

Formation Python

Atelier Pratique AP-PY11

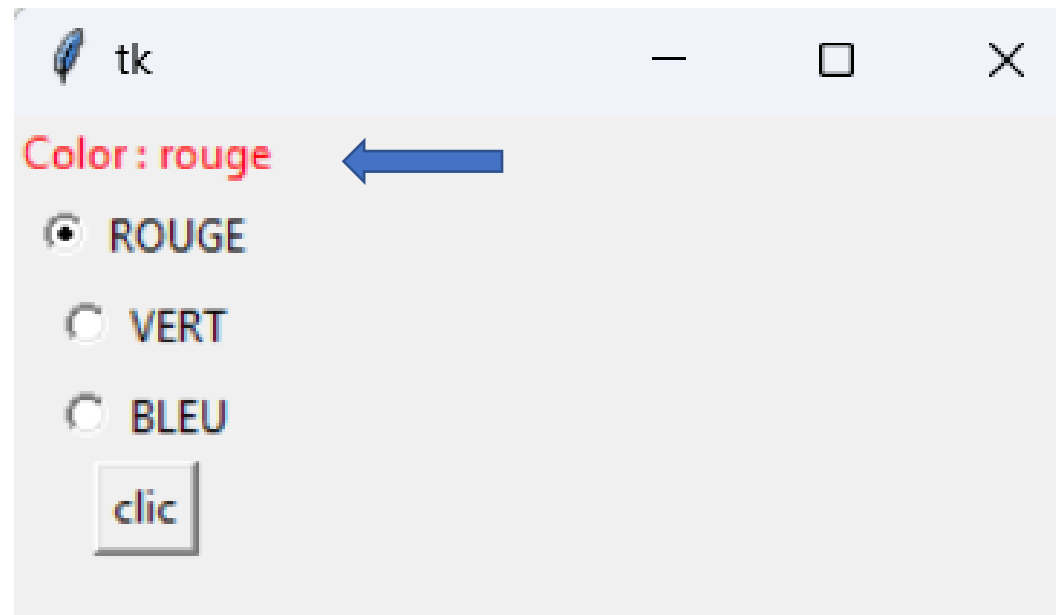
Atelier Pratique AP-PY11 : Exercice 11.3.3

- 1) Créer 3 Radiobuttons avec les labels suivants : **ROUGE** , **VERT** , **BLEU**
- 2) Créer une fonction ***update_description()*** qui met à jour le label **Color** lorsqu'on clique sur le **bouton** comme ceci :

Color : rouge si le Radiobouton ROUGE est coché

Color : vert si le Radiobouton VERT est coché

Color : bleu si le Radiobouton BLEU est coché



SOLUTION

Atelier Pratique AP-PY11 : Exercice 11.7.2

Créer 2 fenêtres :

