EXAMENS ALGORITHMIQUE ET PROGRAMMATION

2022-2023 semestre 5

Syntaxe

```
def tour_de_jeu(plateau, mot, lettre):
    """

    @param plateau:      lettres déjà devinées
    @param mot:          mot à deviner
    @param lettre:       lettre proposée au tour de jeu
    @return:             mot modifié
    """
```

docstring

```
  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
```

```
for tour in range(8):
    proposition = input("proposition de lettre : ")
    tour_de_jeu(pendu, mot, proposition, tour)
    if pendu == mot:
```

Alerte !!

```
tour_de_jeu
f tour_de_jeu(plateau, mot, lettre)
Press Entrée to insert, Tab to replace  Next Tip
```

Mais aussi aide à la saisie

UNIVERSITÉ DE LORRAINE
LORRAINE INP

Syntaxe

```
mot[1: -1]
```

sous-chaîne de l'indice 1 jusqu'à l'indice -1 (dernier) **exclu**

mot = "voiture" mot[1: -1] → "oitur"

```
for i in range(1, 9):
```

Instruction *for* pour que la variable *tour* prenne les valeurs de **1 à 8 inclus**.

```
for i in range(9):
```

☹

```
for i in range(1, 8):
```

☹

```
for i in range(1, 9, 1):
```

😐

UNIVERSITÉ DE LORRAINE
LORRAINE INP

Syntaxe

```
31      for tour in range(8):
32          proposition = input("proposition de lettre : ")
33          tour_de_jeu(pendu, mot, proposition, tour)
34
35          if pendu == mot:
36              break
37          else:
38              print(pendu)
39              historique.append(pendu)
```

Exécutées si pendu == mot

Exécutées si pendu != mot

```
    for tour in range(8):
        proposition = input("proposition de lettre : ")
        tour_de_jeu(pendu, mot, proposition, tour)

        if pendu == mot:
            break

        print(pendu)
        historique.append(pendu)
```

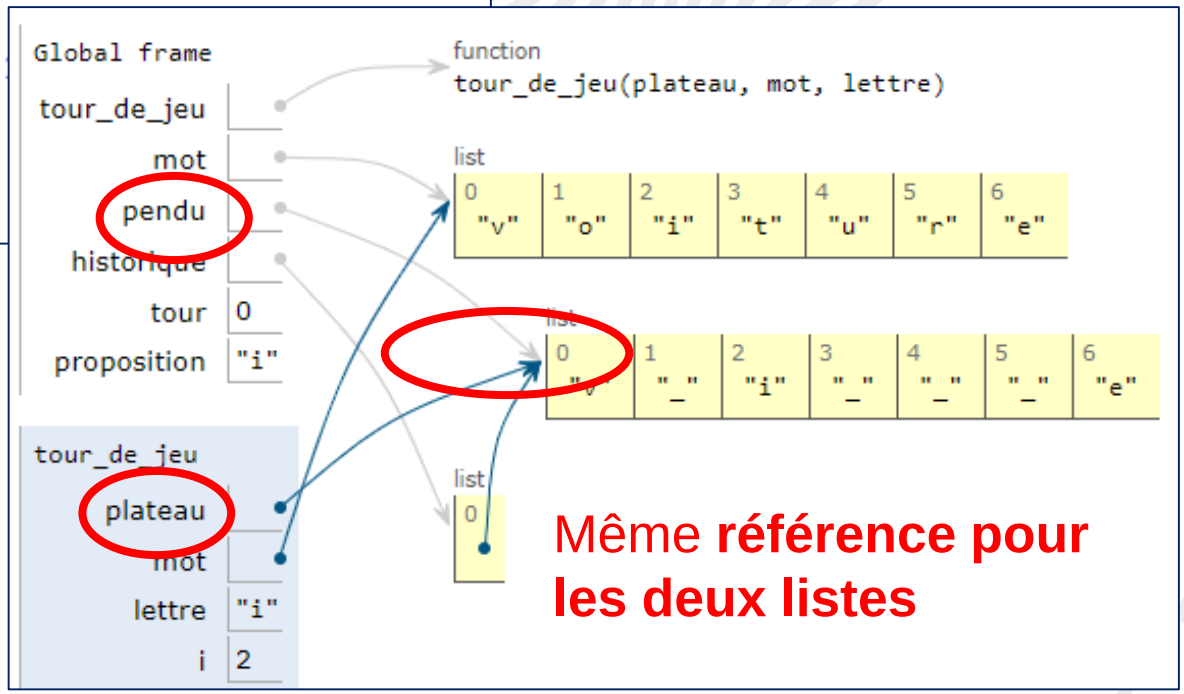
si pendu == mot
on sort alors de l'itération

