement ou d'augmenter le nombre de données test

La figure 19 montre que le nombre d'itérations ne changera rien. A partir de 15 itérations, l'erreur ne diminue quasiment plus.

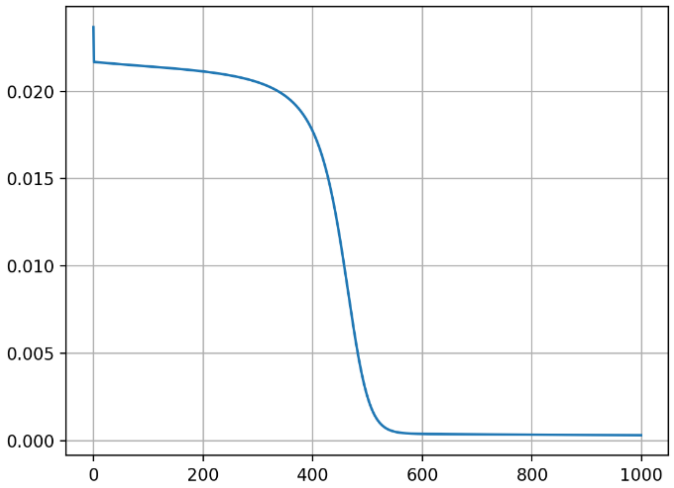
Il faut grandement augmenter le nombre de données d'entrainement ! Et aussi de test tant qu'à faire, histoire de pouvoir valider le modèle sur plus de données.

On peut entrainer et vérifier le réseau créé précédemment sur 1000 lignes de ces données et vérifier qu'il converge en moins de 1000 itérations (alpha=0.01)

```
from random import random as rd
def cree_dataset(N):
    dataset = []
    for i in range(N):
        x1,x2,x3,x4 = rd(),rd(),rd(),rd()
        d = (x1+x2+x3+x4)/4
        data = [x1,x2,x3,x4,d]
        dataset.append(data)
    return np.array(dataset)

N = 1000
Ind = int((75/100)*N)
Donnees = cree_dataset(N)
Donnees_A = Donnees[:Ind]
Donnees_T = Donnees[Ind:]

W1 = np.random.rand(4,4)
W2 = np.random.rand(4,4)
W3 = np.random.rand(1,4)
B1 = np.random.rand(4,1)
B2 = np.random.rand(4,1)
B3 = np.random.rand(1,1)
```




```
alpha = 0.01
Nit = 1000
Lit,Lerr = entrainement(Donnees_A,Nit,alpha)
Affiche(10,Lit,Lerr,'-')

def reconstruction(LD):
    '''Renvoie la table des données avec une colonne supplémentaire
donnant la valeur de delta obtenue avec le réseau'''
    Nl,Nc = LD.shape
    Res = np.zeros([Nl,Nc+1])
    Res[:,:Nc] = LD
    for i in range(Nl):
        D = LD[i]
        X = D[:4].reshape(-1,1)
        Delta = inference(X)
        Res[i,Nc] = Delta
    return Res

Deltaopt_Rec = reconstruction(Donnees)
print(Deltaopt_Rec)

'''
En théorie, on trouve la droite y=x
Je trace à la fois les données d'apprentissage et de test
'''
Lx = np.linspace(0,1,2)
Affiche(11,Lx,Lx,'--')
Lx = Deltaopt_Rec[:,4]
Ly = Deltaopt_Rec[:,5]
Affiche(11,Lx,Ly,'o')
```