Jeu du pendu [tb06] - Examen

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 19 mai 2018

Table des matières

Python - Jeu du pendu (Solution)

Mots-Clés Tableau unidimensionnel, Jeu ■
Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Structures répétitives, Schéma itératif ■
Difficulté ● ● ○

Objectif
Cet exercice réalise le jeu du pendu.

...(énoncé page suivante)...

1 Jeu du pendu / pgjeupendu (20 points)

1.1 Présentation du jeu

Jeu du pendu
Il se joue à deux. L'un pense à un mot, par exemple :

```
L I C E N C E
```

et écrit une suite de tirets à la place des lettres :

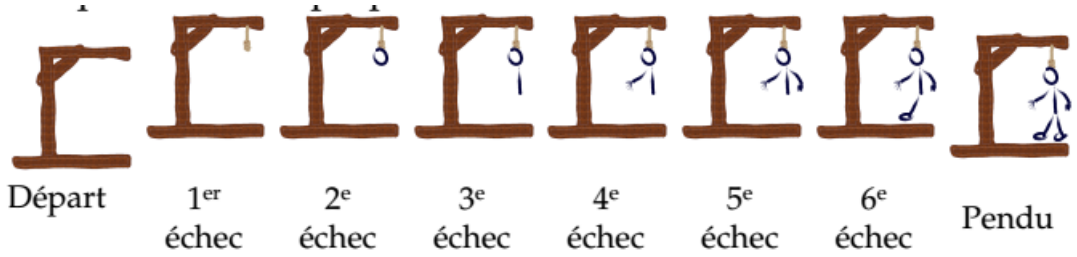
```
_ _ _ _ _ _ _
```

L'autre doit deviner ce mot en ne connaissant au départ que sa longueur. Pour s'aider, il peut demander l'existence d'une lettre. À chaque lettre proposée, si celle-ci figure dans le mot, les occurrences de la lettre sont écrites à leur place. Par exemple, pour `E` :

```
_ _ _ E _ _ E
```

Si la lettre ne figure pas dans le mot, le joueur est pénalisé d'un point. Au bout de sept pénalisations, le joueur est « pendu » à moins qu'il ne trouve le mot avant.

En général le jeu est agrémenté d'une sinistre potence dont le malheureux pendu se dessine progressivement à chaque mauvaise proposition.



Déroulement du programme
La machine organise le jeu.

1. Le premier joueur entre un mot de 3 à 10 lettres pendant que l'autre joueur ne regarde pas.
2. Le mot s'affiche avec des blancs soulignés à la place des lettres.
3. Le programme demande la saisie d'une lettre.
4. Le second joueur tape une lettre.
   - En cas de succès, le mot s'affiche (avec les lettres trouvées).
   - En cas d'échec, le programme indique combien de propositions il reste.
5. Le mot s'affiche avec des lettres et des soulignés selon qu'elles aient été découvertes ou non.
6. Le programme reprend au point 3 sauf si le joueur a épuisé tous ses coups ou trouvé le mot.

À la fin de la partie, le programme affiche si le joueur est pendu ou non.

1.2 Saisie d'un mot (7 points)

L'idée est d'utiliser un tableau d'au plus dix caractères pour mémoriser le mot à deviner, ainsi qu'un tableau de booléens pour les lettres déjà proposées.

De plus, pour éviter d'avoir des mots dont certaines lettres seraient en minuscule et d'autres en majuscule, seules des lettres majuscules pourront être saisies.

(1 point) Définissez les constantes `LMIN=3` (longueur minimale) et `LMAX=10` (longueur maximale) des mots. Puis définissez le type `TabCarac` comme étant un tableau de `LMAX` caractères.

Validez vos définitions avec la solution.

