Interrogation d'informatique
2ème année - Mars 2023

Durée totale : *60mn*
Documents autorisés : *Synthèse d'une ou 2 pages.*

INSA

- Le barème est sur 20 points.
- Le sujet est sur 4 pages - il y a 2 exercices.

Exercice 1 Générateur de mots de passe (10 pts)

Nous souhaitons créer une interface graphique permettant de générer des mots de passe de manière aléatoire en fonction des choix de l'utilisateur. L'allure souhaitée pour l'interface graphique est celle présentée dans la Figure 1.

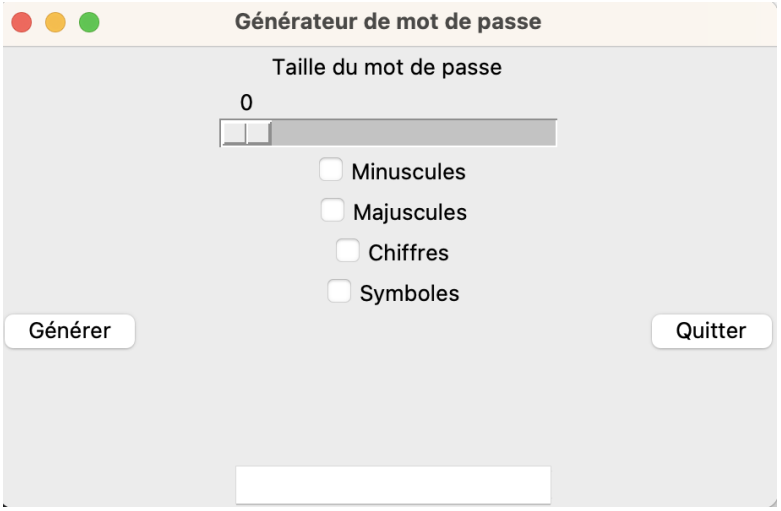

FIGURE 1 – Allure attendue de l'interface graphique

L'interface graphique comprend une étiquette indiquant la "Taille du mot de passe" à générer, une glissière pour choisir la longueur du mot de passe (en haut), des boutons pour "Générer" le mot de passe (à gauche) ou "Quitter" l'application (à droite), un champ de texte (tk.Entry) pour afficher le mot de passe généré (en bas) et des cases à cocher pour sélectionner les types de caractères qui doivent être inclus dans le mot de passe (minuscules, majuscules, chiffres, symboles). On rappelle que la variable de contrôle tkinter associée à une case à cocher vaut "1" si elle est cochée et "0" sinon.

Lorsqu'on clique sur le bouton "Générer", une fonction est appelée pour récupèrer les choix de l'utilisateur sur la longueur du mot de passe et les types de caractères à inclure, puis génère un mot de passe aléatoire et l'affiche dans le champ de texte. Lorsqu'on clique sur le bouton "Quitter", une autre fonction est appelée pour détruire la fenêtre principale de l'interface graphique et ferme l'application.

Le programme informatique sera implémenté dans une classe que nous nommerons *FenetreMDP* dont voici le constructeur :

(Q1.1) ***Écrire la fonction creer_widgets de la classe FenetreMDP permettant de créer les widgets, les paramétrer et les placer selon la configuration de la Figure 1.***

```python
class FenetreMDP():
    def __init__(self):
        self.racine = tk.Tk()
        self.racine.title("Generateur de mot de passe")

        # Longueur du mot de passe : Variable liee a la glissiere, initialisee a 0
        self.tailleMDP = tk.IntVar(0)

        # Variables liees aux cases a cocher, initialisees e 0
        self.choixMin = tk.IntVar(0)
        self.choixMaj = tk.IntVar(0)
        self.choixChif= tk.IntVar(0)
        self.choixSymb= tk.IntVar(0)

        # Chaines de caracteres correspondant a chaque choix
        self.minuscules = "abcdefghijklmnopqrstuvwxyz"
        self.majuscules = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        self.chiffres = "0123456789"
        self.symboles = "&~#{([-|_\^@)=+$]}*%!/:.;?,"

        # Creation et placement des widgets
        self.creer_widgets()
```

```python
    def creer_widgets(self):
        # Le label
        self.etiquette = tk.Label(self.racine,text="Password length")
        self.etiquette.pack(side=tk.TOP)

        # La glissière
        self.scale = tk.Scale(self.racine, orient='horizontal', from_=0,
        to=20, length=200, variable = self.tailleMDP)
        self.scale.pack(side=tk.TOP)

        # Les boutons
        self.boutonGenerer = tk.Button(self.racine, text="Generate")
        self.boutonGenerer.pack(side=tk.LEFT)
        self.boutonQuitter = tk.Button(self.racine, text="Quit")
        self.boutonQuitter.pack(side=tk.RIGHT)

        # Le champ de texte
        self.champTexte = tk.Entry(self.racine)
        self.champTexte.pack(side=tk.BOTTOM)

        # Les cases à cocher
        self.caseMinuscules = tk.Checkbutton(self.racine, text="Lowercase",
        variable = self.choixMin)
        self.caseMinuscules.pack()
        self.caseMajuscules = tk.Checkbutton(self.racine, text="Uppercase",
        variable = self.choixMaj)
        self.caseMajuscules.pack()
        self.caseChiffres = tk.Checkbutton(self.racine, text="Digits",
        variable = self.choixChif)
```

```
        self.caseChiffres.pack()
        self.caseSymboles = tk.Checkbutton(self.racine, text="Symbols",
        variable = self.choixSymb)
        self.caseSymboles.pack()
```

