MINI-PROJET N°1

1. Cahier des charges

Objectif :

→ Créer une interface graphique pour le passage d'un QCM, comme celui créé dans l'ACT précédente.

Fonctionnalités demandées :

- *Titre du QCM (texte)*
- *Explication du fonctionnement du QCM (texte)*
- *Image illustrative (image)*
- *Bouton pour commencer (bouton)*
- *Proposition des réponses à chaque question (radio)*
- *Bouton de validation à chaque question (bouton)*
- *Affichage du score final (texte)*

Fonctionnalités extras (facultatives) :

- *Affichage de la correction (texte)*
- *Bouton pour recommencer (bouton)*

Compétences travaillées :

APP	• Mobiliser les concepts et les technologies adaptés au problème
	• Rechercher l'information utile à l'aide de sources fiables
REA	• Mettre en œuvre une solution, par la traduction d'un algorithme dans un langage de programmation.
	• Imaginer et concevoir une solution, décomposer en blocs, se ramener à des sous-problèmes simples et indépendants, adopter une stratégie appropriée

2. Les débuts avec le module tkinter

Dans cette partie, nous allons apprendre à réaliser une GUI (ou *Graphical User Interface*). Dans python, il y a plusieurs moyen de s'y prendre ; ici nous allons nous intéresser à tkinter.

1. Widgets et mainloop

Les *widgets* (ou *window gadget*) sont des objets graphiques permettant à l'utilisateur d'interagir avec votre programme Python de manière conviviale. Par exemple, ils peuvent être des boutons, des listes de choix, ou encore la zone de texte.

```python
import tkinter as tk

racine = tk.Tk()
label = tk.Label(racine, text="J'adore Python !")
bouton = tk.Button(racine, text="Quitter", command=racine.destroy)
bouton["fg"] = "red"
label.pack()
bouton.pack()
racine.mainloop()
print("C'est fini !")
```

tkinter_ex1.py

Ligne 1 : Import du module tkinter
Ligne 3 : Création de la fenêtre de base racine, comme un objet tkinter
Ligne 4 : Création d'un widget (ou objet spécifique) label ou titre, associé à la fenêtre racine
Ligne 5 : Création d'un widget bouton, avec un texte et une commande, associé à la fenêtre racine.
Ligne 6 : Précision de l'option de rendu du widget bouton. Syntaxe générale widget["option"] = valeur
Ligne 7&8 : Collage des widgets sur la fenêtre racine, avec l'instruction .pack()
Ligne 9 : Lancement du gestionnaire d'événements avec racine.mainloop().

`racine.mainloop()`
C'est le gestionnaire d'évènement qui intercepte la moindre action de l'utilisateur, et qui lance les portions de code associées à chacune de ses actions. Cette instruction est à la fin du script, puisque, on écrit d'abord le code construisant l'interface, et on lance le gestionnaire d'événements une fois l'interface complètement décrite, ce qui lancera au final l'application.
`racine.destroy()`
Pour quitter l'application, on utilise ici la méthode `.destroy()`. Celle-ci casse la `.mainloop()` et arrête ainsi le gestionnaire d'événements. Cela mène à l'arrêt de l'application.

2. Widget Canvas

Le widget canvas de Tkinter est très puissant. Il permet de dessiner des formes diverses (lignes, cercles, etc.), de coller des photos, des images, … et même de tout animer !

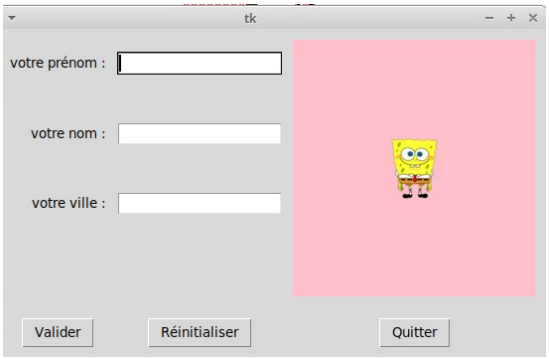
```python
import tkinter as tk

racine = tk.Tk()
canv = tk.Canvas(racine, bg="white", height=200, width=200)
canv.pack()
canv.create_oval(0, 0, 200, 200, outline="red", width=10)
canv.create_line(0, 0, 200, 200, fill="black", width=10)
canv.create_line(0, 200, 200, 0, fill="black", width=10)
racine.mainloop()
```

tkinter_ex2.py

Ligne 4 : Création du widget canvas associé à la fenêtre racine, avec ses dimensions et sa couleur de fond
Ligne 5 : Collage du canvas créé
Ligne 6-8 : Dessins dans le canvas