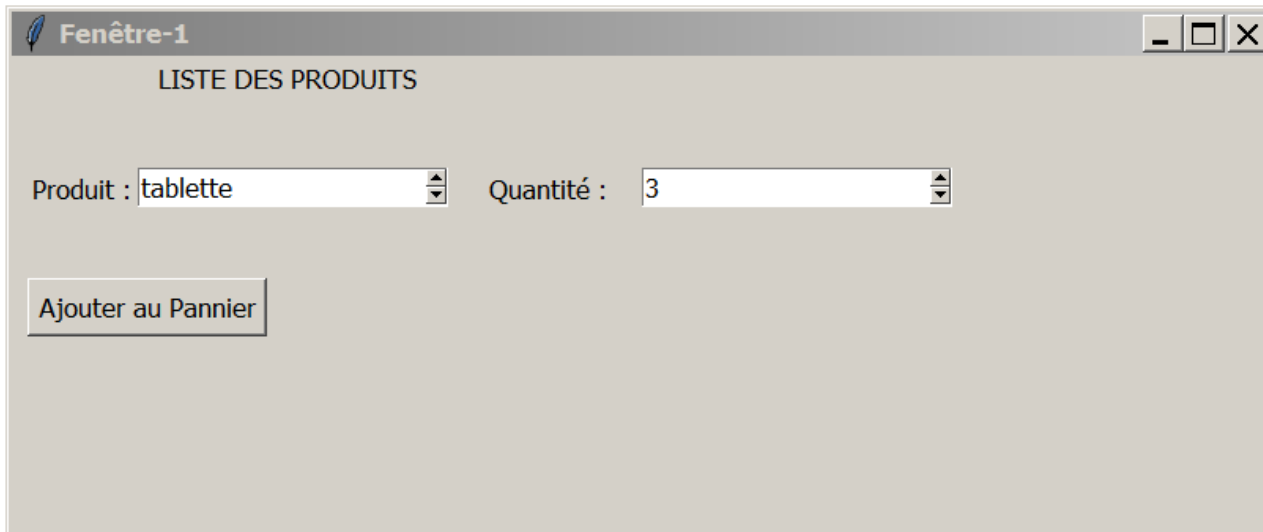
. **Fenêtre-1** : 2 spinbox : 'Produit' , 'Quantité' + bouton 'Ajouter au Panier'

Une fonction `ajouter_panier()` qui se déclenche lorsqu'on clique sur le bouton

`Liste_produits = ["ordinateur", "imprimante", "tablette"]`

`dico_prix = { "ordinateur":700, "imprimante":100, "tablette":90 }`

. **Fenêtre-2** : Afficher l'état du panier + 2 champs : 'Total HT' , 'Total TTC' + bouton 'Payer'

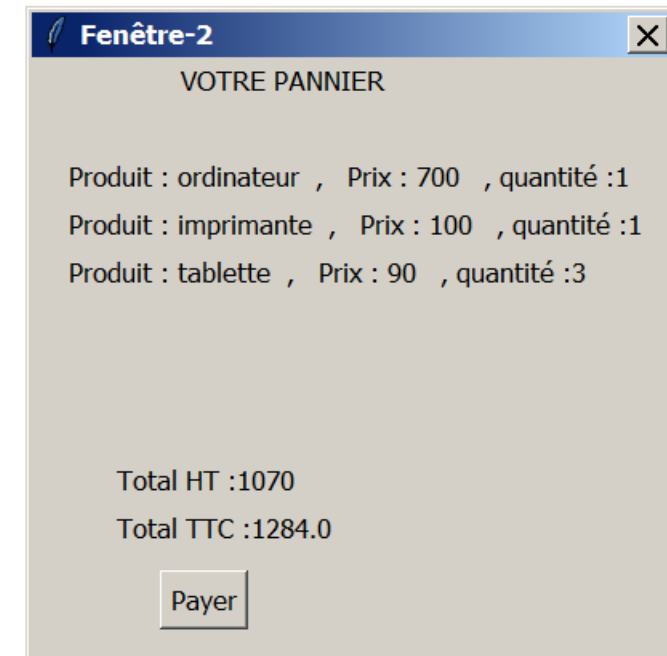


Fenêtre-1

LISTE DES PRODUITS

Produit : tablette Quantité : 3

Ajouter au Pannier



Fenêtre-2

VOTRE PANNIER

Produit : ordinateur , Prix : 700 , quantité :1
Produit : imprimante , Prix : 100 , quantité :1
Produit : tablette , Prix : 90 , quantité :3

