МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №4

з курсу

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Виконала студентка

групи ІТ-21сп

**Романишин.Д.О.**

Прийняв

**Щербак С.С.**

Львів - 2023

**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 виконаних завдань представлено нижче.

**Main.py**

from CustomArt import CustomArt

font = {

'm': '1',

'e':'2',

's':'3'

}

art = CustomArt('hp')

art.justify = 'right'

art.create()

print(art)

art.justify = 'center'

art.create()

print(art)

art.justify = 'left'

art.create()

print(art)

**Fonts.py**

standard = {

'$': u' \_ \n | | \n/ \_\_)\n\\\_\_ \\\n( /\n |\_| \n',

'(': u' \_\_\n / /\n| | \n| | \n| | \n \\\_\\\n',

',': u' \n \n \n \_ \n( )\n|/ \n',

'0': u' \_\_\_ \n / \_ \\ \n| | | |\n| |\_| |\n \\\_\_\_/ \n \n',

'4': u' \_ \_ \n| || | \n| || |\_ \n|\_\_ \_|\n |\_| \n \n',

'8': u' \_\_\_ \n ( \_ ) \n / \_ \\ \n| (\_) |\n \\\_\_\_/ \n \n',

'<': u' \_\_\n / /\n/ / \n\\ \\ \n \\\_\\\n \n',

'@': u' \_\_\_\_ \n / \_\_ \\ \n / / \_` |\n| | (\_| |\n \\ \\\_\_,\_|\n \\\_\_\_\_/ \n',