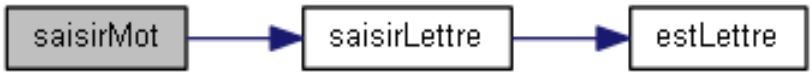
Solution Python @[pgjeupendu.py]

```
LMIN = 3
""" Longueur minimale des mots """
LMAX = 10
""" Longueur maximale des mots """
```

(1 point) Écrivez une fonction `estLettre(c)` qui teste et renvoie `Vrai` si un caractère `c` est une lettre majuscule (donc compris entre `'A'` et `'Z'`), `Faux` sinon.

(2 points) Déduisez une fonction `saisirLettre` qui effectue la saisie contrainte d'une lettre puis la renvoie, c.-à-d. que la fonction doit demander un caractère jusqu'à ce que `estLettre` soit `Vrai`.

(3 points) Enfin écrivez une fonction `saisirMot(mot)` qui effectue la saisie d'un entier `n` (contraint entre `LMIN` et `LMAX`), saisit `n` lettres dans un `TabCarac mot` par appel à la fonction `saisirLettre` puis qui renvoie `n` (longueur de `mot`).



Validez vos fonctions avec la solution.

Solution Python @[pgjeupendu.py]

```python
def estLettre(c):
    """  Prédicat de lettre majuscule

     :param c: un caractère
     :return: Vrai si c est une lettre majuscule, Faux sinon
    """
    return (("A" <= c) and (c <= "Z"))
```

```python
def saisirLettre():
    """  Saisie d'une estLettre

     :return: une estLettre
    """
    c = "_"
    while not estLettre(c):
        c = input("Votre majuscule? ")[0]
    return c
```

```python
def saisirMot(mot):
    """  Saisie d'un TabCarac

     :param mot: un TabCarac
     :return: la taille de mot
    """
    n = 0
    while not (LMIN <= n and n <= LMAX):
        n = int(input("Longueur du mot? "))
    for j in range(0, n):
        mot[j] = saisirLettre()
    return n
```

(0.5 point) Écrivez le début d'un script qui déclare un `TabCarac` puis effectue sa saisie.

Testez.

1.3 Procédures et fonctions utilitaires (4 points)

(1 point) Écrivez une procédure `initTentative(mot,n)` qui initialise les `n` premières cases d'un `TabCarac mot` avec des blancs soulignés (`_`).

Validez votre procédure avec la solution.

Solution Python @[pgjeupendu.py]

```python
def initTentative(mot, n):
    """  Initialisation d'un TabCarac

     :param mot: un TabCarac
     :param n: taille de mot
```

```
    """
    for j in range(0, n):
        mot[j] = "_"
```

Définissez la constante `NLETTRES=26` (nombre de lettres de l'alphabet) puis le type `TabBool` comme étant un tableau de `NLETTRES` booléens.

(1 point) Écrivez une procédure `initPropositions(t,n)` qui initialise à `Faux` les `n` premières cases d'un `TabBool t`.

Validez vos définitions et votre procédure avec la solution.

Solution Python @[pgjeupendu.py]

```
NLETTRES = 26
""" Nombre de lettres """
```

```
def initPropositions(t, n):
    """  Initialisation d'un TabBool

     :param t: un TabBool
     :param n: taille de t
    """
    for j in range(0, n):
        t[j] = False
```

(1 point) Écrivez une procédure `afficherMot(mot,n)` qui affiche les `n` premiers caractères d'un `TabCarac mot`.

(1 point) Écrivez une procédure `afficherPotence(n)` qui affiche le texte d'ordre `n` (entier) de la pendaison :

```
LA POTENCE EST PROCHE... VOUS ETES ...PENDU (<- Texte)
1  2       3   4         5    6    7        (<- valeur de n)
```

Par exemple, pour `n` valant 3 le texte affiché sera :

```
LA POTENCE EST
```

Aide simple

On testera successivement :

- Si `n >= 1` alors affichez `LA`.
- Si `n >= 2` alors affichez `POTENCE`.
- Si `n >= 3` alors affichez `EST`.
- et ainsi de suite jusqu'à `n >= 7`.

Validez vos procédures avec la solution.

