

Documentation projet python

Préparé par:
p6ck.

EUROSPORT

DATE : 20/01/2025 AU 20/03/2025

VUE D'ENSEMBLE

<i>Historique</i>	Création d'un programme python pour permettre de contrôler les stocks de l'entreprise.
<i>Objectifs</i>	<ul style="list-style-type: none">• Développer mes compétences informatiques• Réaliser un véritable projet
<i>Audience cible</i>	L'entreprise peut être l'utilisateur de ce programme.

METTRE EN PLACE SON ENVIRONNEMENT DE TRAVAIL

Premièrement qu'est-ce que TKINTER ?

Tkinter est une bibliothèque standard de Python qui permet de créer des interfaces graphiques (GUI) facilement.

Elle fournit des outils pour ajouter des boutons, des menus, des champs de texte, etc., dans des fenêtres interactives.

Deuxièmement avoir une idée générale du projet.

Quelles informations à prendre en compte lors de la création du code, les informations à mettre etc...

DÉBUT DU CODE

Importer les éléments qu'on utilisera :

```
import tkinter as tk
import pandas as pd
from tkinter import messagebox, ttk
from datetime import datetime
import csv
import os
```

Seconde étape “générer” la fenêtre du code et ajouter les zones de textes :

```
root.geometry("800x600")
root.minsize(400, 300)
root.configure(bg='lightblue')

style = ttk.Style()
style.configure("Treeview", font=("Arial", table_font_size))
style.configure("Treeview.Heading", font=("Arial", table_font_size + 2))

button_font = ("Arial", 15, "bold")

root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=1)
root.columnconfigure(2, weight=1)
root.rowconfigure(3, weight=1)

frame_form = tk.Frame(root)
frame_form.grid(row=1, column=0, columnspan=3, pady=10, sticky="ew")

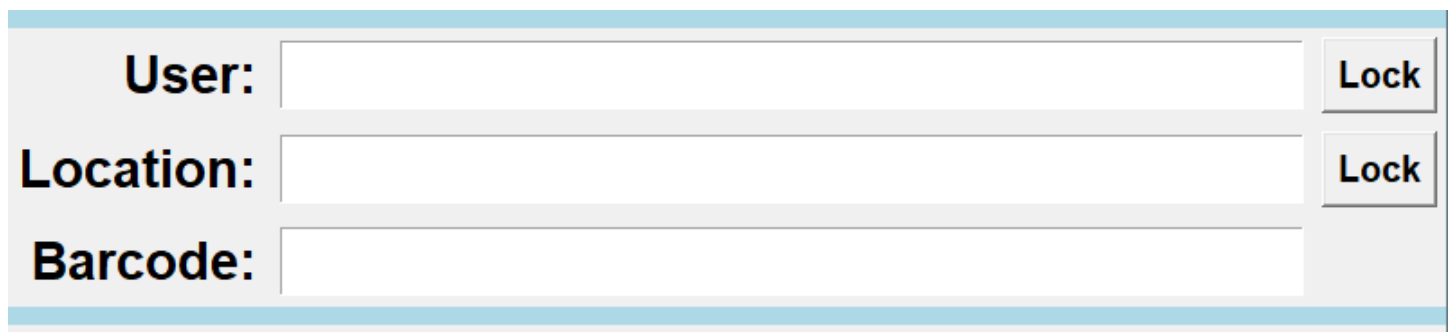
frame_form.columnconfigure(1, weight=1)

tk.Label(frame_form, text="User:", font=("Arial", 22, "bold")).grid(row=0, column=0, padx=5, pady=5, sticky="e")
entry_user = tk.Entry(frame_form, font=("Arial", 22, "bold"))
entry_user.grid(row=0, column=1, padx=5, pady=5, sticky="ew")
lock_button_user = tk.Button(frame_form, text="Lock", font=button_font, command=lambda: toggle_lock(entry_user, lock_button_user))
lock_button_user.grid(row=0, column=2, padx=5, pady=5)

tk.Label(frame_form, text="Location:", font=("Arial", 22, "bold")).grid(row=1, column=0, padx=5, pady=5, sticky="e")
entry_location = tk.Entry(frame_form, font=("Arial", 22, "bold"))
entry_location.grid(row=1, column=1, padx=5, pady=5, sticky="ew")
lock_button_location = tk.Button(frame_form, text="Lock", font=button_font, command=lambda: toggle_lock(entry_location, lock_button_location))
lock_button_location.grid(row=1, column=2, padx=5, pady=5)

tk.Label(frame_form, text="Barcode:", font=("Arial", 22, "bold")).grid(row=2, column=0, padx=5, pady=5, sticky="e")
entry_barcode = tk.Entry(frame_form, font=("Arial", 22, "bold"))
entry_barcode.grid(row=2, column=1, padx=5, pady=5, sticky="ew")
```


