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

Виконав:

ст. гр. ІТ-21сп

Олександр КОЗАК

Прийняв

доцент каф. ІСМ:

Сергій ЩЕРБАК

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Львів-2023

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

**Хід 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.

**main.py**

# Import the Ascii\_art\_runner class from the ascci\_art\_runner module

import ascci\_art\_runner

# Main entry point of the program

class Main:

if \_\_name\_\_ == "\_\_main\_\_":

# Create an instance of the Ascii\_art\_runner class

runner = ascci\_art\_runner.Ascii\_art\_runner()

# Run the ASCII art generator

runner.run\_ascii\_art\_gen()

**ascci\_art\_runner.py**

# Import the Ascii\_art\_generator class from the ascii\_art\_engine module

from ascii\_art\_engine import Ascii\_art\_generator

# Class for running the ASCII art generator

class Ascii\_art\_runner:

def run\_ascii\_art\_gen(self):

# Create an instance of the Ascii\_art\_generator class

ascii\_art\_gen = Ascii\_art\_generator()

# Get user input for text, color, preview choice, and save choice

user\_text, color, preview\_choose, save\_choose = ascii\_art\_gen.user\_input()

# Generate ASCII art based on user input

ascii\_art = ascii\_art\_gen.creating\_ascii\_art(user\_text, color)

# Display the ASCII art preview based on user's preview choice

ascii\_art\_gen.ascii\_art\_preview(ascii\_art, preview\_choose)

# If user chose to save the ASCII art, save it to a text file

if save\_choose == "y":

ascii\_art\_gen.save\_ascii\_art\_to\_txt(ascii\_art, "ascii\_art")

**ascii\_art\_engine.py**

# Import necessary modules

import fonts

from coloration import Coloration

# Class for generating ASCII art

class Ascii\_art\_generator:

def \_\_init\_\_(self):

# Define available colors

self.available\_colors = ['black', 'grey', 'red', 'green', 'blue', 'magenta', 'cyan', 'yellow']

# Get user input for ASCII art generation

def user\_input(self):

user\_text = input("Enter a word or phrase to convert to ASCII-art: ")

# Display available colors and get user's color choice

self.show\_colors()

user\_color = self.get\_color("Select an available color number: ")

# Get user choices for preview and saving

user\_preview\_choose = self.get\_user\_choose("Want to preview ASCII-art before saving it?(y/n): ").lower()

user\_save\_chose = self.get\_user\_choose(

"Do you want to save the created ASCII-art to a text file?(y/n): ").lower()

return user\_text, user\_color, user\_preview\_choose, user\_save\_chose

# Generate ASCII art based on the selected font

def generate\_ascii\_art(self, text):

selected\_font = fonts.banner

ascii\_art = ""

for line in range(len(selected\_font['a'].split('\n'))):

for char in text:

if char.isalpha():

if char in selected\_font:

ascii\_art += selected\_font[char].split('\n')[line]

else:

ascii\_art += " " \* 5 # Default space if character is not available

else:

ascii\_art += "\n" # Move to the next line for non-letter characters

ascii\_art += "\n" # Move to the next line after completing a line of characters

return ascii\_art

# Create colored ASCII art

def creating\_ascii\_art(self, user\_text, user\_color):

ascii\_art = self.generate\_ascii\_art(user\_text)

colored\_ascii\_art = Coloration.colored(ascii\_art, user\_color)

return colored\_ascii\_art

# Display available colors

def show\_colors(self):

print("Available colors:")

for index, color in enumerate(self.available\_colors, start=1):

print(f"{index}. {color}")

# Get user choice (yes or no) with input validation

def get\_user\_choose(self, message):

while True:

try:

choice = input(message)

if choice.lower() in ['y', 'n']:

return choice

else:

print("Select the right option!")

except ValueError:

print("Input must be a string data type!")

# Get user's color choice with input validation

def get\_color(self, message):

while True:

try:

choice = int(input(message))

if 1 <= choice <= len(self.available\_colors):

return self.available\_colors[choice - 1]

else:

print("Select the correct color number!")

except ValueError:

print("Input must be a numeric data type!")

