TP5 bis POO et Tkinter

OBJECTIFS : L'objectif principal de ce TP est d'utiliser la Programmation Orientée Objet en concordance avec la bibliothèque graphique Tkinter. La POO permet d'éviter l'utilisation de variables globales et permet une meilleure maintenance ainsi que plus de facilités pour enrichir un projet. Nous n'aborderons pas la notion d'héritage. Bien que très employée dans la programmation professionnelle car elle permet de créer des objets spécialisés à partir d'un objet de base elle n'est pas au programme.

L'évaluation de ce travail est basée sur le rendu du fichier *.py* qui sera constitué. **On ne demande pas** de constituer de fichier texte contenant des copies d'écran.

⇨ Pour débuter, télécharger le dossier *tp5.zip* contenant les 27 fichiers images qui seront utilisés. Le dézipper dans votre espace personnel sur ***U:*** . Ouvrir un nouveau fichier *.py* dans le dossier contenant ces images et le sauvegarder sous le nom ***tp5.py*** .

1. PREPARER SON FICHIER .py :

⇨ Organiser son fichier en créant un objet Ma_fenetre

- qui contiendra des attributs correspondants aux objets des classes de Tkinter:
  - ***Tk*** : pour pouvoir créer une fenêtre Tkinter,
  - ***Canvas*** : pour pouvoir créer dans cette fenêtre, une zone contenant des images, formes géométriques, ….
  - ***Label*** : pour pouvoir créer dans cette fenêtre, des zones textes,
  - ***Text*** : pour pouvoir y créer des champs de saisie,
  - ***Button*** : pour pouvoir y créer des boutons.qui contiendra des méthodes associées à ces objets par de fonctions callback.
- Un programme principal qui appelle Ma_fenetre

Le projet utilisera aussi :

- La bibliothèque *PIL, les classes Image et ImageTk* pour pouvoir manipuler des images,
- La bibliothèque *random, la fonction randint* pour pouvoir générer des nombres aléatoires.

2. CREER L'OBJET FENETRE DANS TKINTER:

⇨On commence par créer une classe vide pour définir l'objet Ma_fenetre.

```
1  # Modules ##########################################
2  from tkinter import Canvas, Tk,Label,Text, Button
3  from PIL import Image, ImageTk #pip install pillow
4  from random import randint
5
6
7  # Création d'un objet fenêtre ( vide au début)
8  class Ma_fenetre() :
9      # attributs
10     def __init__(self) :
11         # On crée un attribut fenetre
12         self.fen = Tk()
13         self.fen.title("TP11_poo")
14
15
16     # lancement du gestionnaire de fenetre
17         self.fen.mainloop()
18
19     # méthodes
20
21 # Main  ########################################
22 if __name__ == "__main__":
23     # ici débute le programme principal
24     # Création d'une instance de la fenetre principale
25     Ma_fenetre()
```

Ligne 12 : l 'objet Ma_fenetre possède un attribut fen qui est du type fenêtre de Tkinter

Line 13 : on donne un titre à la fenêtre donc **self**.fen.title()

Ligne 15 : on lance le gestionnaire d'évènement sur cet attribut donc **self.**fen.mainloop()

Ligne 27: le programme principal se contente d'instancier un unique objet de type Ma_fenetre

Remarques importantes

- **self.fen** est un attribut de l'objet Ma_fenêtre, il sera donc reconnu par toutes les futures méthodes de cet objet. Il n'y a donc pas besoin de définir une variable globale pour y faire référence dès que l'objet Ma_fenetre est instancié.
- bien mettre le lancement du gestionnaire d'événement en fin de déclaration des widgets et avant les méthodes (l'objet doit être entièrement formé avant de pouvoir le manipuler)

⇨ Exécuter le fichier

3. CREER DES WIDGETS dans cette fenêtre tkinter :

⇨ Ajouter les widgets comme des attributs de l'objet

```python
# Modules ########################################
from tkinter import Canvas, Tk,Label,Text, Button
from PIL import Image, ImageTk #pip install pillow
from random import randint


# Création d'un objet fenêtre ( vide au début)
class Ma_fenetre() :
    # attributs
    def __init__(self) :
        # On crée un attribut fenetre
        self.fen = Tk()
        self.fen.title("TP11_poo")

        #création des widjets
        self.zone_graphique  = Canvas(self.fen,width= 1000, height = 600, bg="black")
        self.zone_graphique.grid(row = 0, column = 0,columnspan = 3)


    # lancement du gestionnaire de fenetre
        self.fen.mainloop()

    # méthodes

# Main  ########################################
if __name__ == "__main__":
    # ici débute le programme principal
    # Création d'une instance de la fenetre principale
    Ma_fenetre()
```