Cela devrait ressembler à ça :



The screenshot shows a Tkinter window with a light blue background. Inside, there is a frame containing three rows of input fields and buttons. The first row is labeled 'User:', the second 'Location:', and the third 'Barcode:'. Each label is in a large, bold, black font. To the right of each label is a text entry field. To the right of each entry field is a button labeled 'Lock'. The buttons are in a smaller, bold, black font. The entire form is enclosed in a light gray border.

A chaque fois que le programme est exécuter un nouveau fichier doit être créer, cela est possible avec cette ligne :

```
# Generate unique file with date and hour
CSV_FILE = f"data_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
table_font_size = 13
```

 data_20250428_213844



28/04/2025 22:22

Fichier CSV Micro...

1 Ko

Pour mieux distinguer réajuster la taille de la police :

```
def adjust_text_size(adjustment):
    global table_font_size
    table_font_size += adjustment
    if table_font_size < 8:
        table_font_size = 8

    style.configure("Treeview", font=("Arial", table_font_size))
    style.configure("Treeview.Heading", font=("Arial", table_font_size + 2))
    tree.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)
```

Création du bouton Lock/Unlock pour verrouiller ou pas la zone de texte :

```
def toggle_lock(entry, button):
    if entry.cget("state") == "normal":
        entry.config(state="disabled")
        button.config(text="Unlock")
    else:
        entry.config(state="normal")
        button.config(text="Lock")
```

Le programme doit charger les données entrées dans la zone positionnée juste en bas de la zone de texte, du plus récent au moins récent :

```
def load_data():
    tree.delete(*tree.get_children())
    if os.path.exists(CSV_FILE):
        with open(CSV_FILE, mode="r", newline="", encoding="utf-8") as file:
            reader = csv.reader(file)
            next(reader, None) # Skip header

            # Read & sort data by descending date (grand -> petit)
            data = sorted(reader, key=lambda row: datetime.strptime(row[3], "%Y-%m-%d %H:%M:%S"), reverse=True)

            for row in data:
                tree.insert("", "end", values=(row[4], row[5]))
```

Cela doit donner :

User:	<input type="text"/>	Lock
Location:	<input type="text"/>	Lock
Barcode:	<input type="text"/>	

Barcode	Quantity
1475	1
14587	1

Maintenant le programme doit être sûr que chaque zone soit rempli avant de valider, les informations sera alors ranger dans un certain l'ordre dans le fichier csv :

```
def add_data():
    user = entry_user.get()
    location = entry_location.get()
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    barcode = entry_barcode.get()
    quantity = 1
    alu = entry_barcode.get()

    if not user or not location or not barcode:
        messagebox.showwarning("Missing fields", "Please fill in all fields.")
        return

    exist_file = os.path.exists(CSV_FILE)

    with open(CSV_FILE, mode="a", newline="", encoding="utf-8") as file:
        writer = csv.writer(file)
        if not exist_file: # Write title if it is a new file
            writer.writerow(["ID", "User", "Location", "Timestamp", "Barcode", "Quantity", "Alu"])
        writer.writerow([tree.get_children().__len__() + 1, user, location, timestamp, barcode, quantity, alu])

    load_data() # Load data to display correctly in the table
    entry_user.delete(0, tk.END)
    entry_location.delete(0, tk.END)
    entry_barcode.delete(0, tk.END)
```

