Exercices de programmation du jeu du pendu - documentation professeur

Accès rapide

Cette section permet de retrouver rapidement les différents liens présents dans ce document.

- Dépôt maître : https://github.com/epeios-q37/hangman-exercises ;
- formulaire pour contacter l'auteur : https://q37.info/s/ggq7x4w7 ;
- documents professeur :
  - ce document : https://q37.info/s/mmdnch9t,
    - version *PDF* en cas de problèmes d'affichage : https://q37.info/s/khbq3tj3,
  - le dépôt *GitHub* contenant les solutions aux exercices : https://github.com/epeios-q37/hangman-fr-teacher,
  - essayer ces solutions en ligne : https://repl.it/github/epeios-q37/hangman-fr-teacher ;
- documents élèves :
  - le document contenant les instructions pour les élèves : https://q37.info/s/zmg4m3fx,
    - version *PDF* en cas de problèmes d'affichage : https://q37.info/s/htkr9drb,
  - le dépôt *GitHub* contenant les fichiers destinés aux élèves : https://github.com/epeios-q37/hangman-fr,
  - pour travailler sur ces exercices dans un environnement de développement en ligne : https://repl.it/github/epeios-q37/hangman-fr.

Présentation

De nos jour, la plupart des jeunes possèdent un smartphone. Il est donc plus facile de les intéresser à la programmation si les exercices se présentent comme les applications qu'ils utilisent sur leurs smartphones.

C'est pourquoi les exercices proposés ici ont une véritable interface graphique, au lieu de l'habituelle et austère interface texte. De plus, avec chaque exercice, un code QR y donnant accès à partir d'un smartphone est affiché. Ainsi, ils verront qu'ils peuvent utiliser leurs smartphones pour accèder à leurs propres applications, et non pas seulement à des applications comme *Facebook*, *YouTube*, *Twitter*....

En outre, ils peuvent facilement partager cet accès avec leurs proches, directement sur leurs smartphones (ou n'importe quel dispositif équipé d'un navigateur web moderne connecté à internet) respectifs. Ces derniers auront ainsi plus d'occasion de les encourager, et les inciter ainsi à progresser.

Vous trouverez plus de détails sur ce type d'exercices à l'adresse https://q37.info/s/knf9hdwp.

Les exercices présentés ici reproduisent les étapes menant à l'élaboration d'un programme, chaque exercice s'appuyant sur le précédent. Ils renforcent l'intérêt des élèves pour la programmation par une utilisation concrète des concepts abordés par les nombreux exercices de programmations que l'on trouve par ailleurs.

Contrairement à la plupart des autres exercices de programmation, chaque exercice ne porte pas sur un concept informatique, mais sur l'élaboration d'une fonctionnalité du programme. L'assimilation des concepts en est ainsi facilitée, car n'étant pas une fin en soi, mais une étape dans la réalisation du programme.

Pour confèrer une dimension ludique à ces exercices, le programme en question est un jeu, le jeu du pendu.

Bien que s'appuyant sur le langage *Python*, ces exercices portent sur la programmation en générale, et non pas sur la programmation en *Python*. C'est pour cela que les solutions générales seront données avant les solutions propres à *Python*.

Utilisation

Pour aborder ces exercices, seules des notions de base en développement *Python* sont nécessaires. Ils sont particulièrement indiqués pour faire découvrir ou initier à la programmation.

Le dépôt https://github.com/epeios-q37/hangman-fr est destiné aux élèves. https://github.com/epeios-q37/hangman-fr-teacher reprend le contenu de ce dépôt, en y ajoutant des informations utiles aux professeurs, notamment les solutions des exercices. De ce fait, ce dernier dépôt est réservé aux professeurs.

Pour distribuer ces exercices aux élèves, vous pouvez directement utiliser le dépôt https://github.com/epeios-q37/hangman-fr ou, si vous êtes familiarisés avec *GitHub*, en utiliser les fonctionnalités pour créer votre propre version de ce dépôt.

