TD Info

Des chiffres et Des lettres

1. Création du jeu

1.1. Le compte est bon

```python
def tirageChiffres():
    """ne prend pas d'argument,
    renvoie une liste tirée de 6 nombres tirés aléatoirement dans la liste L
    On place en fin de liste un nombre tiré aléatoirement entre 100 et 999"""
    L=[1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10,25,50,75,100] #liste des chiffres
    tirage=[]
    N=6
    for k in range(N): # tire au sort 6 chiffres parmi les 14
        tirage.append(L[randint(0,len(L)-1)]) #randint(inclus, inclus) dans ma version python
        L.remove(tirage[-1])
    tirage.append(randint(100,999)) # tire le nombre à trouver
    return tirage # renvoie la liste des 6 + compte
```

1.2. Le mot le plus long

```python
def tirageLettres():
    """fonction sans argument qui demande à l'utilisateur de 'c' ou 'v'
    et qui crée un liste contenant les lettres associées tirées aléatoirement."""
    listeConsonne=['b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w',
    listeVoyelle=['a','e','i','o','u','y']
    tirage=[]
    N=10
    while len(tirage)<=N:
        demande=input("pour consonne, tapez c, pour voyelle v, tapez v ")
        if demande=='c':
            tirage.append(listeConsonne[randint(0,len(listeConsonne)-1)])

        elif demande=='v':
            tirage.append(listeVoyelle[randint(0,len(listeVoyelle)-1)])

        else:
            print("taper c ou v uniquement")
    return tirage
```

```python
def verification (mot):
    #mot= mot + "\n" #ajoute le saut de ligne à la fin du mot pour s'assurer que le mot trouvé est bien
    fichier =open("dictionnaire_trie.txt")
    dico=fichier.read()
    Lsdico = dico.split('\n')

    if mot in Lsdico:
        print(mot+" est bien dans le dictionnaire ")
        #print("{} est bien dans le dictionnaire ".format(mot[:-1]))
    else:
        print("Tu n'essayerais pas de tricher par hasard?")
    fichier.close()
```

Python

2. Création de l'intelligence artificielle

2.1. Le compte est bon

```python
def addition(a,b):
    return a+b

def multiplication(a,b):
    return a*b

def soustraction(a,b):
    if a>b:
        return a-b
    else:
        return b-a


def division(a,b):
    if a%b==0:
        return a//b
    elif b%a==0:
        return b//a
    else:
        return 0
```

Python

```python
def solucChiffres(tirage, limite=300000):
    """prend en arguments un tirage (liste de nombres) et un nombre limite d'essais (entier)
    ne renvoie rien."""
    chaine="" #servira à l'affichage
    compte=tirage[-1] #compte à trouver
    liste=copy.deepcopy(tirage[:-1]) #copie en dur du tirage
    listeOperation=[] # sert à inscrire les opérations au fur et à mesure
    copyListeOperation=[] # sert de copie
    compteApproche=tirage[5] # on fixe arbitrairement la valeur du compte approché provisoire
    compteur=0 #compte le nombre d'opérations pour limiter le temps de calcul
    while liste[-1]!=compte and compteur<limite: # tant que le compte n'est pas trouvé
                                                 # ou tant qu'on a pas atteint la limite
        compteur=compteur+1
        if len(liste)==1 or liste[-1]==0: # s'il n'y a plus qu'un seul élément ou si la division
                                          # ou la soustraction n'a pas abouti, on réinitialise
            liste=copy.deepcopy(tirage[:-1])
            listeOperation=[]

        operation=randint(0,4) #on tire au sort l'opération
        chiffre1=liste[randint(0,len(liste)-1)] #on tire au sort le 1er chiffre
        liste.remove(chiffre1) # on enleve ce chiffre de la liste
        chiffre2=liste[randint(0,len(liste)-1)] #on tire au sort le 2nd chiffre
        liste.remove(chiffre2)

        if operation==0:
            liste.append(addition(chiffre1,chiffre2))
            listeOperation.append(chiffre1)
            listeOperation.append("+")
            listeOperation.append(chiffre2)
            listeOperation.append("=")
            listeOperation.append(addition(chiffre1,chiffre2))
            listeOperation.append("\n")

        elif operation==1:
            liste.append(multiplication(chiffre1,chiffre2))
            listeOperation.append(chiffre1)
            listeOperation.append("x")
            listeOperation.append(chiffre2)
            listeOperation.append("=")
            listeOperation.append(multiplication(chiffre1,chiffre2))
            listeOperation.append("\n")

        elif operation==2:
            liste.append(soustraction(chiffre1,chiffre2))
            if chiffre1>chiffre2:
                listeOperation.append(chiffre1)
                listeOperation.append("-")
                listeOperation.append(chiffre2)
                listeOperation.append("=")
                listeOperation.append(soustraction(chiffre1,chiffre2))
            else:
                listeOperation.append(chiffre2)
```

