Epreuve d'Informatique - 18 janvier 2023 - 1h

Ecrire directement sur les feuilles de l'énoncé.

Rendre toutes les feuilles (même vierges) avec votre nom dessus.

Documents autorisés : notes de cours et T.D., polys de cours.
Vous veillerez à soigner la présentation des programmes et à choisir des noms de variables « significatifs ».
Le nombre de lignes précisé permet d'avoir une idée de la complexité du programme demandé, mais il n'est qu'approximatif et peut varier d'une solution à l'autre. Idem pour le temps indiqué qui doit vous aider à gérer le temps passé sur chaque exercice.

Exercice I - Syntaxe

Compétences évaluées : AF2 Compétences en programmation dans le langage de programmation Python

A partir du script ci-dessous (qu'on ne vous demande pas de comprendre), répondez aux questions suivantes (*environ 15 minutes*) :

```
9   def tour_de_jeu(plateau, mot, lettre):
10      """
11
12      @param plateau:       lettres déjà devinées
13      @param mot:          mot à deviner
14      @param lettre:       lettre proposée au tour de jeu
15      @return:             mot modifié
16      """
17      if lettre not in mot[1: -1]:
18          print(lettre, "n'apparaît pas dans le mot à trouver")
19      else:
20          for i in range(1, len(mot) - 1):
21              if mot[i] == lettre:
22                  plateau[i] = lettre
23
24  if __name__ == "__main__":
25      mot = input("mot : ")
26      mot = list(mot)
27      pendu = ["_"] * len(mot)
28      pendu[0] = mot[0]
29      pendu[-1] = mot[-1]
30      historique = [pendu]
31
32      for tour in range(8):
33          proposition = input("proposition de lettre : ")
34          tour_de_jeu(pendu, mot, proposition, tour)
35          if pendu == mot:
36              break
37
38          print(pendu)
39          historique.append(pendu)
40
41      if pendu == mot:
42          print("Gagné")
43      else:
44          print("Perdu - le mot à trouver était : ", mot)
```

Epreuve d'Informatique - 18 janvier 2023 - 1h

Question 1 - Sur quelles lignes se situe la *docstring* de la fonction *tour_de_jeu* ?

Question 2 - Que signifie cette instruction : *mot[1: -1]* ? Que donne-t-elle si *mot = "voiture"*

Question 3 - A la ligne 37 on pourrait s'attendre à avoir un *else* comme ci-dessous. Pourquoi n'est-ce pas nécessaire ?

```
32      for tour in range(8):
33          proposition = input("proposition de lettre : ")
34          tour_de_jeu(pendu, mot, proposition, tour)
35          if pendu == mot:
36              break
37
38          print(pendu)
39          historique.append(pendu)
40
41      if pendu == mot:
42          print("Gagné")
43      else:
44          print("Perdu - le mot à trouver était : ", mot)
45
```

Question 4 - Comment modifier l'instruction *for* pour que la variable *tour* prenne les valeurs de 1 à 8 inclus.

Question 5 - La fonction *tour_de_jeu* ne retourne rien alors qu'elle modifie la variable *plateau* ? Est-ce normal ? Pourquoi ?

Question 6 - Quand on lance l'exécution de ce programme, on obtient l'erreur suivante. Sur quelle ligne a-t-elle été détectée ? Comment la corriger ? Est-ce qu'on aurait pu la détecter avant l'exécution du programme ?

```
  File "C:\Users\fay7\Nextcloud\1 - ENSG1A\ENSG1A-2022-2023\Semestre 5\Evaluations\Corrections\Pythonesquerie.py", line 34, in <module>
    tour_de_jeu(Pendu, mot, proposition, tour)
TypeError: tour_de_jeu() takes 3 positional arguments but 4 were given
```

Exercice II - Adresse IP

Compétences évaluées :
AF 1 - Compétences méthodologiques et de conceptualisation (algorithmique)

Une adresse IP est définie par quatre nombres compris entre 0 et 255, séparés par des points.

Exemple : 212.85.150.134

Une adresse IP est correctement construite si :

- elle comporte au moins **4** séries de chiffres séparées par des **points** (".") ;
- les caractères situés entre les points sont des chiffres compris entre 0 et 255.