(Q1.2) Écrire la fonction genererMotDePasse "gestionnaire d'événement (callback)" de la classe FenetreMDP permettant de récupérer les choix de l'utilisateur, de générer le mot de passe associé et de l'afficher dans le champ de texte. Selon les choix de l'utilisateur, la fonction devra construire un mot de passe en tirant aléatoirement (math.randint) parmi une ou plusieurs de ces chaines de caractères.

```
    def genererMotDePasse(self,event):
        self.champTexte.delete(0, tk.END)
        taille=self.tailleMDP.get()
        ch_minuscules=(self.choixMin.get()==1)
        ch_majuscules=(self.choixMaj.get()==1)
        ch_chiffres=(self.choixChif.get()==1)
        ch_symboles=(self.choixSymb.get()==1)
        caracteres = ""
        motDePasse = ""
        if ch_minuscules:
            caracteres += self.minuscules
        if ch_majuscules:
            caracteres += self.majuscules
        if ch_chiffres:
            caracteres += self.chiffres
        if ch_symboles:
            caracteres += self.symboles

        if len(caracteres) != 0:
            for i in range(taille):
                motDePasse += caracteres[randint(0,len(caracteres)-1)]
            self.champTexte.insert(tk.INSERT,motDePasse)
```

(Q1.3) Écrire la fonction quitter "gestionnaire d'événement (callback)" de la classe FenetreMDP permettant de détruire la fenêtre principale de l'interface graphique et de quitter l'application.

```
    def quitter(self,event):
        self.racine.destroy()
```

(Q1.4) Préciser quelles instructions sont à ajouter et où pour déclencher les évènements associés au clic gauche sur les boutons "Générer" et "Quitter".

```
    def creer_widgets(self):
        # Les boutons
        self.boutonGenerer.bind('<Button-1>',self.genererMotDePasse)
        self.boutonQuitter.bind('<Button-1>',self.quitter)
```

Exercice 2 Jeu de fléchettes (10 pts)

FIGURE 2 – Allure attendue du jeu de fléchettes

Nous souhaitons programmer une interface graphique avec un widget canvas. Quand l'utilisateur clique sur le bouton "Tirer" un cercle de couleur noire de rayon 10 pixels est dessiné dans le canvas, à un emplacement aléatoire.

L'utilisateur peut effacer le dernier cercle dessiné en cliquant sur le bouton "Effacer le dernier tir" ou effacer tous les cercles en cliquant sur le bouton "Effacer tout". L'utilisateur peut également effacer un cercle dessiné en cliquant avec le bouton gauche de la souris sur ce cercle.

L'allure souhaitée pour l'interface graphique est celle présentée dans la Figure 2.

Une ébauche du programme informatique associé est donné dans la classe *Fenetre_avec_graphique* à la page 4.

(Q2.1) Donner le code à ajouter dans la fonction appelée Cercle pour qu'elle dessine un cercle de rayon 10 pixels et dont la position du centre est définie aléatoirement (math.randint) dans la zone graphique donnée par le Canevas.

```
def Cercle(self,event):
    """ Dessine un cercle de centre (x,y) et de rayon r """
    x = random.randint(0,self.Largeur)
    y = random.randint(0,self.Hauteur)
    r = 10
    # on dessine un cercle dans la zone graphique
    item = self.Canevas.create_oval(x-r, y-r, x+r, y+r, fill='black')
```

```
        # on ajoute l'item dans la liste
        self.ListeItemCercles.append(item)
```

N.B. L'identifiant de ce cercle devra être récupéré et conservé dans la liste *ListeItemCercles*. Par ailleurs, les flèches peuvent toucher les bords et les cercles peuvent donc ne pas être affichés complètement.