# Display ASCII art

def show\_ascii\_art(self, ascii\_art):

print(ascii\_art)

# Save ASCII art to a text file

def save\_ascii\_art\_to\_txt(self, final\_ascii\_art, file\_name):

with open(f"{file\_name}.txt", "w") as file:

file.write(final\_ascii\_art)

print(f"Your ASCII-art successfully saved to a file named {file\_name}.txt! :D")

# Display ASCII art preview based on user's choice

def ascii\_art\_preview(self, ascii\_art, preview\_chose):

if preview\_chose == 'y':

self.show\_ascii\_art(ascii\_art)

**coloration.py**

class Coloration():

COLORS = {

"black": 30,

"grey": 30, # Actually black but kept for backwards compatibility

"red": 31,

"green": 32,

"yellow": 33,

"blue": 34,

"magenta": 35,

"cyan": 36,

"light\_grey": 37,

"dark\_grey": 90,

"light\_red": 91,

"light\_green": 92,

"light\_yellow": 93,

"light\_blue": 94,

"light\_magenta": 95,

"light\_cyan": 96,

"white": 97,

}

RESET = "\033[0m"

def colored(

text: str,

color: str | None = None,

) -> str:

"""Colorize text.

Available text colors:

black, red, green, yellow, blue, magenta, cyan, white,

light\_grey, dark\_grey, light\_red, light\_green, light\_yellow, light\_blue,

light\_magenta, light\_cyan.

Example:

colored('Hello, World!', 'red')

"""

fmt\_str = "\033[%dm%s"

if color is not None:

text = fmt\_str % (Coloration.COLORS[color], text)

return text + Coloration.RESET

**fonts.py**

banner = {

    'A': u'   #    \n  # #   \n #   #  \n#     # \n####### \n#     # \n#     # \n        \n',

    'B': u'######  \n#     # \n#     # \n######  \n#     # \n#     # \n######  \n        \n',

    'C': u' #####  \n#     # \n#       \n#       \n#       \n#     # \n #####  \n        \n',

    'D': u'######  \n#     # \n#     # \n#     # \n#     # \n#     # \n######  \n        \n',

    'E': u'####### \n#       \n#       \n#####   \n#       \n#       \n####### \n        \n',

    'F': u'####### \n#       \n#       \n#####   \n#       \n#       \n#       \n        \n',

    'G': u' #####  \n#     # \n#       \n#  #### \n#     # \n#     # \n #####  \n        \n',

    'H': u'#     # \n#     # \n#     # \n####### \n#     # \n#     # \n#     # \n        \n',

    'I': u'### \n #  \n #  \n #  \n #  \n #  \n### \n    \n',

    'J': u'      # \n      # \n      # \n      # \n#     # \n#     # \n #####  \n        \n',

    'K': u'#    # \n#   #  \n#  #   \n###    \n#  #   \n#   #  \n#    # \n       \n',

    'L': u'#       \n#       \n#       \n#       \n#       \n#       \n####### \n        \n',

    'M': u'#     # \n##   ## \n# # # # \n#  #  # \n#     # \n#     # \n#     # \n        \n',

    'N': u'#     # \n##    # \n# #   # \n#  #  # \n#   # # \n#    ## \n#     # \n        \n',

    'O': u'####### \n#     # \n#     # \n#     # \n#     # \n#     # \n####### \n        \n',

    'P': u'######  \n#     # \n#     # \n######  \n#       \n#       \n#       \n        \n',

    'Q': u' #####  \n#     # \n#     # \n#     # \n#   # # \n#    #  \n #### # \n        \n',

    'R': u'######  \n#     # \n#     # \n######  \n#   #   \n#    #  \n#     # \n        \n',

    'S': u' #####  \n#     # \n#       \n #####  \n      # \n#     # \n #####  \n        \n',

    'T': u'####### \n   #    \n   #    \n   #    \n   #    \n   #    \n   #    \n        \n',

    'U': u'#     # \n#     # \n#     # \n#     # \n#     # \n#     # \n #####  \n        \n',