Total HT :1070
Total TTC :1284.0

Payer

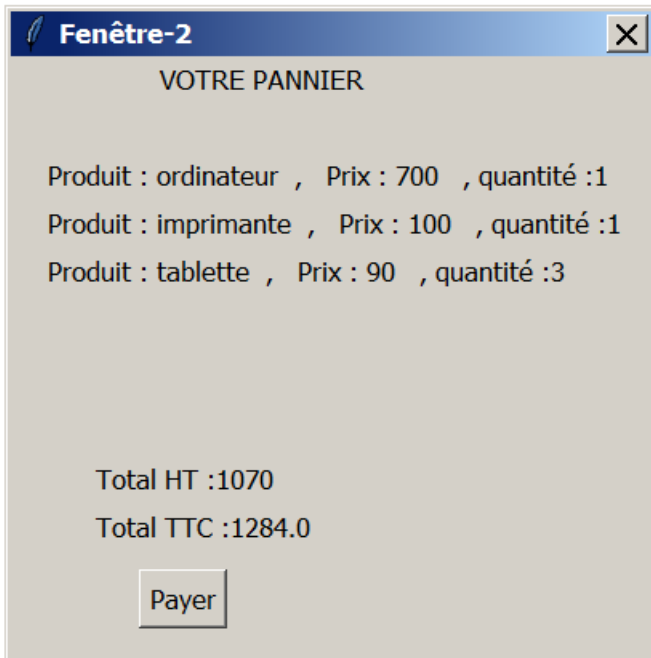
Atelier Pratique AP-PY11 : Exercice 11.7.2 (Cont)

Créer :

. **Fenêtre-3 : PAIEMENT:** Elle s'ouvre lorsqu'on clique sur le bouton 'Payer'

. **Status :** Un **pop-up** qui s'ouvre lorsqu'on clique sur le bouton 'Soumettre'

Utiliser l'instruction Python : `showinfo("Status", "Paiement effectué avec succès")`



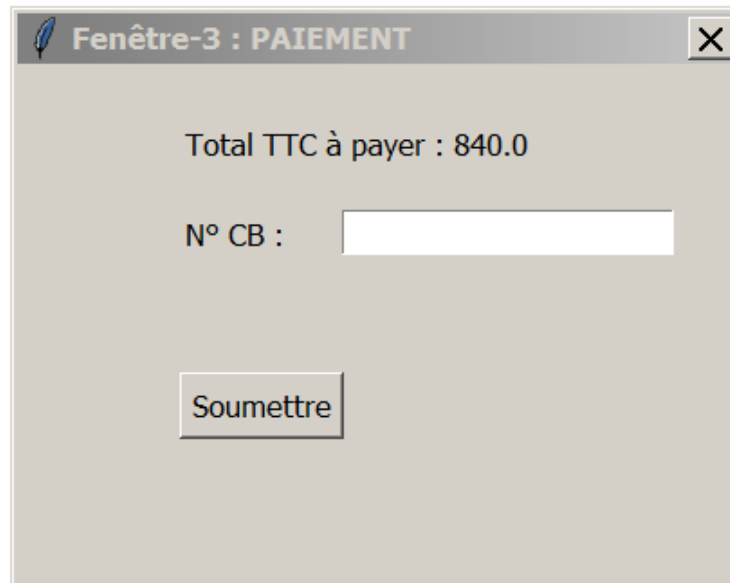
Fenêtre-2

VOTRE PANNIER

Produit : ordinateur , Prix : 700 , quantité :1
Produit : imprimante , Prix : 100 , quantité :1
Produit : tablette , Prix : 90 , quantité :3

Total HT :1070
Total TTC :1284.0

Payer

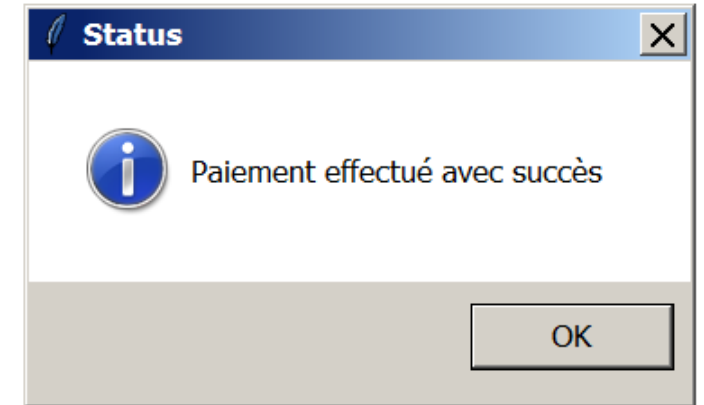


Fenêtre-3 : PAIEMENT

Total TTC à payer : 840.0

N° CB :

