PROJET 3:

Générateur de Contenus Narratifs

Elhadji Omar DIENG

Baye Lahad DIA

Mbaye Babacar LO

Emmanuel Fouda Ayabone DIEDHIOU

Plan

- Introduction
- Création des classes
- Mise en place d'une base de données
- Programme test et Application graphique (Tkinter)
- Conclusion

Introduction

Dans le cadre de notre projet de Programmation Orientée Objet (POO), nous avons développé un **Générateur de Contenus Narratifs**.
L'objectif principal était de concevoir une application capable de créer automatiquement des histoires dynamiques à partir de modèles prédéfinis de **personnages**, **lieux** et **événements**.

Pour cela, nous avons appliqué les principes fondamentaux de la POO, notamment la création et l'utilisation de classes, l'encapsulation des données, et la **composition** pour combiner différents éléments narratifs.

Afin de rendre le projet interactif et convivial, nous avons intégré une **interface graphique** réalisée avec **Tkinter**, permettant à l'utilisateur de générer et de visualiser ses histoires d'un simple clic.

1 . Création des classes

Dans le fichier Classe3.py, nous avons créé les classes personnages, lieu et événement et composition qui nous permettront plus tard de générer des histoires dynamiques.

a) Classes principales

1. **Classe `Personnage`**

    - **Attributs** : prénom, nom, âge, profession, traits de caractère.
    - **Méthodes** : constructeur, affichage du personnage.

```python
class Personnage:
    def __init__(self, prenom, nom, age, profession, caractere):
        self.prenom = prenom
        self.nom = nom
        self.age = age
        self.profession = profession
        self.caractere = caractere
    def __str__(self):
        return f"{self.prenom} {self.nom} un(e) {self.profession} de {self.age} ans , {self.caractere}."
```

2. **Classe `Lieu`**

    - **Attributs** : nom du lieu, description, ambiance.

    - **Méthodes** : constructeur, affichage du lieu

```python
class Lieu:
    def __init__(self, nomL, typeL, description):
        self.lieu = nomL
        self.typeL = typeL
        self.desc = description
    def __str__(self):
        return f"{self.lieu} un(e) {self.typeL}  {self.desc}"
```

3. **Classe `Événement`**

    - **Attributs** : description(action ou péripétie de l'événement), conséquences
    - **Méthodes** : constructeur, affichage de l'événement

```python
class Evenement:
    def __init__(self, action, consequence):
        self.action = action
        self.consequence = consequence
    def __str__(self):
        return f"{self.action} et {self.consequence}"
```

b) Composition

- **Attributs** : personnage, lieu, événement.
- **Méthodes** : constructeur, affichage de l'histoire
- Une histoire est composée d'**un personnage**, **un lieu**, et **un événement**.

- Comme son nom l'indique, la classe **composition** combine ces trois éléments dans une classe `composition.`Elle permet ainsi de générer des histoires dynamiques à partir des modèles de personnages, lieux et événements.

- Au lieu d'utiliser des héritages nous afficherons directement les objets déjà instanciés

```python
class Composition:
    def __init__(self, personnage, lieu, evenement ):
        self.personnage = personnage
        self.lieu = lieu
        self.evenement = evenement
    def __str__(self):
        return f"Un jour, {self.personnage} se rendit a {self.lieu} et {self.evenement}."
```

NB:

Dans toutes nos classes établis ci-dessus nous avons utilisé la méthode spéciale

<< **__str__( )** >> qui nous permettra dans la suite de pouvoir afficher l'objet de la classe dans le format défini dans l'instruction return.

2 . Mise en place d'une base de données

Dans un fichier data.py nous avons instancié plusieurs modèles de personnages, de lieux et d'événements qui serviront de base de données afin de pouvoir générer nos histoires dynamiquement.
Pour cela nous avons d'abord importé les classes Personnages, Lieu et Evenement de Classe3.py.

```python
from Classe3 import Personnage, Lieu, Evenement
```

a. Modèles de personnages

Nous avons créé et instancié une liste de plusieurs objets de personnages

```python
personnages = [
    Personnage("Moussa", "DIONE", 34, "Enseignant", "rêveur passionné d'énigmes anciennes"),
    Personnage("Omar", "Dieng", 21, "Explorateur", "curieux et toujours prêt à prendre des risques"),
    Personnage("Babacar", "LY", 20, "Scientifique", "brillant mais parfois distrait par ses propres théories"),
    Personnage("Aïssatou", "BA", 27, "Archéologue", "persévérante et fascinée par les civilisations perdues"),
    Personnage("Fatou", "SARR", 30, "Guerrière", "courageuse, protectrice et dotée d'un sens de l'honneur inébranlable")
]
```

b. Modèles de lieux

Nous avons créé et instancié une liste de plusieurs objets lieux

```
lieux = [
    Lieu("La cité souterraine", "cité", "cachée sous terre, éclairée par des cristaux luminescents et gardée par des stat
    Lieu("La Forêt Enchantée", "forêt", "pleine de mystères, où les arbres murmurent aux voyageurs et les rivières change
    Lieu("La vieille forteresse", "forteresse", "abandonnée depuis des siècles, hantée par les échos dune bataille oublié
    Lieu("L'Île des Brumes", "île", "entourée d'un brouillard éternel où l'on entend parfois des voix"),
    Lieu("La Bibliothèque des Étoiles", "bibliothèque", "perdue dans un désert et contenant des manuscrits écrits dans un
]
```

