INFO-H-100 - Informatique

Séance d'exercices 8
Structuration d'un programme

Université Libre de Bruxelles
Faculté des Sciences Appliquées

2013-2014

Enoncé - Le pendu

Le pendu est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent. Le jeu se joue traditionnelement à deux, mais nous programmerons une version à un joueur.

Etapes de jeu

- L'ordinateur choisit un mot au hasard dans un dictionnaire.
- Tant que le joueur n'a pas épuisé ses 10 essais :
  1. Le joueur choisit une lettre
  2. L'ordinateur vérifie si cette lettre est dans le mot et affiche les occurrences de cette lettre le cas échéant.
- Si le joueur trouve le mot, il gagne.
- L'ordinateur propose une nouvelle partie au joueur.

Quelle sont les tâches à accomplir ?

Quelle sont les tâches à accomplir ?

- Lancer le jeu
- Récupérer un dictionnaire de mots depuis un fichier
- Sélectionner aléatoirement un mot depuis ce dictionnaire
- Demander une lettre au joueur (tant qu'il ne s'agit pas d'une seule lettre)
- Vérifier si la lettre est présente dans le mot et remplacer les occurrences de celle-ci dans le mot à trouver
- Vérifier si le mot a bien été trouvé
- Proposer au joueur de faire une nouvelle partie

Découpe en fonctions

Bonne pratique : ne pas mélanger des `return` avec des `print/input`

Les fonctions de "logique du jeu" (pas de `print`)

1. **load_dico(filename)**
   Fonction renvoyant un `set` chargé depuis un fichier
2. **get_word_from_dico(dico)**
   Fonction renvoyant un mot choisi aléatoirement dans le `set` fourni en paramètre
3. **update_guess(word,word_guessed,letter)**
   Fonction remplaçant les occurrences d'une lettre dans le mot à deviner et le renvoyant
4. **generate_guess(word)**
   Fonction renvoyant le mot que le joueur doit découvrir, mais avec les lettres masquées

Découpe en fonctions

Bonne pratique : ne pas mélanger des `return` avec des `print/input`

Fonction interagissant avec le joueur (avec des `print`)

1. **game()**
   La boucle de jeu principale ; fonction lançant le jeu puis demandant à l'utilisateur s'il désire rejouer
2. **hang()**
   Fonction exécutant une seule partie du pendu
3. **ask_yes_no()**
   Fonction demandant à l'utilisateur d'introduire `o` ou `n`
4. **ask_letter()**
   Fonction demandant à l'utilisateur une lettre
5. **show_status()**
   Fonction indiquant au joueur le nombre d'essais restant et le statut actuel du mot à découvrir
6. **endgame()**
   Fonction affichant un message approprié en cas de victoire ou de défaite

Implémentation

Les fonctions de "logique du jeu" (pas de `print`)

1. **load_dico(filename)**
   Fonction renvoyant un `set` chargé depuis un fichier
2. **init_game()**
   Fonction renvoyant les paramètres du jeu soit, le nombre d'essais et le dictionnaire
3. **get_word_from_dico(dico)**
   Fonction renvoyant un mot choisi aléatoirement dans le `set` fourni en paramètre
4. **generate_guess(word)]**
   Fonction renvoyant le mot que le joueur doit découvrir, mais avec les lettres masquées
5. **update_guess(word,word_guessed,letter)**
   Fonction remplaçant les occurrences d'une lettre dans le mot à deviner et le renvoyant
6. **check_win()**
   Fonction booléenne indiquant si la partie est gagnée

Fonction : load_dico(filename)

```
def load_dico(filename):
    """
    Desc:   Return a set of words loaded from a given file.
    Input:  filename (str)
    Output: dico (set)
    """
    f = open(filename)
    return set(f.read().split())
```

Fonction : init_game(dico)

```
def init_game(dico):
    """
    Desc:   Return the variables of the game (number of trials
            and word to guess)
    Input:  dico (set)
    Output: trials (int), word, (str)
    """
    trials = 10
    word = get_word_from_dico(dico)
    return trials, word
```

Fonction : get_word_from_dico(dico)

```
def get_word_from_dico(dico):
    """
    Desc:   Return a randomly selected word from a set.
    Input:  dico (set)
    Output: word (str)
    """
    return random.sample(dico, 1)[0].lower()
```

