МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж



Лабораторна робота №4

з дисципліни «Спеціалізовані мови програмування»

на тему «Розробка ASCII ART генератора для візуалізації 2D-фігур»

Виконала студентка

групи РІ-32

Титиш О.І.

Прийняв:

Щербак С.С

Львів – 2024

**Мета:** Створення Генератора ASCII-арту без використання зовнішніх бібліотек

**План роботи**

Завдання 1: Введення користувача

Створіть програму Python, яка отримує введення користувача щодо слова або фрази, яку вони хочуть перетворити в ASCII-арт.

Завдання 2: Набір символів

Визначте набір символів (наприклад, '@', '#', '\*', тощо), які будуть використовуватися для створення ASCII-арту. Ці символи будуть відображати різні відтінки.

Завдання 3: Розміри Art-у

Запитайте у користувача розміри (ширина і висота) ASCII-арту, який вони хочуть створити. Переконайтеся, що розміри в межах керованого діапазону

Завдання 4: Функція генерації Art-у

Напишіть функцію, яка генерує ASCII-арт на основі введення користувача, набору символів та розмірів. Використовуйте введення користувача, щоб визначити, які символи використовувати для кожної позиції в Art-у.

Завдання 5: Вирівнювання тексту

Реалізуйте опції вирівнювання тексту (ліво, центр, право), щоб користувачі могли вибирати, як їх ASCII-арт розміщується на екрані.

Завдання 6: Відображення мистецтва

Відобразіть створений ASCII-арт на екрані за допомогою стандартних функцій друку Python.

Завдання 7: Збереження у файл

Додайте можливість зберігати створений ASCII-арт у текстовий файл, щоб користувачі могли легко завантажувати та обмінюватися своїми творіннями.

Завдання 8: Варіанти кольорів

Дозвольте користувачам вибирати опції кольорів (чорно-білий, відтінки сірого) для свого ASCII-арту.

Завдання 9: Функція попереднього перегляду

Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їх ASCII-арту перед остаточним збереженням

Завдання 10: Інтерфейс, зрозумілий для користувача

Створіть інтерфейс для користувача у командному рядку, щоб зробити програму легкою та інтуїтивно зрозумілою для використання.

**Код програми**

ascii\_art\_generator.py

from Shared.utils2\_LAB4 import base\_font, align\_text  
  
  
class ArtGenerator:  
 LETTERS = base\_font()  
  
*#Функція генерації ASCII-арту* @staticmethod  
 def generate\_art(text, width, height, custom\_symbol, color\_option):  
 text = text.upper()  
 letter\_width = min(5, (width - len(text) + 1) // len(text))  
 letter\_height = min(5, height)  
 art = [[' ' for \_ in range(width)] for \_ in range(height)]  
  
 x\_offset = 0  
 for char in text:  
 letter = ArtGenerator.LETTERS.get(char, ArtGenerator.LETTERS[' '])  
 for y in range(letter\_height):  
 for x in range(letter\_width):  
 if x + x\_offset < width and y < height:  
 if letter[y][x] != ' ':  
 if color\_option == 'black\_white':  
 art[y][x + x\_offset] = custom\_symbol  
 else: *# grayscale* intensity = (x + y) % 4  
 art[y][x + x\_offset] = ['░', '▒', '▓', '█'][intensity]  
 x\_offset += letter\_width + 1  
 if x\_offset >= width:  
 break  
  
 return [''.join(row) for row in art]  
  
 @staticmethod  
 def \_scale\_letter(letter, width, height, custom\_symbol, color\_option):  
 scaled = [[' ' for \_ in range(width)] for \_ in range(height)]  
 orig\_height, orig\_width = len(letter), len(letter[0])  
  
 for y in range(height):  
 for x in range(width):  
 orig\_y = int(y \* orig\_height / height)  
 orig\_x = int(x \* orig\_width / width)  
 if letter[orig\_y][orig\_x] != ' ':  
 if color\_option == 'black\_white':  
 scaled[y][x] = custom\_symbol  
 else: *# grayscale* intensity = int((y \* x \* 25) / (height \* width))  
 scaled[y][x] = chr(9617 + min(intensity // 8, 3)) *# Unicode block elements* return scaled

file\_operations.py

class FileManager:  
 *#Збереження у файл* @staticmethod  
 def save\_to\_file(art, filename):  
 try:  
 with open(filename, 'w', encoding='utf-8') as file:  
 for row in art:  
 file.write(row + '\n')  
 return True  
 except IOError:  
 return False

utils2.py

*#Вирівнювання тексту*def align\_text(art, alignment, width):  
 if alignment == 'left':  
 return [line.ljust(width) for line in art]  
 elif alignment == 'center':  
 return [line.center(width) for line in art]  
 elif alignment == 'right':  
 return [line.rjust(width) for line in art]  
 return art  
  
def base\_font():  
 return {  
 'A': [  
 ' @ ',  
 ' @ @ ',  
 '@ @',  
 '@@@@@',  
 '@ @'  
 ],  
 'B': [  
 '@@@@ ',  
 '@ @',  
 '@@@@ ',  
 '@ @',  
 '@@@@ '  
 ],  
 'C': [  
 ' @@@ ',  
 '@ ',  
 '@ ',  
 '@ ',  
 ' @@@ '  
 ],  
 'D': [  
 '@@@@ ',  
 '@ @',  
 '@ @',  
 '@ @',  
 '@@@@ '  
 ],  
 'E': [  
 '@@@@@',  
 '@ ',  
 '@@@ ',  
 '@ ',  
 '@@@@@'  
 ]

…

}

cli.py

class CLI:  
  
 *#Введення користувача* @staticmethod  
 def get\_user\_input():  
 return input("Введіть слово або фразу для перетворення в ASCII-арт: ")  
  
 *#Розміри ASCII-арту* @staticmethod  
 def get\_art\_dimensions():  
 while True:  
 try:  
 width = int(input("Введіть ширину ASCII-арту (від 20 до 100): "))  
 height = int(input("Введіть висоту ASCII-арту (від 5 до 20): "))  
 if 20 <= width <= 100 and 5 <= height <= 20:  
 return width, height  
 print("Розміри повинні бути в межах вказаного діапазону.")  
 except ValueError:  
 print("Будь ласка, введіть цілі числа.")  
  