1) Ecrire une fonction ***IPvalide*** qui retourne *True* si une adresse (chaîne de caractères) fournie en paramètre définit une adresse IP correcte et *False* sinon. On suppose que l'adresse fournie en paramètre n'est constituée que de chiffres et de points. (*environ 8 lignes – 10 minutes*)

Indications : on peut utiliser les fonctions :

- *ch.split(sep)* découpe *ch* et retourne une liste de sous-chaînes de caractères en utilisant sep comme caractère séparateur
  exemple : *adresse = "212.85.150.134"*
  phrase.split(".") → *["212", "85", "150", "134"]*
- *ch.isdigit()* renvoie *True* si tous les caractères de la chaîne ch sont des chiffres et qu'elle contient au moins un caractère, *False* sinon.

exemple :	*chaine = "128"*	*chaine.isdigit() retourne True*
	chaine = "A276"	*chaine.isdigit() retourne False*

2) Une table DNS permet d'associer un nom de serveur à une adresse IP. Cette table est structurée sous forme d'un dictionnaire tel que :

DNS[serveur] = adresseIP

Exemple : DNS = {"www.univ-lorraine.fr": "193.50.135.38,

"www.fnac.com": "96.16.248.133" ...}

Ecrire une fonction *ajoute_serveur* qui ajoute un nom de serveur et l'adresse IP lui correspondant dans la table DNS.

Cette fonction reçoit en paramètre la table DNS, le nom du serveur et l'adresse IP.

Avant d'ajouter cette adresse dans la table, il convient de vérifier que ni le serveur, ni l'adresse IP n'existent déjà dans la table DNS. Si c'est le cas, la table n'est pas modifiée et un message d'erreur est affiché. (*environ 8 lignes – 6 minutes*)

Avant d'ajouter cette adresse dans la table, il convient de vérifier que :

- l'adresse IP est valide
- et ni le serveur, ni l'adresse IP n'existent déjà dans la table DNS.

Si c'est le cas, la table n'est pas modifiée et un message d'erreur est affiché. (*environ 10 lignes – 6 minutes*)

Exercice III - Suite de Héron

Compétences évaluées :
AF 1 - Compétences méthodologiques et de conceptualisation (algorithmique)

Considérant *a* un réel strictement plus grand que 1, on définit la suite de Héron ainsi :

$$u_0 = terme\ quelconque > 0$$
$$u_i = \frac{1}{2}\left(u_{i-1} + \frac{a}{u_{i-1}}\right)$$

Cette suite converge vers $\sqrt{a}$.

Ecrire une fonction ***suiteU*** qui, pour *a* et *epsilon* donnés en paramètres retourne le premier *n* qui permette d'atteindre une valeur de la suite proche de $\sqrt{a}$ à *epsilon* près. (*7 lignes environ – 8 minutes*).

Exercice IV - Réseau

Compétences évaluées :
AF 1 - Compétences méthodologiques et de conceptualisation (algorithmique)
AF 2- Compétences en programmation dans le langage de programmation Python

1) Pour décrire les composants d'un réseau quelconque, on dispose pour chaque élément *a* de ce réseau d'une liste *connexions* telle que :

connexion[i] = 1 si l'élément *i* est en contact avec l'élément *a*

= 0 sinon

a) Ecrire une fonction *mes_contacts* qui construit la liste des éléments (indices) en contact avec l'élément considéré. La liste des contacts sous la forme de 0 et 1 est fournie en paramètre. (*environ 6 lignes – 8 minutes*)

Exemple : si connexion = [1, 0, 0, 1, 0, 1]

mes_contacts(connexion) retourne la liste [0, 3, 5]

b) Réécrire la fonction précédente à l'aide d'une liste en compréhension (2 lignes).

2) L'ensemble du réseau est décrit dans une matrice telle que :

reseau[a][b] = 1 si et seulement si a est en relation avec b

reseau[a][b] = 0 sinon

Exemple pour une relation sur 6 valeurs.

reseau[1][3] = 1 cela signifie que 1 est en relation avec 3

1	*1*	*1*	*1*	*1*	*1*
0	*1*	*0*	*1*	*0*	*1*
0	*0*	*1*	*0*	*0*	*1*
0	*0*	*0*	*1*	*0*	*0*
0	*1*	*0*	*0*	*1*	*0*
0	*1*	*0*	*0*	*0*	*1*

Ecrire la **fonction** *est_symétrique* qui permet de vérifier que la matrice *reseau* passée en paramètre est bien symétrique. Cette fonction retourne *True* si la matrice est symétrique et *False* sinon (*environ 8 lignes – 8 minutes*)

$\mathcal{R}$ est symétrique si $\forall a, b$ si a est en relation avec b alors b est en relation avec a