'D': u' \_\_\_\_ \n| \_ \\ \n| | | |\n| |\_| |\n|\_\_\_\_/ \n \n',

'H': u' \_ \_ \n| | | |\n| |\_| |\n| \_ |\n|\_| |\_|\n \n',

'L': u' \_ \n| | \n| | \n| |\_\_\_ \n|\_\_\_\_\_|\n \n',

'P': u' \_\_\_\_ \n| \_ \\ \n| |\_) |\n| \_\_/ \n|\_| \n \n',

'T': u' \_\_\_\_\_ \n|\_ \_|\n | | \n | | \n |\_| \n \n',

'X': u'\_\_ \_\_\n\\ \\/ /\n \\ / \n / \\ \n/\_/\\\_\\\n \n',

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

'`': u' \_ \n( )\n \\|\n \n \n \n',

'd': u' \_ \n \_\_| |\n / \_` |\n| (\_| |\n \\\_\_,\_|\n \n',

'h': u" \_ \n| |\_\_ \n| '\_ \\ \n| | | |\n|\_| |\_|\n \n",

'l': u' \_ \n| |\n| |\n| |\n|\_|\n \n',

'p': u" \n \_ \_\_ \n| '\_ \\ \n| |\_) |\n| .\_\_/ \n|\_| \n",

't': u' \_ \n| |\_ \n| \_\_|\n| |\_ \n \\\_\_|\n \n',

'x': u' \n\_\_ \_\_\n\\ \\/ /\n > < \n/\_/\\\_\\\n \n',

'|': u' \_ \n| |\n| |\n| |\n| |\n|\_|\n',

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

"'": u' \_ \n( )\n|/ \n \n \n \n',

'+': u' \n \_ \n \_| |\_ \n|\_ \_|\n |\_| \n \n',

'/': u' \_\_\n / /\n / / \n / / \n/\_/ \n \n',

'3': u' \_\_\_\_\_ \n|\_\_\_ / \n |\_ \\ \n \_\_\_) |\n|\_\_\_\_/ \n \n',

'7': u' \_\_\_\_\_ \n|\_\_\_ |\n / / \n / / \n /\_/ \n \n',

';': u' \n \_ \n(\_)\n \_ \n( )\n|/ \n',

'?': u' \_\_\_ \n|\_\_ \\\n / /\n |\_| \n (\_) \n \n',

'C': u' \_\_\_\_ \n / \_\_\_|\n| | \n| |\_\_\_ \n \\\_\_\_\_|\n \n',

'G': u' \_\_\_\_ \n / \_\_\_|\n| | \_ \n| |\_| |\n \\\_\_\_\_|\n \n',

'K': u" \_ \_\_\n| |/ /\n| ' / \n| . \\ \n|\_|\\\_\\\n \n",

'O': u' \_\_\_ \n / \_ \\ \n| | | |\n| |\_| |\n \\\_\_\_/ \n \n',

'S': u' \_\_\_\_ \n/ \_\_\_| \n\\\_\_\_ \\ \n \_\_\_) |\n|\_\_\_\_/ \n \n',

'W': u'\_\_ \_\_\n\\ \\ / /\n \\ \\ /\\ / / \n \\ V V / \n \\\_/\\\_/ \n \n',

'[': u' \_\_ \n| \_|\n| | \n| | \n| | \n|\_\_|\n',

'\_': u' \n \n \n \n \_\_\_\_\_ \n|\_\_\_\_\_|\n',

'c': u' \n \_\_\_ \n / \_\_|\n| (\_\_ \n \\\_\_\_|\n \n',

'g': u' \n \_\_ \_ \n / \_` |\n| (\_| |\n \\\_\_, |\n |\_\_\_/ \n',

'k': u' \_ \n| | \_\_\n| |/ /\n| < \n|\_|\\\_\\\n \n',

'o': u' \n \_\_\_ \n / \_ \\ \n| (\_) |\n \\\_\_\_/ \n \n',

's': u' \n \_\_\_ \n/ \_\_|\n\\\_\_ \\\n|\_\_\_/\n \n',

'w': u' \n\_\_ \_\_\n\\ \\ /\\ / /\n \\ V V / \n \\\_/\\\_/ \n \n',

'{': u' \_\_\n / /\n | | \n< < \n | | \n \\\_\\\n',

'"': u' \_ \_ \n( | )\n V V \n \n \n \n',

'&': u' \_\_\_ \n ( \_ ) \n / \_ \\/\\\n| (\_> <\n \\\_\_\_/\\/\n \n',

'\*': u' \n\_\_/\\\_\_\n\\ /\n/\_ \_\\\n \\/ \n \n',

'.': u' \n \n \n \_ \n(\_)\n \n',

'2': u' \_\_\_\_ \n|\_\_\_ \\ \n \_\_) |\n / \_\_/ \n|\_\_\_\_\_|\n \n',

'6': u" \_\_ \n / /\_ \n| '\_ \\ \n| (\_) |\n \\\_\_\_/ \n \n",

':': u' \n \_ \n(\_)\n \_ \n(\_)\n \n',

'>': u'\_\_ \n\\ \\ \n \\ \\\n / /\n/\_/ \n \n',

'B': u' \_\_\_\_ \n| \_\_ ) \n| \_ \\ \n| |\_) |\n|\_\_\_\_/ \n \n',

'F': u' \_\_\_\_\_ \n| \_\_\_|\n| |\_ \n| \_| \n|\_| \n \n',

'J': u' \_ \n | |\n \_ | |\n| |\_| |\n \\\_\_\_/ \n \n',

'N': u' \_ \_ \n| \\ | |\n| \\| |\n| |\\ |\n|\_| \\\_|\n \n',

'R': u' \_\_\_\_ \n| \_ \\ \n| |\_) |\n| \_ < \n|\_| \\\_\\\n \n',

'V': u'\_\_ \_\_\n\\ \\ / /\n \\ \\ / / \n \\ V / \n \\\_/ \n \n',

'Z': u' \_\_\_\_\_\n|\_\_ /\n / / \n / /\_ \n/\_\_\_\_|\n \n',

'^': u' /\\ \n|/\\|\n \n \n \n \n',

'b': u" \_ \n| |\_\_ \n| '\_ \\ \n| |\_) |\n|\_.\_\_/ \n \n",

'f': u' \_\_ \n / \_|\n| |\_ \n| \_|\n|\_| \n \n',

'j': u' \_ \n (\_)\n | |\n | |\n \_/ |\n|\_\_/ \n',

'n': u" \n \_ \_\_ \n| '\_ \\ \n| | | |\n|\_| |\_|\n \n",

'r': u" \n \_ \_\_ \n| '\_\_|\n| | \n|\_| \n \n",

'v': u' \n\_\_ \_\_\n\\ \\ / /\n \\ V / \n \\\_/ \n \n',

'z': u' \n \_\_\_\_\n|\_ /\n / / \n/\_\_\_|\n \n',

'~': u' /\\/|\n|/\\/ \n \n \n \n \n',

'!': u' \_ \n| |\n| |\n|\_|\n(\_)\n \n',

'%': u' \_ \_\_\n(\_)/ /\n / / \n / /\_ \n/\_/(\_)\n \n',

')': u'\_\_ \n\\ \\ \n | |\n | |\n | |\n/\_/ \n',

'-': u' \n \n \_\_\_\_\_ \n|\_\_\_\_\_|\n \n \n',

'1': u' \_ \n/ |\n| |\n| |\n|\_|\n \n',

'5': u' \_\_\_\_ \n| \_\_\_| \n|\_\_\_ \\ \n \_\_\_) |\n|\_\_\_\_/ \n \n',

'9': u' \_\_\_ \n / \_ \\ \n| (\_) |\n \\\_\_, |\n /\_/ \n \n',

'=': u' \n \_\_\_\_\_ \n|\_\_\_\_\_|\n|\_\_\_\_\_|\n \n \n',

'A': u' \_ \n / \\ \n / \_ \\ \n / \_\_\_ \\ \n/\_/ \\\_\\\n \n',

'E': u' \_\_\_\_\_ \n| \_\_\_\_|\n| \_| \n| |\_\_\_ \n|\_\_\_\_\_|\n \n',

'I': u' \_\_\_ \n|\_ \_|\n | | \n | | \n|\_\_\_|\n \n',

'M': u' \_\_ \_\_ \n| \\/ |\n| |\\/| |\n| | | |\n|\_| |\_|\n \n',

'Q': u' \_\_\_ \n / \_ \\ \n| | | |\n| |\_| |\n \\\_\_\\\_\\\n \n',

'U': u' \_ \_ \n| | | |\n| | | |\n| |\_| |\n \\\_\_\_/ \n \n',

'Y': u'\_\_ \_\_\n\\ \\ / /\n \\ V / \n | | \n |\_| \n \n',

']': u' \_\_ \n|\_ |\n | |\n | |\n | |\n|\_\_|\n',

'a': u' \n \_\_ \_ \n / \_` |\n| (\_| |\n \\\_\_,\_|\n \n',

'e': u' \n \_\_\_ \n / \_ \\\n| \_\_/\n \\\_\_\_|\n \n',

'i': u' \_ \n(\_)\n| |\n| |\n|\_|\n \n',

'm': u" \n \_ \_\_ \_\_\_ \n| '\_ ` \_ \\ \n| | | | | |\n|\_| |\_| |\_|\n \n",

'q': u' \n \_\_ \_ \n / \_` |\n| (\_| |\n \\\_\_, |\n |\_|\n',

'u': u' \n \_ \_ \n| | | |\n| |\_| |\n \\\_\_,\_|\n \n',

'y': u' \n \_ \_ \n| | | |\n| |\_| |\n \\\_\_, |\n |\_\_\_/ \n',

'}': u'\_\_ \n\\ \\ \n | | \n > >\n | | \n/\_/ \n',

" ": u' \n \n \n \n \n \n'}

banner3 = {