    'V': u'#     # \n#     # \n#     # \n#     # \n #   #  \n  # #   \n   #    \n        \n',

    'W': u'#     # \n#  #  # \n#  #  # \n#  #  # \n#  #  # \n#  #  # \n ## ##  \n        \n',

    'X': u'#     # \n #   #  \n  # #   \n   #    \n  # #   \n #   #  \n#     # \n        \n',

    'Y': u'#     # \n #   #  \n  # #   \n   #    \n   #    \n   #    \n   #    \n        \n',

    'Z': u'####### \n     #  \n    #   \n   #    \n  #     \n #      \n####### \n        \n',

    ' ': u'   \n   \n   \n   \n   \n   \n   \n   \n',

    '!': u'### \n### \n### \n #  \n    \n### \n### \n    \n',

    '`': u'### \n### \n #  \n  # \n    \n    \n    \n    \n',

    "'": u'### \n### \n #  \n#   \n    \n    \n    \n    \n',

    '"': u'### ### \n### ### \n #   #  \n        \n        \n        \n        \n        \n',

    ';': u'    \n### \n### \n    \n### \n### \n #  \n#   \n',

    '#': u'  # #   \n  # #   \n####### \n  # #   \n####### \n  # #   \n  # #   \n        \n',

    '@': u' #####  \n#     # \n# ### # \n# ### # \n# ####  \n#       \n #####  \n        \n',

    '$': u' #####  \n#  #  # \n#  #    \n #####  \n   #  # \n#  #  # \n #####  \n        \n',

    '%': u'###   # \n# #  #  \n### #   \n   #    \n  # ### \n #  # # \n#   ### \n        \n',

    '&': u'  ##    \n #  #   \n  ##    \n ###    \n#   # # \n#    #  \n ###  # \n        \n',

    '\\': u'#       \n #      \n  #     \n   #    \n    #   \n     #  \n      # \n        \n',

    '/': u'      # \n     #  \n    #   \n   #    \n  #     \n #      \n#       \n        \n',

    '|': u'# \n# \n# \n  \n# \n# \n# \n  \n',

    '(': u'  ## \n #   \n#    \n#    \n#    \n #   \n  ## \n     \n',

    ')': u'##   \n  #  \n   # \n   # \n   # \n  #  \n##   \n     \n',

    '<': u'   # \n  #  \n #   \n#    \n #   \n  #  \n   # \n     \n',

    '>': u'#    \n #   \n  #  \n   # \n  #  \n #   \n#    \n     \n',

    '[': u'##### \n#     \n#     \n#     \n#     \n#     \n##### \n      \n',

    ']': u'##### \n    # \n    # \n    # \n    # \n    # \n##### \n      \n',

    '{': u'  ### \n #    \n #    \n##    \n #    \n #    \n  ### \n      \n',

    '}': u'###   \n   #  \n   #  \n   ## \n   #  \n   #  \n###   \n      \n',

    '+': u'      \n  #   \n  #   \n##### \n  #   \n  #   \n      \n      \n',

    '\*': u'        \n #   #  \n  # #   \n####### \n  # #   \n #   #  \n        \n        \n',

    '^': u'  #   \n # #  \n#   # \n      \n      \n      \n      \n      \n',

    '-': u'      \n      \n      \n##### \n      \n      \n      \n      \n',

    '=': u'      \n      \n##### \n      \n##### \n      \n      \n      \n',

    '\_': u'        \n        \n        \n        \n        \n        \n        \n####### \n',

    '~': u' ##     \n#  #  # \n    ##  \n        \n        \n        \n        \n        \n',

    ',': u'    \n    \n    \n    \n### \n### \n #  \n#   \n',

    '.': u'    \n    \n    \n    \n### \n### \n### \n    \n',

    ':': u' #  \n### \n #  \n    \n #  \n### \n #  \n    \n',

    '?': u' #####  \n#     # \n      # \n   ###  \n   #    \n        \n   #    \n        \n',

    'a': u'       \n  ##   \n #  #  \n#    # \n###### \n#    # \n#    # \n       \n',