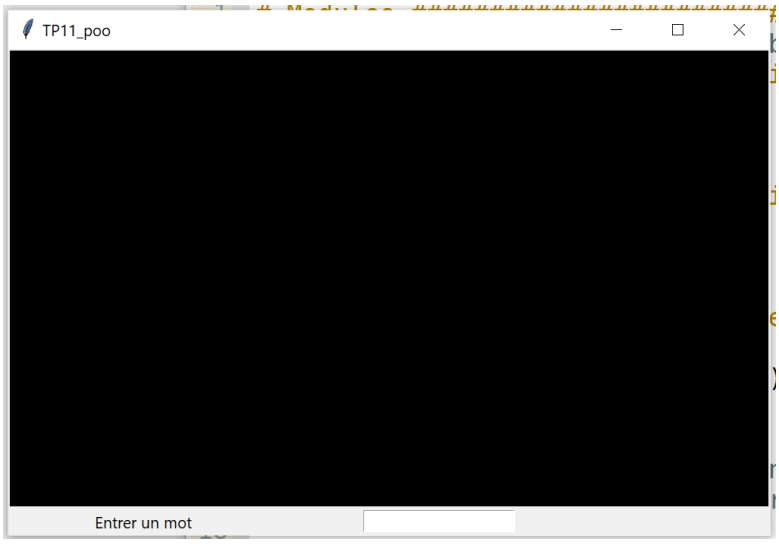
⇨ Exécuter ce fichier.

Pour l'instant on crée seulement une zone graphique de 1000 px de large par 600 px de haut, avec un fond d'écran noir. Cette zone est un objet de la classe *Canvas* qui est stocké ici dans la variable nommée *zone_graphique*. Cet objet est positionné dans la fenêtre *Tkinter* avec la méthode *grid()* sur la 1ère colonne (column = 0) et la 1ère ligne (row = 0). On indique que cette colonne sera large et qu'elle occupera la largeur des 3 colonnes qui seront placées ensuite (columnspan = 3).

Continuons l'ajouts de widgets avec un Label et un Text:

```python
# Modules ########################################
from tkinter import Canvas, Tk,Label,Text, Button
from PIL import Image, ImageTk #pip install pillow
from random import randint


# Création d'un objet fenêtre ( vide au début)
class Ma_fenetre() :
    # attributs
    def __init__(self) :
        # On crée un attribut fenetre
        self.fen = Tk()
        self.fen.title("TP11_poo")

        #création des widjets
        self.zone_graphique  = Canvas(self.fen,width= 1000, height = 600, bg="black")
        self.zone_graphique.grid(row = 0, column = 0,columnspan = 3)

        self.mon_texte = Label(self.fen, text = "Entrer un mot")
        self.mon_texte.grid(row = 1, column=0)

        self.champ_saisie = Text(self.fen, height=1,width= 14)
        self.champ_saisie.grid(row= 1, column= 1)

    # lancement du gestionnaire de fenetre
        self.fen.mainloop()

    # méthodes

# Main  ########################################
if __name__ == "__main__":
    # ici débute le programme principal
    # Création d'une instance de la fenetre principale
    Ma_fenetre()
```

⇨ Exécuter ce fichier et vérifier que l'on obtient :



Cela sans variables globales car tous les widgets sont des attributs du seul objets instancié.

⇨ Compléter une dernière fois le script de l'objet Ma_fenetre , en insérant sur la seconde ligne (*row = 1*) et la troisième colonne (*column = 2*) un bouton qui permettra à l'utilisateur de créer un ***évènement***. Ce champ de saisi est un objet de la classe *Button*, qui sera stocké dans la variable *bouton_valider* .

