université
PARIS-SACLAY

Programmation en Python
Master 2 Réseaux Télécoms
TP 8

1 Une calculatrice graphique

Nous allons réaliser une calculette graphique rudimentaire en utilisant le package `PyQt5` (voir Figure 1).

FIGURE 1 – L'interface de la calculette.

Vous devez completer le code des méthodes suivantes dans le fichier `qtcalc.py` :

1. `validateInput(self)` : cette méthode vérifie si les textes dans les champs d'entrée `edit_a` et `edit_b` sont interprétables comme des nombres flottants. Si c'est le cas, la méthode retourne un tuple de ces nombres, sinon elle appelle la fonction `alert` en indiquant dans le message que l'entrée est non-conforme, puis retourne `None`.
2. `do_add(self)`, `do_sub(self)`, `do_mul(self)` et `do_div(self)` : Ces méthodes sont appelées quand l'utilisateur appuie sur les boutons correspondants. Elles doivent valider les valeurs dans les champs d'entrée (utiliser la méthode `validateInput`), puis faire l'opération arithmétique demandée et mettre à jour la valeur du résultat `label_ans`. Notez que `do_div(self)` doit aussi vérifier que le dénominateur est différent de zéro, dans le cas contraire on affiche un message d'erreur en utilisant la fonction `alert`.

2 Jeu du pendu

Dans cet exercice nous allons créer une application graphique pour un jeu du pendu. Le but de ce jeu consiste à deviner un mot sans faire trop d'erreurs. Au départ, toutes les lettres du mot secret sont cachées. À chaque lettre devinée correctement, toutes les occurrences de cette lettre dans le mot secret sont démasquées (voir Figure 2).

FIGURE 2 – L'interface du jeu du pendu.

L'interface du jeu est donnée par la classe `Hangman` dans le fichier `hangman.py`. L'état du jeu est entièrement déterminé par les membres-données suivantes :

1. `secret` : le mot secret à deviner.
2. `guessed` : L'ensemble des lettres proposées par l'utilisateur à un instant donné.
3. `errors` : Le nombre d'erreurs à un instant donné. Notez que cette variable est redondante, car il est toujours possible de calculer le nombre d'erreurs à partir des valeurs de `secret` et `guessed`.

Vous allez également avoir besoin du dictionnaire `dict_buttons` qui associe à chaque lettre de l'alphabet le bouton correspondant. Vous allez compléter le code des fonctions suivantes :

1. `letter_guessed(self, c)` : Cette fonction est appelée lorsque l'utilisateur appuie sur le bouton correspondant à la lettre `c`. La fonction aura l'effet suivant :
   (a) On ajoute la lettre dans l'ensemble `guessed`.
   (b) On déactive le bouton correspondant à la lettre (utiliser la méthode `setEnabled` de la classe `QPushButton`).
   (c) On recalcule le mot affiché (utiliser la méthode `cleartext`).

(d) On met à jour le nombre d'erreurs, si ce nombre dépasse la constante `MAX_ERRORS`, on appelle la méthode `end_game` avec le message `You lost!`

(e) Sinon, on vérifie si le mot est entièrement trouvé, auquel cas on appelle la méthode `end_game` avec le message `You won!`

(f) Sinon, on met à jour le message du label `status` en affichant le nombre d'erreurs commises.

2. `end_game(self, msg)` : Cette fonction met à jour le label `status`, en utilisant le message passé en argument. Ensuite elle déactive tous les boutons avec des lettres.

3. `reset(self)` : Cette fonction est appelée au début de chaque jeu. Elle aura l'effet suivant :

(a) On réactive tous les boutons avec des lettres.

(b) On choisit le mot `secret` au hasard dans la liste `SECRETS`.

(c) On réinitialise les variables `guessed` et `errors`

(d) On réinitialise le mot affiché (utiliser la méthode `cleartext`).

(e) On remet à vide le texte du label `status`.