Dans le fichier Excel l'ordre des informations sera comme cela :

ID, User, Location, Timestamp, Barcode, Quantity, Alu

Maintenant la création des boutons zoom +/- pour agrandir ou pas les informations :

```
tk.Button(root, text="Zoom +", font=button_font, command=lambda: adjust_text_size(2)).grid(row=5, column=0, pady=10)
tk.Button(root, text="Zoom -", font=button_font, command=lambda: adjust_text_size(-2)).grid(row=5, column=1, pady=10)
```

Cela ressemble à ça :

Zoom +	Zoom -
---------------	---------------

Maintenant création du bouton quitter :

```
# Button (leave) displays the verification areas and checks before leaving
leave_button = tk.Button(root, text="Leave", font=button_font, bg="#ff4d4d", fg="white", command=show_check_fields)
leave_button.grid(row=5, column=2, pady=10)
```

Une fois cliquer une zone de vérification apparaît :

User:	<input type="text" value="pierre"/>	<input type="button" value="Unlock"/>	<input type="text"/>
Location:	<input type="text" value="shop"/>	<input type="button" value="Unlock"/>	<input type="text"/>
Barcode:	<input type="text"/>		

Ce morceau de code permettra de vérifier si les informations correspondent :

```
def check_leave():
    """Verify the fields and change their color"""
    user_check = entry_check_user.get().strip()
    location_check = entry_check_location.get().strip()

    user_correct = (user_check == entry_user.get().strip())
    location_correct = (location_check == entry_location.get().strip())

    # Change the color according to the match
    entry_check_user.config(bg="lightgreen" if user_correct else "red")
    entry_check_location.config(bg="lightgreen" if location_correct else "red")

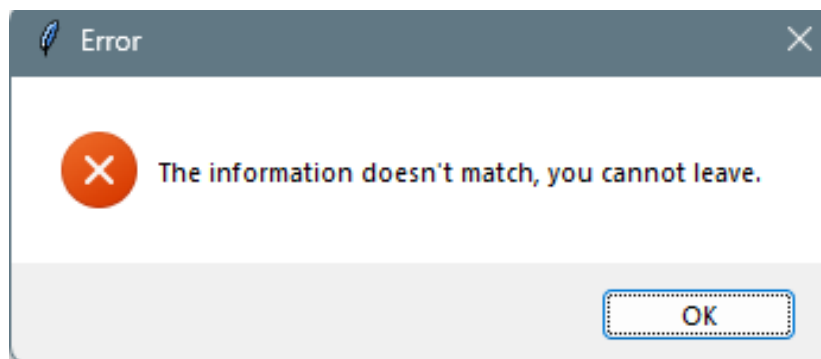
    if user_correct and location_correct:
        root.bind('<Return>', lambda event: root.quit()) # Press enter key for leaving
    else:
        root.unbind('<Return>') # Desactivate enter key if there is an error
        messagebox.showerror("Error", "The information doesn't match, you cannot leave.")
```

Si les infos correspondent alors vert :

User:	<input type="text" value="pierre"/>	<input type="button" value="Unlock"/>	<input type="text" value="pierre"/>
Location:	<input type="text" value="shop"/>	<input type="button" value="Unlock"/>	<input type="text" value="shop"/>
Barcode:	<input type="text"/>		

Sinon :

User:	<input type="text" value="pierre"/>	<input type="button" value="Unlock"/>	<input type="text" value="piere"/>
Location:	<input type="text" value="shop"/>	<input type="button" value="Unlock"/>	<input type="text" value="shop"/>
Barcode:	<input type="text"/>		



Si c'est donc vert alors le programme se ferme

A notifier que si on bloque la zone user et location et que un scanner est connecté à l'appareil utilisant le programme alors le code barre se mettra automatiquement et se validera également tous seul.

CONCLUSION

Retombées du projet

Ce programme peut permettre de scanner rapidement les stocks et de retrouver les produits également. Ce programme sera utilisé dans l'entreprise plus tard.