Pour associer une fonctionnalité au bouton on définit une méthode **debut (self)** equi est appelée à chaque fois que l'on clique sur le bouton. Cette méthode utilise une fonction « callback » en anglais. Elle est associée à un évènement au « clique » sur le bouton

```
    self.bouton_valider= Button(self.fen, text = "Valider" , width =12 , command =  self.debut)
    self.bouton_valider.grid(row=1, column=2)

# lancement du gestionnaire de fenetre
    self.fen.mainloop()

# méthodes
def debut(self):
    print("Tu as cliqué sur le bouton")
    mot =  self.champ_saisie.get("1.0", "end-1c")
    print("Le texte entré est :",  mot)
```

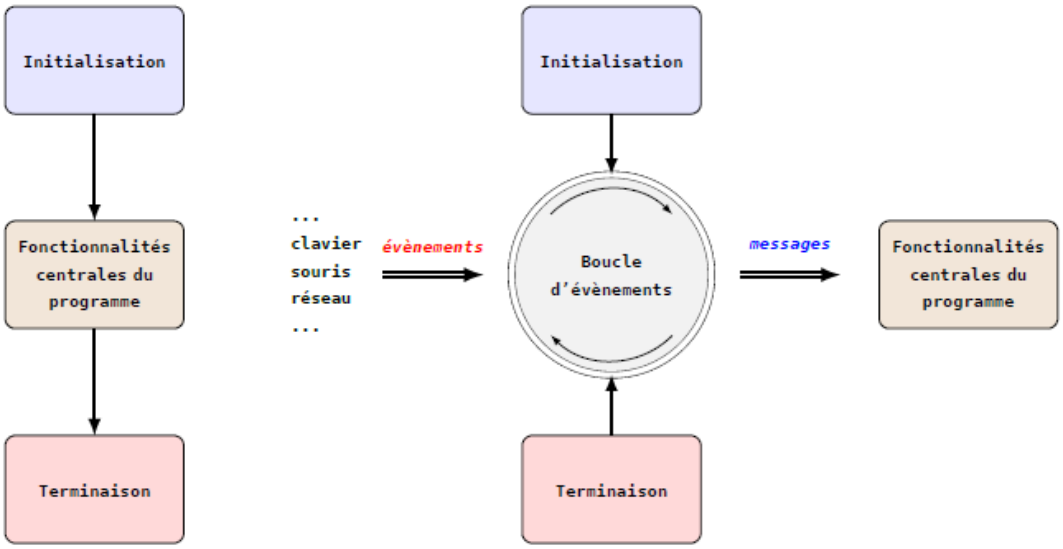
⇨ Exécuter ce fichier et tester le bon fonctionnement du champ de saisie et du bouton.

La fenêtre Tkinter contient ainsi 4 widgets. Le widget principal est le canvas qui permettra dans le tp suivant, d'accueillir des éléments graphiques. Ce canvas a une largeur de 1000 px et une hauteur de 600 px :

INFOS importantes :

4. A RETENIR:

L'application utilise la bibliothèque graphique Tkinter une **GUI (Graphical User Interface).**



Cette bibliothèque remplace la programmation séquentielle traditionnelle par la prise en compte d'évènements qui provoquent des actions.

- La fenêtre principale est un objet défini par sa classe class
  - Les widgets empruntés à Tkinter sont les attributs de cet objet
  - Les fonctions callback associés aux widgets permettant une interaction de l'application sont programmés dans les méthodes de la fenêtre principale
  - ainsi les attributs et méthodes sont asséssible dans toute l'instance de l'objet créé et ne nécessite pas de variables globales
- Pour les widgets de Tkinter

  - Pour créer un objet widget, on utilise la syntaxe : nom_objet = nom_classe(attributs)
  - Pour positionner cet objet widget dans le fenêtre Tk, on utilise la méthode grid() :
  - nom_objet.grid(arguments)
  - Pour modifier un objet widget déjà créé, on utilise la méthode configure() :
  - nom_objet.configure(attributs à modifier)
  - Pour supprimer un objet widget, on utilise la méthode destroy() : nom_objet.destroy()

DOCUMENT A RENDRE :

Cette partie vous a permis de découvrir la bibliothèque Tkinter et de créer une fenêtre comprenant plusieurs widgets :

- 1 canvas (objet de la classe Canvas) qui pourra ultérieurement accueillir des éléments graphiques
- 1 widget de type texte (objet de la classe Label)
- 1 widget champ de saisie (objet de la classe Text)
- 1 widget bouton (objet de la classe Button) dont l'action est liée à un évènement.

Transférer le fichier t5.py **par l'intermédiaire de l'onglet transfert** du site https://nsibranly.fr en utilisant le code : **tp5** .