ЗВІТ
про виконання лабораторної роботи № 4
«Розробка ASCII ART генератора для візуалізації 2D-фігур »
з дисципліни
«Спеціалізовані мови програмування»
ст. групи PI-31
Танечник Марічки

Мета: Створення Генератора ASCII-арту без використання зовнішніх бібліотек
Умова завдання:

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 твореннями.

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.

Текст програми:

```
import os
import math
import random
import GlobalVariables as Global
from pyfiglet import figlet_format, FigletFont


class Ascii:
    @staticmethod
    def verify_width(width):
        if width <= 0:
            try:
                return os.get_terminal_size().columns
            except OSError:
                return 220
        elif width > 0:
            return width
        else:
            return 220

    @staticmethod
    def print(text, is_random=False):
        if is_random:
            font = random.choice(FigletFont.getFonts())
            color = "\033[" + str(random.randint(31, 39)) + "m"
        else:
            font = Global.font
            color = Global.color

        art = figlet_format(text, font=font, width=Global.width)

        print(color + art + Global.color_reset)
import os

from assets.font import chars

def getInput():
```

```
    user_input = input('Enter the phrase: ')

    return user_input

def changeSymbol(art, symbol):
    updated_art = ""
    for char in art:
        if char != '\n' and char != ' ':
            updated_art += symbol
        else:
            updated_art += char
    return updated_art


def askToSaveArt(folderToSave, art):
    isArtToSave = input('Do you want to save your art? (y/n): ')

    if isArtToSave == 'y':
        saveArt(folderToSave, art)
    else:
        pass

def saveArt(folderToSave, art):
    file_name = input('Give a file name: ')
    formated_file_name = folderToSave + file_name + '.txt'

    with open(formated_file_name, 'w') as file:
        file.write(art)

def previewArt(art):
    print(art)


def scaleArt(ascii_art, width_factor, height_factor):
    width_factor = int(width_factor)
    height_factor = int(height_factor)

    scaled_lines = []
    for line in ascii_art.splitlines():
```

```
        scaled_line = "".join(char * width_factor for char in line)
        for _ in range(height_factor):
            scaled_lines.append(scaled_line)
    return "\n".join(scaled_lines)

def draw_char(text, symbol="*"):
    result = [""] * 6  # Assuming each character ASCII art is 6 lines high

    for char in text.upper():
        if char in chars:  # Check if character exists in our ASCII template dictionary
            for i in range(6):  # Iterate over each line of the ASCII character
                line = chars[char][i].replace("*", symbol)  # Replace default symbol with the specified one
                result[i] += line + "  "  # Add spacing between characters
        else:
            for i in range(6):  # If character is not found, add spaces
                result[i] += " " * (len(chars.get('A', [''])[0]) + 2)

    return "\n".join(result)



def align_art(ascii_art, width, alignment):
    aligned_lines = []
    for line in ascii_art.splitlines():
        if alignment == "left":
            aligned_lines.append(line.ljust(width))
        elif alignment == "center":
            aligned_lines.append(line.center(width))
        elif alignment == "right":
            aligned_lines.append(line.rjust(width))
        else:
            aligned_lines.append(line)
    return "\n".join(aligned_lines)
```

Висновки: виконуючи ці завдання, я створила генератор ASCII-арту з нуля, та надала можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволить користувачам глибше розібратися як створюється ASCII-арт