Fonction : generate_guess(word)

```
def generate_guess(word):
    """
    Desc:   Return a string of 'word's length composed only of '_'.
    Input:  word (str)
    Output: word_guessed (str)
    """

    return "_" * len(word)
```

Fonction : update_guess(word,word_guessed,letter)

```
def update_guess(word,word_guessed,letter):
    """
    Desc:   Check for occurences of 'letter' in 'word' and replace
            those letters in 'word_guessed' at the same positions.
            Return the modified 'word_guessed'
    Input:  word (str) , word_guessed (str) , letter (str)
    Output: word_guessed (str)
    """
    word_guessed = list(word_guessed)
    for i in range(len(word)):
        if word[i] == letter:
            word_guessed[i] = letter
    return "".join(word_guessed)
```

Fonction : check_win(win)

```
def check_win(word_guessed, word):
    """
    Desc:   Check if the word has been guessed.
            Return 'True' if the word has been guessed, 'False'
            otherwise.
    Input:  win (str, str)
    Output: (bool)
    """
    return word_guessed == word
```

Implémentation

Fonction interagissant avec le joueur (avec des `print`)

1. **game()**
   La boucle de jeu principale ; fonction lançant le jeu puis demandant à l'utilisateur s'il désire rejouer
2. **hang()**
   Fonction exécutant une seule partie du pendu.
3. **ask_yes_no()**
   Fonction demandant à l'utilisateur d'introduire `o` ou `n`
4. **ask_letter()**
   Fonction demandant à l'utilisateur une lettre
5. **show_status()**
   Fonction indiquant au joueur le nombre d'essais restant et le statut actuel du mot à découvrir
6. **endgame()**
   Fonction affichant un message approprié en cas de victoire ou de défaite

Fonction : ask_letter(message)

```
def ask_letter(message):
    """
    Desc:   Return the answer of a user to the message received
            in parameter. The question is asked until the answer
            is one alphabetic letter.
    Input:  message (str)
    Output: answer (str)
    """
    answer = input(message)
    while len(answer) != 1 or not answer.isalpha():
        print("Entrée erronée, veuillez réessayer")
        answer = input(message)
    return answer
```

Fonction : ask_yes_no(message)

```
def ask_yes_no(message):
    """
    Desc:   Return 'True' if a user's answer to the message
            received in parameter is 'o', and 'False' if it's
            answer is 'n'. The question is asked until the
            answer is either 'o' or 'n'.
    Input:  message (str)
    Output: answer (bool)
    """
    answer = input(message+" o/n ").lower()
    while answer != "o" and answer != "n":
        answer = input(message+" o/n ").lower()
    return answer == "o"
```

Fonction : endgame(word_guessed, word)

```
def endgame(word_guessed, word):
    """
    Desc:   Print the end game message.
    Input:  endgame (bool)
    Output:
    """
    if check_win(word_guessed, word):
        print("Bravo, vous avez gagné !")
    else:
        print("Dommage, vous avez perdu...")
        print("Le mot à deviner était : " + word)
```

Fonction : show_status(trials,word_guessed)

```
def show_status(trials, word_guessed):
    """
    Desc:   Show the status of the game (number of trials
            and word to guess)
    Input:  trials (int), word_guessed (str)
    Output:
    """
    print("Il vous reste " + str(trials) + " essais")
    print("Le mot est actuellement : ",word_guessed)
```

Fonction : hang()

```
def hang(dico):
    """
    Desc:   Execute one game of 'Hanged Man' using the set 'dico'.
    Input:  dico (set)
    Output:
    """
    trials, word = init_game(dico)
    word_guessed = generate_guess(word)
    show_status(trials, word_guessed)
    while trials != 0 and not check_win(word_guessed, word):
        letter = ask_letter("Indiquez une lettre : ").lower()
        if letter not in word:
            trials -= 1
        else:
            word_guessed = update_guess(word,word_guessed,letter)
        show_status(trials, word_guessed)
    endgame(word_guessed, word)
```

Fonction : game()

```
def game():
    """
    Desc:   Start a new game until the user wants to stop playing.
    Input:
    Output:
    """
    dico = load_dico("dico.txt")
    hang(dico)
    while(ask_yes_no("Faire une nouvelle partie ?")):
        hang(dico)
```