Python

```python
                    listeOperation.append("-")
                    listeOperation.append(chiffre1)
                    listeOperation.append("=")
                    listeOperation.append(soustraction(chiffre1, chiffre2))
                listeOperation.append("\n")

            else:
                liste.append(division(chiffre1, chiffre2))
                if chiffre1>chiffre2:
                    listeOperation.append(chiffre1)
                    listeOperation.append("/")
                    listeOperation.append(chiffre2)
                    listeOperation.append("=")
                    listeOperation.append(division(chiffre1, chiffre2))
                else:
                    listeOperation.append(chiffre2)
                    listeOperation.append("/")
                    listeOperation.append(chiffre1)
                    listeOperation.append("=")
                    listeOperation.append(division(chiffre1, chiffre2))
                listeOperation.append("\n")

            if abs(liste[-1]-compte)<abs(compteApproche-compte): #si on s'est rapproché du compte
                compteApproche = liste[-1]
                copyListeOperation=copy.deepcopy(listeOperation) #sauvegarde en dur de la liste d'opération

        if liste[-1]==compte:
            print("le compte est bon")
            copyListeOperation.append("=")
            copyListeOperation.append(compte)
            for k in copyListeOperation: # transforme la liste en chaine de caracteres
                chaine=chaine+str(k)
            print(chaine)
        else:
            print("le compte est approché")
            copyListeOperation.append("=")
            copyListeOperation.append(compteApproche)
            for k in copyListeOperation: # transforme la liste en chaine de caracteres
                chaine=chaine+str(k)
            print(chaine)
```

Python

2.2. Le mot le plus long

```python
def nouvDico():
    """fonction sans argument. Permet de créer un nouveau fichier txt dans lequel on place les mots du
     dictionnaire du plus long au plus court ne renvoie rien"""
    g=open("dictionnaire_trie.txt","a")
    f=open("dictionnaire.txt","r")
    dico=f.read()
```

Python

Python

```
    Lsdico = dico.split('\n')
    for n in range(10,0,-1): #de 10 à 1 en décroissant
        for mot in Lsdico:
            if len(mot)==n:
                g.write(mot+'\n') #ne pas oublier le saut de ligne
        f.close()
    g.close()
```

Python

```
def solucLettre(listeTirage, tailleMot=10, nbreMotAAffiche=10):
    """prend en argument un tirage (liste de caractères)
    l'option tailleMot permet de rechercher des mots de taille non optimisée
    renvoie une liste contenant les mots les plus longs présents dans le dictionnaire """
    f=open("dictionnaire_trie.txt","r")
    dico=f.read()
    Lsdico = dico.split('\n')
    nbreMotTrouve=0
    motTrouve=""
    ind=0
    motDico=Lsdico[ind]
    Ls_mots=[]

    while nbreMotTrouve<nbreMotAAffiche and len(motDico)>0 : #tant que tous les mots n'ont pas été trou
        if len(motDico)<=tailleMot: # utilisé uniquement avec l'option tailleMot
            motTrouve="" # réinitialisation de la variable
            booleen=True # réinitialisation du booleen
            positionLettre=0 #indice qui s'incrémente à chaque fois qu'on se déplace dans le mot

            while booleen==True: #tant que cela convient

                if len(motDico)==positionLettre+1:
                    nbreMotTrouve=nbreMotTrouve+1 # c'est un bon mot
                    booleen=False #pour sortir de la boucle while
                    Ls_mots+=[motDico]
                    print(motDico)

                elif motDico.count(motDico[positionLettre])>listeTirage.count(motDico[positionLettre]) :
#on compare l'occurence des lettres du tirage avec celui du mot
                    booleen=False #le mot ne convient pas, il faut sortir de la boucle
                positionLettre+=1
        ind+=1
        motDico=Lsdico[ind]
    f.close()
```