Les solutions aux exercices présentées dans ce document peuvent être vue à l'œuvre en allant à l'adresse https://repl.it/github/epeios-q37/hangman-fr-teacher. Les fichiers contenus dans le répertoire `Student` sont ceux mis à disposition des étudiants.

Si vous avez des retours à faire concernant ces exercices, vous pouvez contacter l'auteur à l'adresse https://q37.info/s/ggq7x4w7, ou passer par le dépôt maître, d'où sont dérivés les deux dépôts présentés ici, à l'adresse https://github.com/epeios-q37/hangman-exercises.

Il existe deux façons pour les élèves de travailler sur ces exercices. Soit en utilisant un ordinateur sur lequel est installé *Python* (utilisation en local), soit dans un navigateur web (utilisation en ligne).

Utilisation en local

Pour une utilisation en local sur un ordinateur équipé de *Python*, il suffit aux élèves de télécharger et désarchiver le fichier suivant :

https://github.com/epeios-q37/hangman-fr/archive/master.zip. Alternativement, s'ils possèdent les connaissances adéquates, ils peuvent aussi faire un `git clone https://github.com/epeios-q37/hangman-fr` à partir d'une console.

Les exercices consistent à écrire du code dans le fichier `pendu.py` situé à la racine du dépôt. Pour le lancement, il suffira d'exécuter la commande `python3 pendu.py` dans une console, à partir de la racine du dépôt. S'ouvrira alors un navigateur web leur donnant accès à leur exercice. Le programme sera arrêté avec un *CTRL-C*.

Ce dépôt contient, à la racine, un fichier `LISEZMOI.html` qui leur donne les instructions pour chaque exercice.

Utilisation en ligne

Pour une utilisation en ligne en utilisant un navigateur web, sans rien avoir à installer, il suffit de suivre le lien suivant : https://repl.it/github/epeios-q37/hangman-fr. Cela ouvre une session dans *Repl.it*, un environnement de développement en ligne.

Le fichier `LISEZMOI.html` leur donnant les instructions pour chaque exercice se situe à l'adresse https://q37.info/s/zmg4m3fx. Il faut l'ouvrir via cette adresse, car l"accès à ce fichier via *GitHub* ou *Repl.it* n'en affichera que le code source.

Comme ci-dessus, l'exercice consistera à écrire du code dans le fichier `pendu.py`. Pour lancer l'exercice, il suffira de cliquer sur le bouton vert. Cela provoquera l'affichage d'un code QR que l'on cliquera (ou éventuellement scannera avec un smartphone) pour accèder à l'interface de l'exercice.

Pour revenir au code source, on fermera l'onglet (ou la fenêtre) qui s'est ouverte suite au click sur le code QR. Pour relancer l'application, il suffira de cliquer à nouveau sur le bouton vert (dont le libellé change en fonction du contexte).

Repl.it ne nécessite pas la création d'un compte pour être utilisé. Néanmoins, ouvrir un compte permet d'y stocker ses projets et de les retrouver d'une session à l'autre. On peut aussi, sans compte, sauvegarder en local l'ensemble d'un projet.

Les exercices

Les informations données ici viennent en complément de celles indiquées dans le fichier `Student/LISEZMOI.html`.

ATTENTION : ne pas oublier de modifier la ligne `from workshop.fr.… import *` en fonction de l'exercice.

Exercice a

Particularités

- Édition d'un fichier source ;
- lancement d'un programme ;
- arrêt d'un programme ;
- affectation d'un booléen ;
- création d'une chaîne de caractères ;
- valeur de retour d'une fonction.

Solution

```
def choisirMot():
  return "arbre"
```

Exercice b

Particularités

- Paramètre de fonction ;
- instruction conditionnelle ;
- taille d'une chaîne de caractères ;
- booléen :
  - test,
  - transtypage ;
- opérateur de comparaison ;
- instruction conditionnelles.