Soumettre



Status

 Paiement effectué avec succès

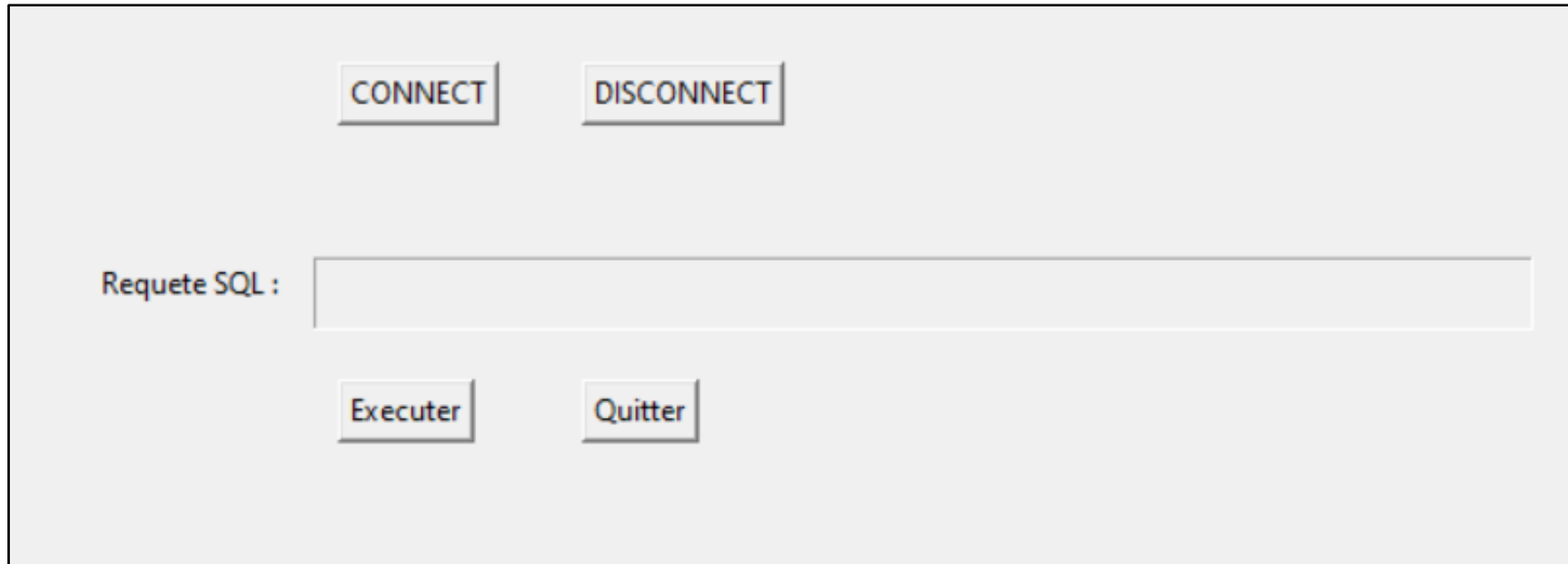
OK



SOLUTION

Atelier Pratique AP-PY11 : Exercice 11.13

Créer un interpréteur de requêtes SQL



The image shows a graphical user interface for a SQL query interpreter. It features a light gray background with a black border. At the top, there are two buttons: 'CONNECT' and 'DISCONNECT'. Below these, on the left, is the text 'Requete SQL :'. To the right of this text is a long, empty rectangular input field for entering SQL queries. At the bottom of the interface, there are two more buttons: 'Executer' and 'Quitter'.