    'b': u'       \n#####  \n#    # \n#####  \n#    # \n#    # \n#####  \n       \n',

    'c': u'       \n ####  \n#    # \n#      \n#      \n#    # \n ####  \n       \n',

    'd': u'       \n#####  \n#    # \n#    # \n#    # \n#    # \n#####  \n       \n',

    'e': u'       \n###### \n#      \n#####  \n#      \n#      \n###### \n       \n',

    'f': u'       \n###### \n#      \n#####  \n#      \n#      \n#      \n       \n',

    'g': u'       \n ####  \n#    # \n#      \n#  ### \n#    # \n ####  \n       \n',

    'h': u'       \n#    # \n#    # \n###### \n#    # \n#    # \n#    # \n       \n',

    'i': u'  \n# \n# \n# \n# \n# \n# \n  \n',

    'j': u'       \n     # \n     # \n     # \n     # \n#    # \n ####  \n       \n',

    'k': u'       \n#    # \n#   #  \n####   \n#  #   \n#   #  \n#    # \n       \n',

    'l': u'       \n#      \n#      \n#      \n#      \n#      \n###### \n       \n',

    'm': u'       \n#    # \n##  ## \n# ## # \n#    # \n#    # \n#    # \n       \n',

    'n': u'       \n#    # \n##   # \n# #  # \n#  # # \n#   ## \n#    # \n       \n',

    'o': u'       \n ####  \n#    # \n#    # \n#    # \n#    # \n ####  \n       \n',

    'p': u'       \n#####  \n#    # \n#    # \n#####  \n#      \n#      \n       \n',

    'q': u'       \n ####  \n#    # \n#    # \n#  # # \n#   #  \n ### # \n       \n',

    'r': u'       \n#####  \n#    # \n#    # \n#####  \n#   #  \n#    # \n       \n',

    's': u'       \n ####  \n#      \n ####  \n     # \n#    # \n ####  \n       \n',

    't': u'      \n##### \n  #   \n  #   \n  #   \n  #   \n  #   \n      \n',

    'u': u'       \n#    # \n#    # \n#    # \n#    # \n#    # \n ####  \n       \n',

    'v': u'       \n#    # \n#    # \n#    # \n#    # \n #  #  \n  ##   \n       \n',

    'w': u'       \n#    # \n#    # \n#    # \n# ## # \n##  ## \n#    # \n       \n',

    'x': u'       \n#    # \n #  #  \n  ##   \n  ##   \n #  #  \n#    # \n       \n',

    'y': u'      \n#   # \n # #  \n  #   \n  #   \n  #   \n  #   \n      \n',

    'z': u'       \n###### \n    #  \n   #   \n  #    \n #     \n###### \n       \n',

    '0': u'  ###   \n #   #  \n#     # \n#     # \n#     # \n #   #  \n  ###   \n        \n',

    '1': u'  #   \n ##   \n# #   \n  #   \n  #   \n  #   \n##### \n      \n',

    '2': u' #####  \n#     # \n      # \n #####  \n#       \n#       \n####### \n        \n',

    '3': u' #####  \n#     # \n      # \n #####  \n      # \n#     # \n #####  \n        \n',

    '4': u'#       \n#    #  \n#    #  \n#    #  \n####### \n     #  \n     #  \n        \n',

    '5': u'####### \n#       \n#       \n######  \n      # \n#     # \n #####  \n        \n',

    '6': u' #####  \n#     # \n#       \n######  \n#     # \n#     # \n #####  \n        \n',

    '7': u'####### \n#    #  \n    #   \n   #    \n  #     \n  #     \n  #     \n        \n',

    '8': u' #####  \n#     # \n#     # \n #####  \n#     # \n#     # \n #####  \n        \n',

    '9': u' #####  \n#     # \n#     # \n ###### \n      # \n#     # \n #####  \n        \n',

}

def get\_font(letter):

    return banner.get(letter.upper())

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

*Рис. 1 Результат виконання завдання*

**Висновок:** Виконуючи ці завдання, я створив створити генератор ASCII-арту з нуля, та надала можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволить їм глибше розібратися як створюється ASCII-арт.