Exécutées si on ne sort pas de l'itération
donc si pendu != mot

Syntaxe

```
def tour_de_jeu(plateau, mot, lettre):
    """..."""
    if lettre not in mot[1: -1]:
        print(lettre, "n'apparaît pas dans le mot à trouver")
    else:
        for i in range(1, len(mot) -
            if mot[i] == lettre:
                plateau[i] = lettre
```

Passage de *plateau* en paramètre par **référence**



Même **référence pour les deux listes**

- ***pendu* change de valeur à chaque tour de jeu**
- **Puisque c'est une liste qui est passée en paramètre par référence**

```
for tour in range(8):
    proposition = input("proposition de lettre : ")
    tour_de_jeu(pendu, mot, proposition)
```

Adresse IP


```python
def IPvalide(adresse):
    """..."""

    # on découpe la chaîne en prenant le caractère "." comme séparateur
    lchiffres = adresse.split(".")
    # on vérifie que l'adresse contient bien 4 chiffres
    if len(lchiffres)!=4 :
        return False

    # on vérifie qu'on n'a que des chiffres compris
    for elt in lchiffres:
        if not elt.isdigit():
            return False
        if elt<0 or elt>255:
            return False

    # dans tous les autres cas, on retourne True
    return True
```

```python
print(IPvalide("212.85.150.134"))
print(IPvalide("212.85.150.134."))
print(IPvalide("212.85.150"))
print(IPvalide("212.A85.150"))
```

Vérification syntaxe Adresse IP


```
def IPvalide(adresse):
    """..."""

    # on découpe la chaîne en prenant le caractère "." comme séparateur
    lchiffres = adresse.split(".")
```

ch.split(".")

adresse.split(.)

```
    for elt in lchiffres:
        if not elt.isdigit():
            return False
        else:
            return True
```

```
    if int(elt)<0 or int(elt)>255:
        return False
    else:
        return True
```

```
    if 0 <= int(elt) <= 255:
        return True
    else:
        return False
```

if not elt.isdigit:

UNIVERSITÉ DE LORRAINE

LORRAINE INP

```
if not elt.isdigit() or int(elt)<0 or int(elt)>255:
    return False
```

```
if int(elt)<0 or int(elt)>255 or not elt.isdigit():
    return False
```

Tester si c'est un entier avant de tester l'intervalle de valeurs

```
for i in range(len(lchiffres)):
    elt = lchiffres[i]
```

```
for elt in lchiffres:
```

Serveur DNS


```python
def ajoute_serveur(DNS, serveur, adresse):
    if not IPvalide(adresse):
        print("Adresse invalide")
        return

    if serveur in DNS:
        print("nom de seveur existant")
        return

    if adresse in DNS.values():
        print("Adresse existante")
        return

    DNS[serveur] = adresse
```

```python
if serveur in DNS.keys():
    print("nom de seveur existant")
    return
```

```python
for serveur, ad in DNS.items():
    if ad==adresse:
        print("Adresse existante")
        return
```

Return DNS

Suite de Héron


```python
def suiteU(a, epsilon):
    """..."""

    if a <= 1:
        return None

    u = 1
    n = 0

    r = a**0.5 # Evite de faire plusieurs fois le calcul

    while abs(u - r) > epsilon:

        u = 0.5*(u + a/u)
        n += 1

    return n
```

Initialisation avec paramètre de la fonction, randint, etc.