(Q2.2) Donner le code à ajouter dans la fonction appelée Undo pour qu'elle supprime le dernier cercle dessiné dans la zone graphique.

```
def Undo(self,event):
    """ Efface le dernier cercle"""
    if self.ListeItemCercles != []:
        item = self.ListeItemCercles[-1]
        # on efface le cercle
        self.Canevas.delete(item)
        # on supprime l'item de la liste
        self.ListeItemCercles.pop(-1)
```

N.B. On rappelle que la fonction *list.pop(index)* avec l'argument index supprime et renvoie l'élément à l'index de position.

(Q2.3) Donner le code à ajouter dans la fonction appelée EffacerTout pour qu'elle supprime tous les cercles dessinés dans la zone graphique.

```
def EffacerTout(self,event):
    """ Efface tous les cercles"""
    for item in self.ListeItemCercles:
        self.Canevas.delete(item)
    self.ListeItemCercles.clear()
```

(Q2.4) Préciser quelles instructions sont à ajouter et où pour déclencher l'évènement associé au clic gauche sur le Canevas pour supprimer un cercle avec la fonction UndoManuel.

```
def Cercle(self,event):
    """ Dessine un cercle de centre (x,y) et de rayon r """
    self.Canevas.tag_bind(item, '<Button-1>', self.UndoManuel)
```

(Q2.5) Écrire une fonction UndoManuel qui permette de supprimer un cercle sélectionné par un clic gauche sur la zone graphique du Canevas par l'utilisateur.

```
def UndoManuel(self,event):
    """ Efface un cercle en mode manuel"""
    if self.ListeItemCercles != []:
        item_clicked = self.Canevas.find_withtag("current")[0]
        # on efface le cercle
        self.Canevas.delete(item_clicked)
        self.ListeItemCercles.remove(item_clicked)
```

```
# ============================================================================
#             Solution alternative
# ============================================================================
            # index de l'item
            # index=0
            # while self.ListeItemCercles[index] != item_clicked and index < len(self.Li
            #     index += 1
            # # on supprime l'item de la liste
            # self.ListeItemCercles.pop(index)
```

```
class Fenetre_avec_graphique():

    def __init__(self):
        # Creation de la fenetre principale
        self.racine = tk.Tk()
        self.racine.title('Cible')

        # Initialisation de la liste des items des cercles
        self.ListeItemCercles = []

        # Image de fond
        self.photo = tk.PhotoImage(file="tk_cible.gif", master=self.racine)

        # Creation d'un widget Canvas (zone graphique)
        self.Largeur = self.photo.width()
        self.Hauteur = self.photo.height()

        self.creer_widgets()

    def creer_widgets(self):

        self.Canevas = tk.Canvas(self.racine,width = self.Largeur, height =self.Hauteur)
        self.Canevas.create_image(0,0,anchor=tk.NW, image=self.photo)
        self.Canevas.pack()

        # Creation d'un widget Button
        self.BoutonGo = tk.Button(self.racine, text ='Tirer')
        self.BoutonGo.bind('<Button-1>',self.Cercle)
        self.BoutonGo.pack(side = tk.LEFT, padx = 10, pady = 10)

        # Creation d'un widget Button
        self.BoutonEffacer = tk.Button(self.racine, text ='Effacer le dernier tir')
        self.BoutonEffacer.bind('<Button-1>',self.Undo)
        self.BoutonEffacer.pack(side = tk.LEFT, padx = 10, pady = 10)

        # Creation d'un widget Button
        self.BoutonEffacerTout = tk.Button(self.racine, text ='Effacer tout')
        self.BoutonEffacerTout.bind('<Button-1>',self.EffacerTout)
        self.BoutonEffacerTout.pack(side = tk.LEFT, padx = 10, pady = 10)

    def Cercle(self,event):
        """ Dessine un cercle de centre (x,y) et de rayon r """

    def Undo(self,event):
        """ Efface le dernier cercle"""

    def UndoManuel(self,event):
        """ Efface un cercle en mode manuel"""

    def EffacerTout(self,event):
        """ Efface tous les cercles"""

app=Fenetre_avec_graphique()
app.racine.mainloop()
```