'$': 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',

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

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

'8': 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',

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

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

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

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

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

'X': 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',

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

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

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

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

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

'x': 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',

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

'7': 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',

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

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

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

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

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

'W': 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',

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

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

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

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

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

'w': 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',

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

'6': 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',

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

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

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

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

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

'V': 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',

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

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

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

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

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

'v': 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',

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

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

'9': 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',

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

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

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

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

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

'Y': 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',

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

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

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

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

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

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

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

**CustomArt.py**

from ArtGenerator import ArtGenerator

from fonts import \*

class CustomArt(ArtGenerator):

\_font = standard

\_user\_height = 5

\_user\_width = 5

\_direction\_message = ''

\_FONTS = {

'1': standard,

'2': banner3,

}

def \_custom\_justify(self):

message = self.message

if self.justify == 'center':

message = message.rjust(20, ' ')

# message = message.rjust(40, '')

elif self.justify == 'right':

message = message.rjust(40)

# message = message.rjust(80, '')

elif self.justify == 'left':

message = message.ljust(0, ' ')

# message = message.ljust(80, '')

return message

def \_create(self, \*\*kwargs):

if 'font' in kwargs:

font = kwargs['font']

else:

font = self.\_font

self.\_direction\_message = self.\_custom\_justify()

art = []

max\_lines = max(len(font.get(letter, '').split('\n')) for letter in self.\_direction\_message)

for line\_num in range(max\_lines):

line = ""

for letter in self.\_direction\_message:

letter\_lines = self.\_font.get(letter, '').split('\n')

if line\_num < len(letter\_lines):

if self.justify == 'left':

line += letter\_lines[line\_num].ljust(len(letter\_lines[0]) + 1)

elif self.justify == 'center':

line += letter\_lines[line\_num].center(len(letter\_lines[0]) + 1)

elif self.justify == 'right':

line += letter\_lines[line\_num].rjust(len(letter\_lines[0]) + 1)

else:

line += ' ' \* (len(letter\_lines[0]) + 1)

art.append(line)

output = '\n'.join(art)

return output

def \_art\_zoom(self):

art = self.\_create(font=banner3)

return art

def prev\_view(self):

art = self.\_create()

art = (self.\_color + art)

return art

def create(self):

try:

self.\_ascii\_art = self.\_create()

except NameError as e:

raise NameError(f"Error: {e}")