c. Modèles d'événements

Nous avons créé et instancié une liste de plusieurs objets événement

```
evenements = [
    Evenement("a affronté ses peurs en traversant un labyrinthe d'ombres mouvantes",
              "a trouvé sa véritable voie en aidant un peuple oublié à retrouver la lumière"),
    Evenement("a découvert un passage secret derrière une cascade chantante",
              "a libéré un esprit ancien qui lui a confié un secret capable de changer l'équilibre du monde"),
    Evenement("a trouvé un artefact magique dans une salle interdite",
              "a modifié à jamais le destin du royaume en éveillant les gardiens légendaires"),
    Evenement("a combattu un dragon endormi depuis mille ans",
              "a gagné la loyauté de la créature en lui rendant son trésor perdu"),
    Evenement("a résolu l'énigme d'une porte millénaire",
              "a ouvert un chemin vers un monde parallèle")
]
```

3. Programme Test et Application graphique (Tkinter)

Dans un fichier Main.py nous avons créé un programme principal qui utilise les modèles de personnages, de lieux et d'événements définis dans data.py pour générer les histoires. Le résultat sur application graphique graphique grâce à l'intégration de Tkinter.

a. Importation des modules

- La classe Composition pour combiner les modèles de personnages, de lieux et d'événements définis dans data.py (qui sont aussi importés)
- le module tkinter
- le module random pour la sélection aléatoire des modèles

```
from Classe3 import Composition
from data import personnages , lieux , evenements
import tkinter as tk
import random
```

b. Intégration de Tkinter

Après avoir importer Tkinter plus haut nous allons l'utiliser pour afficher les histoires qui seront générées dynamiquement pour un rendu et un affichage plus convivial que sur le terminal windows.

- La fonction generer_histoire( )

  L'utilisation de random.choice permet de sélectionner aléatoirement les personnages, lieux et événements importés.

  La Composition combiner cela et l'insert sous forme de texte.

```
def generer_histoire():
    p = random.choice(personnages)
    l = random.choice(lieux)
    e = random.choice(evenements)
    histoire = Composition(p, l, e)
    texte_resultat.config(state='normal')
    texte_resultat.delete("0.0", tk.END)
    texte_resultat.insert(tk.END, histoire)
    texte_resultat.config(state='disabled')
```

- Création de la fenetre (dimension, couleur,) et ajout d'un cadre contenant tous les éléments de la fenêtre

```
fenetre = tk.Tk()
fenetre.title("Générateur d'Histoires")
fenetre.geometry("800x500")
fenetre.config(bg="#B9E0FF")

# Cadre principal
cadre = tk.Frame(fenetre, bg="#B9E0FF")
cadre.pack(pady=20)
```

- Un titre (en gros caractère) qui met en avant l'utilité de l'application

```
titre = tk.Label(cadre, text="Générateur d'Histoires Dynamiques", font=("Impact", 20, "bold"), bg="#f0f0f0")
titre.pack(pady=10)
```

- Un bouton Generer qui prend comme paramètre de commande la fonction generer_histoire avec toutes ces autres propriétés.

  Chaque fois qu'on appuie dessus la fonction s'exécute et génère de nouveaux histoires.

```
bouton = tk.Button(cadre, text="🎲 Générer une Histoire", font=("Arial", 14), command=generer_histoire, bg="#4CAF50", fg
bouton.pack(pady=10)
```

- Une zone de texte où sera affichée l'histoire générée. A chaque nouvelle génération d'histoire le texte actuel est supprimé et remplacé par le nouveau text.

```
# Zone de texte
texte_resultat = tk.Text(cadre, height=16, width=100, wrap='word', font=("Arial", 12))
texte_resultat.config(state='disabled', bg="#ffffff")
texte_resultat.pack(pady=10)
```

- Un bouton quitter qui permet de fermer la fenêtre graphique. Elle met fin à la boucle infinie d'exécution de la fenêtre.

```
bouton_quitter = tk.Button(fenetre, text="Quitter", command=fenetre.destroy, bg="#d9534f", fg="white", width=10)
bouton_quitter.pack()

# Boucle principale
fenetre.mainloop()
```

c. Exécution du programme

A présent nous allons executer Main.py afin de générer dynamiquement quelques histoires.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python

PS C:\Users\HP\Desktop\Semestre3\Ḥ_POO\Projet2025> & C:/Users/HP/AppData/Local/Programs/Python/Python313/python.exe c:/Users/HP/Desktop/Semestre3/Ḥ_POO/Projet2025/
Main.py
```

Affichage de la fenêtre graphique sans histoire pour l'instant.



➢ 1 ère histoire générée

On appuie sur le bouton 🎲Générer une Histoire



➢ 2 ème histoire générée

On appuie encore pour générer une nouvelle histoire sans quitter la fenêtre.



Conclusion

Ce projet a permis de mettre en pratique nos compétences en **Programmation Orientée Objet** et en **manipulation de données aléatoires**. Elle nous a aussi permis de parcourir et de découvrir d'autres aspects de la POO et de développer notre créativité à travers l'apprentissage et l'utilisation **de l'interface graphique Tkinter**. Cette interface utilisateur va au-delà d'un simple affichage sur un terminal et permet de personnaliser l'affichage pour un rendu plus que convivial.