Solutions

```
def choisirMot(suggestion):
  if len(suggestion) != 0:
    return suggestion
  else:
    return "arbre"
```

Une fois que cette solution aura été trouvée, on pourra proposer de remplacer `if len(suggestion) != 0:` par `if len(suggestion):`, puis par `if suggestion:` en soulignant que c'est propre à *Python* (et à certains autres langages).

Exercice c

La seconde solution est optionnelle, et on pourra y revenir plus tard.

Particularités

Pour la première version, aucune en particulier ; il s'agit juste d'une amélioration.

Pour la seconde version :

- tuples :
  - création,
  - récupération d'un élément par son index,
  - récupération taille ;
- import d'une fonction particulière d'un module,
- fonction `randint`.

Solutions

Première version :

```
def choisirMot(suggestion,motAuHasard):
  if suggestion:
    return suggestion
  else:
    return motAuHasard
```

seconde version :

```
from random import randint

…

# Placer les mots de son choix.
DICTIONNAIRE = ("arbre", "maison", "chaise")


def choisirMot(suggestion):
  if suggestion:
    return suggestion
  else:
    return DICTIONNAIRE[randint(0, len(DICTIONNAIRE)-1)]
```

Exercice d

Particularités

- Boucle `for` ;
- opérateur `in` ;
- fonction `range` ;
- index et chaîne de caractères.

Solutions

```
def lettreEstDansMot(lettre,mot):
    for i in range(0,len(mot)):
        if mot[i] == lettre:
            return True

    return False
```

Une fois cette solution trouvée, on pourra proposer la solution suivante, en soulignant qu'elle est propre à *Python* (et certains autres langages).

```
def lettreEstDansMot(lettre,mot):
  return lettre in mot
```

Exercice e

Particularités

- Affectation d'une valeur à une variable ;
- concaténation de chaînes de caractères ;
- éventuellement instruction conditionnelle ternaire.

Solutions

```
def donnerMasque(mot,pioches):
    masque = ""

    for lettre in mot:
        if lettreEstDansMot(lettre,pioches):
            masque = masque + lettre
        else:
            masque = masque + "_"

    return masque
```

Une fois cette solution trouvée, on pourra proposer de remplacer `masque = masque + …` par `masque += …`.

On pourra également, à la place de :

```
if lettreEstDansMot(lettre,pioches):
    masque += lettre
else:
    masque += "_"
```

proposer :

```
masque += lettre if lettreEstDansMot(lettre,pioches) else "_"
```

Exercice f

Particularités

Instructions conditionnelles multiples (équivalent du *switch… case…* d'autres langages).

Solutions

```
def majCorps(nombreErreurs):
  if nombreErreurs == 1:
    partieCorps = P_TETE
  elif nombreErreurs == 2:
    partieCorps = P_TRONC
  elif nombreErreurs == 3:
    partieCorps = P_BRAS_GAUCHE
  elif nombreErreurs == 4:
    partieCorps = P_BRAS_DROIT
  elif nombreErreurs == 5:
```

```
    partieCorps = P_PIED_DROIT
  elif nombreErreurs == 6:
    partieCorps = P_PIED_GAUCHE
  elif nombreErreurs == 7:
    partieCorps = P_VISAGE

  dessinerPartieCorps(partieCorps)
```

Autre solution :

```
PARTIES_CORPS = (
  P_TETE,
  P_TRONC,
  P_BRAS_GAUCHE,
  P_BRAS_DROIT,
  P_PIED_GAUCHE,
  P_PIED_DROIT,
  P_VISAGE
)

def majCorps(nombreErreurs):
  dessinerPartieCorps(PARTIES_CORPS[nombreErreurs-1])
```

Autres exercices

La partie des notices professeurs et élèves concernant ces exercices sont en cours d'élaboration.