**ArtGenerator.py**

import pyfiglet

from colorama import Fore

class ArtGenerator:

\_message = ''

\_ascii\_art = None

\_font = 'standard'

\_color = Fore.RESET

\_width = 80

\_direction = 'auto'

\_justify = 'auto'

\_user\_height = None

\_user\_width = None

\_FONTS = {

'1': "standard",

'2': "slant",

'3': "big",

'4': "block",

'5': "bubble",

'6': "digital",

'7': "isometric1",

'8': "isometric2",

'9': "letters",

'10': "script",

'11': "shadow",

'12': "starwars",

}

\_COLORS = {

'BLACK': Fore.BLACK,

'RED': Fore.RED,

'GREEN': Fore.GREEN,

'YELLOW': Fore.YELLOW,

'BLUE': Fore.BLUE,

'MAGENTA': Fore.MAGENTA,

'CYAN': Fore.CYAN,

'WHITE': Fore.WHITE,

'RESET': Fore.RESET,

}

\_DIRECTIONS = {

'LTR': 'left-to-right',

'RTl': 'right-to-left',

}

\_JUSTIFIES = {

'CENTER': 'center',

'RIGHT': 'right',

'LEFT': 'left',

}

# GET/SET MESSAGE

@property

def message(self):

return self.\_message

@message.setter

def message(self, value):

self.\_message = value

# GET/SET FONT

@property

def font(self):

return self.\_font

@font.setter

def font(self, value):

self.\_font = value

# GET/SET COLOR

@property

def color(self):

return self.\_color

@color.setter

def color(self, value):

self.\_color = value

# GET/SET WIDTH

@property

def width(self):

return self.\_width

@width.setter

def width(self, value):

self.\_width = value

# GET/SET DIRECTION

@property

def direction(self):

return self.\_direction

@direction.setter

def direction(self, value):

self.\_direction = value

# GET/SET JUSTIFY

@property

def justify(self):

return self.\_justify

@justify.setter

def justify(self, value):

self.\_justify = value

# GET/SET User height

@property

def user\_height(self):

return self.\_user\_height

@user\_height.setter

def user\_height(self, value):

self.\_user\_height = value

# GET/SET User height

@property

def user\_width(self):

return self.\_user\_width

@user\_width.setter

def user\_width(self, value):

self.\_user\_width = value

def \_\_init\_\_(self, message=''):

self.\_message = message

self.\_ascii\_art = pyfiglet.figlet\_format(self.\_message)

def set\_custom\_font(self, symbol):

self.\_font = 'banner3'

self.create()

# self.\_ascii\_art = self.\_ascii\_art(font='')

for original\_char, user\_char in zip("#/", symbol):

self.\_ascii\_art = self.\_ascii\_art.replace(original\_char, user\_char)

def save(self, filename):

with open(filename, 'w') as file:

self.create()

file.write(self.\_ascii\_art)

def \_create(self, \*\*kwargs):

if 'font' in kwargs:

font = kwargs['font']

else:

font = self.\_font

return pyfiglet.Figlet(font=font, direction=self.\_direction, justify=self.\_justify, width=self.\_width)

def create(self):

try:

art = self.\_create()

self.\_ascii\_art = art.renderText(self.\_message)

except NameError as e:

raise NameError(f"Error: {e}")

def prev\_view(self):

art = self.\_create()

art = (self.\_color + art.renderText(self.\_message))

return art

def \_art\_zoom(self):

art = self.\_create(font='banner3')

art = art.renderText(self.\_message)

return art

def zoom(self):

line = ''

lines = []

len\_line = 0

len\_lines = 0

mx = 0

art = self.\_art\_zoom()

for i in art:

if i != '\n':

line += i

mx += 1

else:

if len\_line < mx:

len\_line = mx

mx = 0

lines += [line]

line = ''

len\_lines = lines.\_\_len\_\_()

square\_base = len\_lines \* len\_line

square\_update = self.\_user\_height \* self.\_user\_width

if square\_update >= square\_base:

pw = round(square\_update / square\_base)

else:

pw = round(square\_base / square\_update)

new\_lines = []

for line in lines:

new\_line = ''

for l in line:

new\_line += l

new\_line += l

new\_lines += [new\_line] \* pw

laq = ''

for line in new\_lines:

laq += line + '\n'

self.\_ascii\_art = laq

def \_\_str\_\_(self):

return self.\_color + self.\_ascii\_art

**Результат виконання програми**



**Висновок:** під час виконання лабораторної роботи я навчився розробляти ASCII ART генератор для візуалізації текстових даних без використання зовнішніх бібліотек.