Suite de Héron


```
while u < r - epsilon or u > r + epsilon:
```

```
    return u
    print(n)
    return u, n
```

Réseau


```python
def mes_contacts(connexion):
    """..."""

    contact = []

    for i in range(len(connexion)):

        if connexion[i] == 1:
            contact.append(i)

    return contact
```

Ici, on a besoin de l'indice (c'est ce qui est sauvegardé dans contact) !

```python
contact += [i]
```

☹

Réseau


```
def mes_contacts(connexion):
    """..."""

    contact = []
    compteur = 0

    for elt in connexion:

        if elt == 1:
            contact.append(compteur)

        compteur += 1

    return contact
```

Réseau


```
for elt in connexion:

    if elt == 1:
        contact.append(connexion[elt])
```

```
for elt in connexion:

    if elt == 1:
        contact.append(connexion.index(elt))
```

```
for elt in connexion:

    if elt == 1:
        contact.append(elt)
```

Réseau

Liste en compréhension

```
[i for i in range(len(connexion)) if connexion[i] == 1]
```

Réseau

Réseau[i][j]	Réseau[j][i]	Symétrique ?
1	1	✓
1	0	✗
0	1	✗
0	0	✓

Si le réseau n'est pas carré, il ne peut être symétrique (pire, on peut avoir réseau[i][j] défini et réseau[j][i] qui n'existe pas !)

Réseau


```python
def est_symetrique(reseau):
    """
    Indique si la matrice réseau est symétrique.
    :param reseau: matrice de 0 et de 1
    :return: booléen
    """

    n = len(reseau)

    if n != len(reseau[0]):
        return False

    for i in range(n):
        for j in range(n):

            if reseau[i][j] != reseau[j][i]:
                return False

    return True
```

Réseau


```python
for i in range(n):
    for j in range(n):

        if reseau[i][j] != reseau[j][i]:
            return False

        else:
            return True
```

```python
for i in range(n):
    for j in range(n):

        if reseau[i][j] == reseau[j][i]:
            return True
```

Réseau

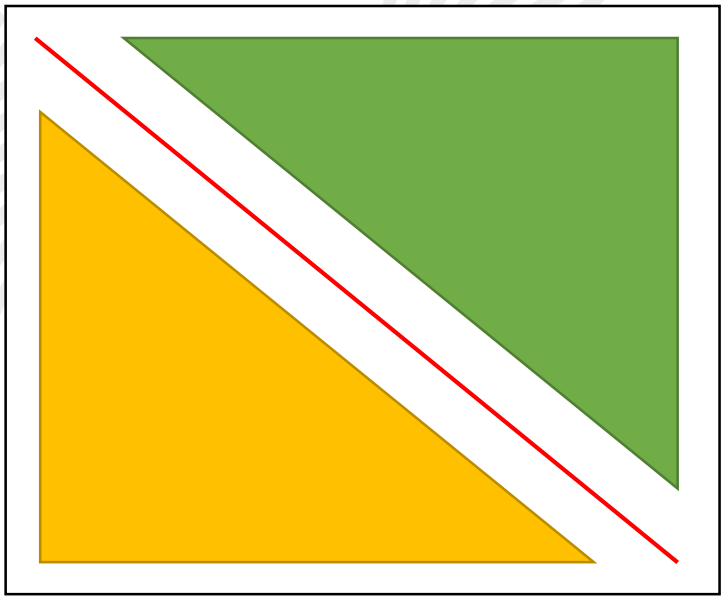

```
for ligne in reseau:
    for colonne in ligne:

        if reseau[ligne][colonne] != reseau[colonne][ligne]:
            return False
```

Réseau

Optimisation :

- Il n'est pas utile de vérifier les éléments diagonaux (i = j)
- On évite de tester les éléments 2x
- => Tester seulement le triangle supérieur (ou le triangle inférieur)



Réseau


```python
def est_symetrique(reseau):
    """..."""

    n = len(reseau)

    if n != len(reseau[0]):
        return False

    for i in range(1, n):
        for j in range(i):

            if reseau[i][j] != reseau[j][i]:
                return False

    return True
```