 @staticmethod  
 def get\_alignment():  
 while True:  
 alignment = input("Виберіть вирівнювання (left/center/right): ").lower()  
 if alignment in ['left', 'center', 'right']:  
 return alignment  
 print("Будь ласка, виберіть коректне вирівнювання.")  
  
 *#Набір символів* @staticmethod  
 def get\_custom\_symbol():  
 return input("Введіть символ для створення ASCII-арту (наприклад, '@', '#', '\*'): ")  
  
 *#Варіанти кольорів* @staticmethod  
 def get\_color\_option():  
 while True:  
 color = input("Виберіть колірну схему (black\_white/grayscale): ").lower()  
 if color in ['black\_white', 'grayscale']:  
 return color  
 print("Будь ласка, виберіть коректну колірну схему.")  
  
 *#Відображення мистецтва* @staticmethod  
 def display\_art(art):  
 for row in art:  
 print(row)  
  
 @staticmethod  
 def get\_save\_choice():  
 return input("Зберегти цей ASCII-арт? (y/n): ").lower() == 'y'  
  
 @staticmethod  
 def get\_filename():  
 return input("Введіть ім'я файлу для збереження: ")

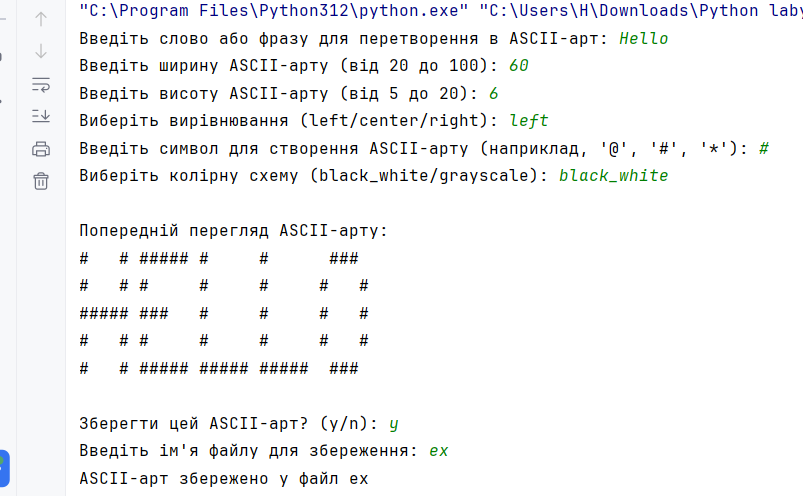
runner.py

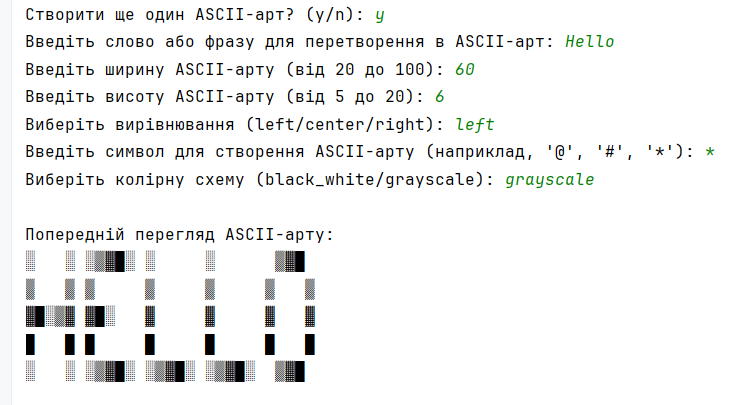
from UI.cli\_LAB4 import CLI  
from BLL.ascii\_art\_generator\_LAB4 import ArtGenerator  
from DAL.file\_operations\_LAB4 import FileManager  
from Shared.utils2\_LAB4 import align\_text  
from colorama import init

def main():  
 cli = CLI()  
  
 while True:  
 text = cli.get\_user\_input()  
 width, height = cli.get\_art\_dimensions()  
 alignment = cli.get\_alignment()  
 custom\_symbol = cli.get\_custom\_symbol()  
 color\_option = cli.get\_color\_option()  
  
 art\_generator = ArtGenerator()  
 ascii\_art = art\_generator.generate\_art(text, width, height, custom\_symbol, color\_option)  
  
 aligned\_art = align\_text(ascii\_art, alignment, width)  
  
 print("\nПопередній перегляд ASCII-арту:")  
 cli.display\_art(aligned\_art)  
  
 if cli.get\_save\_choice():  
 filename = cli.get\_filename()  
 if FileManager.save\_to\_file(aligned\_art, filename):  
 print(f"ASCII-арт збережено у файл {filename}")  
 else:  
 print("Помилка при збереженні файлу.")  
  
 if input("Створити ще один ASCII-арт? (y/n): ").lower() != 'y':  
 break  
  
 print("Дякуємо за використання ASCII Art генератора!")

if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Результат виконання програми**

****

****

*Рисунок 1, 2. Результати виконання*

Посилання на GitHub:

**Висновок:** При виконанні даної лабораторної роботи, я створила генератор ASCII-арту з нуля, та надала можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволило мені глибше розібратися як створюється ASCII-арт.