- **CONNECT** :
 - Si le client est déjà connecté, afficher un pop-up : ***Une connexion est déjà en cours***
 - Si connexion avec succès, afficher un pop-up : ***Connexion : OK*** , et **activer le saisie** dans le champ SQL

Créer un interpréteur de requêtes SQL (Cont)

- **DISCONNECT** :
 - Si le client n'est pas connecté, afficher un pop-up : ***Pas de connexion en cours***
 - Si le client est connecté, afficher un pop-up : ***Bye ...***, et **désactiver le saisie** dans le champ SQL
- **EXECUTER** : Exécute de la requête SQL
- **QUITTER** :
 - Si le client est connecté, afficher un pop-up : ***Une connexion est en cours***
 - Si le client n'est pas connecté, **fermer toutes les fenêtres**

Atelier Pratique AP-PY11 : Exercice 11.13

Créer un interpréteur de requêtes SQL (Cont)

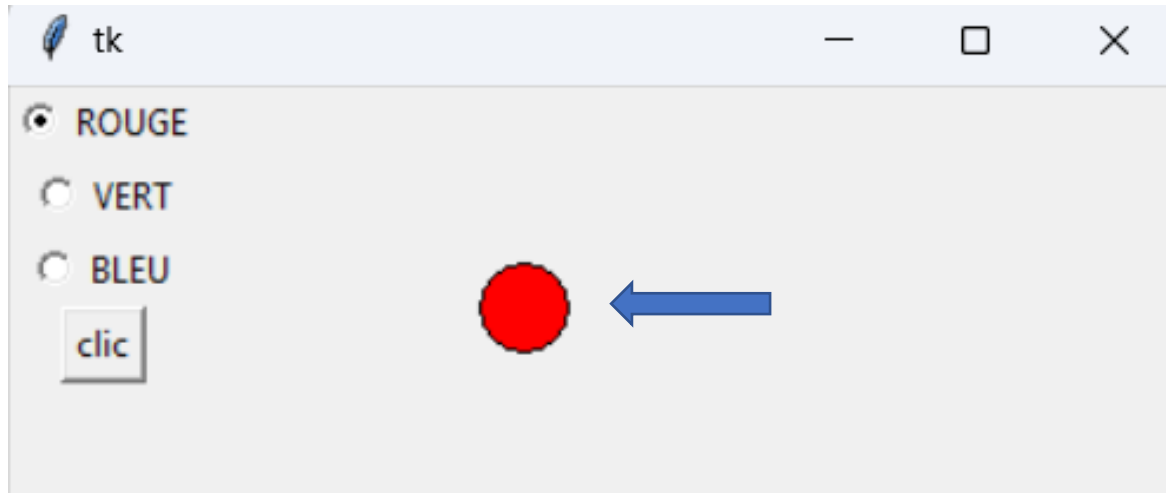
- 1) Créer une fonction *loaddb()* : charger la base de données *employe.db* à partir de *EMPLOYEE.xlsx* : module *loaddb.py*
- 2) Créer une fenêtre *fen1* comme indiqués sur la figure pour saisir la requête, et une *fen2* pour afficher les résultats
- 3) Créer une fonction *exec_sql()*.
 - Entrée : Le texte de la requête SQL
 - Appel d'une fonction *transaction()* qui exécute de la requête SQL et retourne le resultat et le status
 - Sortie : Le résultat de la requête, liste de rows si *SELECT*, ou un status si autre requête
- 4) Tester avec les requêtes SQL dans le fichier *PYExo11.13.4.txt*



SOLUTION

Atelier Pratique AP-PY11 : Exercice 11.15

- 1) Créer 3 Radiobuttons avec les labels suivants : **ROUGE** , **VERT** , **BLEU**
- 2) Créer un widget **canvas1** de type canevas avec la fonction **create_oval((x1,y1), (x2,y2))**
- 3) Créer une fonction **update_color()** qui modifie la couleur du canevas lorsqu'on clique sur le **bouton** comme ceci :



SOLUTION