3. Exemple d'une fenêtre de saisie

Une fenêtre vide (démarrage de l'application ou réinitialisation) permet de saisir nom, prénom et ville.
La validation affiche les trois valeurs saisie.



```python
from tkinter import *

racine = Tk()

# les labels
label1 = Label(racine, text="votre prénom :")
label1.grid(row=1, column=1, sticky="E", padx=10)
label2 = Label(racine, text="votre nom :")
label2.grid(row=2, column=1, sticky="E", padx=10)
label3 = Label(racine, text="votre ville :")
label3.grid(row=3, column=1, sticky="E", padx=10)

labelValider = Label(racine, text="") # label de la chaîne de validation
labelValider.grid(row=4, column=1, columnspan=2, sticky="W",padx=10)

# les entrées
entree1 = Entry(racine)
entree1.grid(row=1, column=2)
entree1.focus_set()
entree2 = Entry(racine)
entree2.grid(row=2, column=2)
entree3 = Entry(racine)
entree3.grid (row=3, column=2)

# le cannevas et son image (232x245)
cannevasImg = Canvas(racine, width=245, height=258, bg="pink")
photo = PhotoImage(file="bob.png")
image = cannevasImg.create_image(124, 131, image=photo)
cannevasImg.grid(row=1, column=3, rowspan=4, padx=10, pady=10)
```

```python

# les boutons et leur gestion
def valider () :
    chaine = entree1.get()+" // "+ entree2.get()+ " // "+ entree3.get()
    labelValider.config(text=chaine)
def initialiser ():
    labelValider.config(text="")
    entree3.delete (0, END)
    entree2.delete (0, END)
    entree3.delete (0, END)
    entree1.focus_set()

boutonQuitter = Button(racine, text="Quitter", command=racine.destroy)
boutonQuitter.grid(row=5, column=3, pady=10)
boutonValider = Button(racine, text="Valider", command=valider)
boutonValider.grid(row=5, column=1, pady=10)
boutonInitialiser = Button(racine, text="Réinitialiser", command=initialiser)
boutonInitialiser.grid(row=5, column=2, pady=10)

racine.mainloop()
```

tkinter_ex3.py

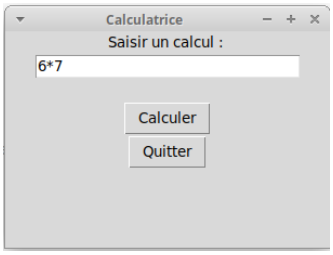
La grille (ou grid) est disposée de la façon suivante :

<table>
<tr><td>Label1</td><td>entree1</td><td rowspan="4">Cannevas + photo</td></tr>
<tr><td>Label2</td><td>entree2</td></tr>
<tr><td>Label3</td><td>entree3</td></tr>
<tr><td colspan="2">labelValider</td></tr>
<tr><td>BoutonValider</td><td>boutonInitialiser</td><td>boutonQuitter</td></tr>
</table>

`photo = PhotoImage (file="bob.png")`
Cette instruction créer une représentation interne de la photo. Cette représentation permet l'accès aux paramètres de la photo.
`image = cannevasImg.create_image(124, 131, image = photo)`
Cette instruction dessine l'image sur le cannevas en mettant son centre aux coordonnées (124, 131) du cannevas. Il n'y a pas redimensionnement de l'image.
`entree3.get()`
La méthode get() d'un objet de la classe Entry retourne la chaîne frappée dans le widget.
`entree3.delete (0, END)`
La méthode delete(n, p) efface les caractères entre les emplacement n (inclus) et p. Si on remplace la seconde coordonnée par l'identificateur prédéfini END, toute la chaîne est effacée.
`entree1.focus_set()`
Cette instruction utilise une méthode qui donne le focus au widget qualifié. Ici, le curseur de saisie de texte est actif dans l'entrée du haut.
`config(text = chaine)`
La méthode config() permet de changer la valeur d'un attribut du widget qu'elle qualifie (les attributs sont les paramètres utilisables lors de la réation d'un widget).
`sticky = "E"`
Le widget est placé dans la case du tableau en utilisant comme repères les points cardinaux les valeurs possibles sont N, S, E, W (bords) NE, NW, SE, SW (coins).
`chaine = entree1.get()+" // "+ entree2.get()+ " // "+ entree3.get()`
Cette instruction utilise les saisies des 3 entrées et les concatène en une variable chaine.

4. Exemple d'une calculatrice

Une interface de saisie de calcul simple.



```python
from tkinter import *

racine = Tk()
racine.title("Calculatrice")
racine.minsize(300,200)

# Titre (Label)
titre = Label(racine, text="Saisir un calcul : ")
titre.pack()
# Frame (cadre)
cadre = Frame(racine)
cadre.pack()

# Saisie de l'expression mathématique : Entrée (Entry)
expression = StringVar()
expression.set("6*7")   # texte par défaut affiché dans l'entrée
entree = Entry(cadre, textvariable=expression, width=30)
entree.pack()

# Résultat du calcul
resultat = StringVar()
sortie = Label(cadre, textvariable=resultat)
sortie.pack()

def calculer():
   resultat.set(eval(expression.get()))

# Bouton pour exécuter les calculs
bouton = Button(cadre, text="Calculer", command=calculer)
bouton.pack()

quitter = Button(racine, text="Quitter", command=racine.destroy)
quitter.pack()
racine.mainloop()
```

tkinter_ex4.py

5. Autres widgets utiles

Nous avons vu les widgets Button, Canvas, Label, Entry mais il en existe bien d'autres. En voici une liste avec une brève explication pour chacun :

- **Checkbutton :** affiche des cases à cocher.
- **Listbox :** affiche une liste d'options à choisir (comme dans la figure 1).
- **Radiobutton :** implémente des « boutons radio ».
- **Menubutton** et **Menu** : affiche des menus déroulants.
- **Message :** affiche un message sur plusieurs lignes (extensions du widget Label).
- **Scale :** affiche une règle graduée pour que l'utilisateur choisisse parmi une échelle de valeurs.
- **Scrollbar :** affiche des ascenseurs (horizontaux et verticaux).
- **Text :** crée une zone de texte dans lequel l'utilisateur peut saisir un texte sur plusieurs lignes (comme dans la figure 1).
- **Spinbox :** sélectionne une valeur parmi une liste de valeurs.
- **tkMessageBox :** affiche une boîte avec un message.

Ressources autres :

https://python.doctor/page-tkinter-interface-graphique-python-tutoriel

https://docs.python.org/fr/3/library/tkinter.html