Solution Python @[pgjeupendu.py]

```
def afficherMot(mot, n):
    """  Affichage d'un TabCarac

     :param mot: un TabCarac
     :param n: taille de mot
    """
    for j in range(0, n):
        print(mot[j], end="")
    print()
```

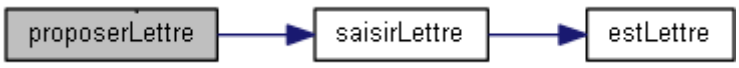
```
def afficherPotence0(n):
    """  Affichage de la potence

     :param n: nombre de coups ratés
    """
    if n >= 1:
        print("LA ", end="")
    if n >= 2:
        print("POTENCE ", end="")
    if n >= 3:
        print("EST ", end="")
    if n >= 4:
        print("PROCHE... ", end="")
    if n >= 5:
        print("VOUS ", end="")
    if n >= 6:
        print("ETES ", end="")
    if n >= 7:
        print("...PENDU ", end="")
    print(" (reste ", (MAXCOUPS - n), " coups)", sep="")
```

1.4 Proposition d'une lettre (4 points)

(3 points) Écrivez une fonction `inclureDansMot(c,adeviner,mot,n)` qui inclut le caractère `c` dans un `TabCarac mot` si celui-ci apparaît dans les `n` premiers caractères d'un `TabCarac adeviner`. Dans ce cas, la fonction renvoie `Vrai` si `c` a été trouvé **au moins** une fois dans `adeviner`, `Faux` sinon.

(1 point) Écrivez une fonction `proposerLettre(prop)` qui saisit une lettre par appel à la fonction `saisirLettre` jusqu'à ce qu'elle n'ait pas déjà été proposée dans le `TabBool prop`. La fonction fixera ensuite le booléen correspondant à `Vrai` dans `prop` puis renverra la lettre saisie.



Validez vos fonctions avec la solution.

Solution Python @[pgjeupendu.py]

```python
def inclureDansMot(c, adeviner, mot, n):
    """  Insère un caractère dans un TabCarac

     :param c: un caractère
     :param adeviner: mot a-deviner
     :param mot: mot du joueur
     :param n: taille des TabCarac
     :return: Vrai si c apparait dans adeviner
    """
    ok = False
    for j in range(0, n):
        if adeviner[j] == c:
            mot[j] = c
            ok = True
    return ok
```

```python
def proposerLettre(prop):
    """  Proposition d'une lettre

     :param prop: un TabBool
     :return: la lettre proposée
    """
    c = saisirLettre()
    while prop[c - "A"]:
        print("Majuscule deja proposee...")
        c = saisirLettre()
    prop[c - "A"] = True
    return c
```

1.5 Divination (5 points)

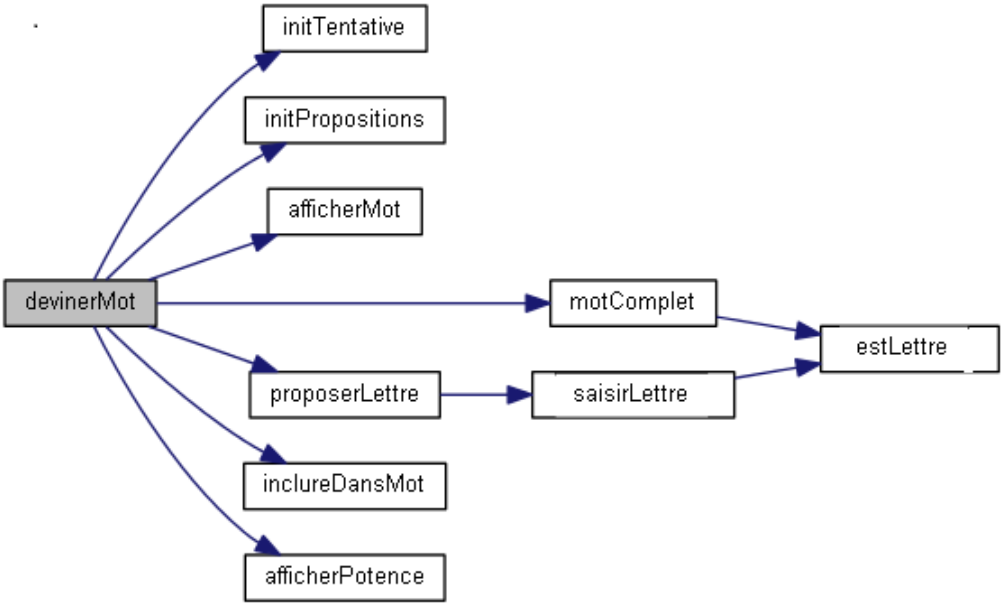
Reste à définir la procédure qui permet à un joueur de proposer des lettres jusqu'à ce qu'il trouve le mot de `n` lettres d'un `TabCarac adeviner` en au plus sept coups. Cette procédure devra :

- Déclarer et initialiser un tableau de booléens.
- Afficher partiellement le mot à deviner.
- Puis, tant que le joueur n'a pas épuisé ses sept propositions et qu'il n'a pas trouvé le mot complet :
  - — Demander et effectuer la saisie d'une lettre.
  - — Réaliser les changements à faire dans le tableau de booléens en fonction de la lettre proposée.
  - — Si la lettre est trouvée, afficher partiellement le mot à deviner.
  - — Sinon diminuer le nombre de propositions restantes et afficher la potence.
- Finalement afficher le résultat en indiquant si le joueur est pendu ou non.

(1 point) Écrivez une fonction `motComplet(mot,n)` qui teste et renvoie `Vrai` si les `n` premières cases d'un `TabCarac mot` sont des `estLettres`, `Faux` sinon (c.-à-d. qu'il reste au moins un blanc souligné).

Définissez la constante `MAXCOUPS=7` (nombre de coups maximum).

(3 points) Écrivez une procédure `devinerMot(adeviner,n)` qui réalise l'algorithme spécifié ci-avant, c.-à-d. qui permet à un joueur de proposer des lettres jusqu'à ce qu'il trouve le mot `adeviner` de `n` lettres ou qu'il ait épuisé ses `MAXCOUPS` propositions.



Validez votre fonction, définition et procédure avec la solution.

Solution Python @[pgjeupendu.py]

```
def motComplet(mot, n):
    """  Prédicat de mot trouvé

     :param mot: un TabCarac
     :param n: taille de mot
     :return: Vrai si toutes les lettres ont été trouvées
    """
    j = 0
    while j < n and estLettre(mot[j]):
        j += 1
    return (j == n)
```

```
MAXCOUPS = 7
""" Nombre maximum de coups """
```

```
def devinerMot(adeviner, n):
    """  Jeu du pendu

     :param adeviner: mot a-deviner
```

```
     :param n: taille de adeviner
    """
    mot = ["" for x in range(LMAX)]
    initTentative(mot, n)
    prop = [False for x in range(NLETTRES)]
    initPropositions(prop, NLETTRES)
    ncoups = 0
    afficherMot(mot, n)
    while ncoups < MAXCOUPS and not motComplet(mot, n):
        c = proposerLettre(prop)
        ok = inclureDansMot(c, adeviner, mot, n)
        if ok:
            afficherMot(mot, n)
        else:
            ncoups += 1
            afficherPotence(ncoups)
    if motComplet(mot, n):
        print("Super Gagne en ", ncoups, " coups", sep="")
    else:
        print("Pendu")
```

(0.5 point) Complétez votre script en appelant la procédure précédente.

Validez votre script avec la solution.

Solution Python @[pgjeupendu.py]

```
def PGJeupendu():
    adeviner = ["" for x in range(LMAX)]
    n = saisirMot(adeviner)
    devinerMot(adeviner, n)
```

2 Références générales

Comprend [Chappelier-CPP1 :c7 :ex45] ■