

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

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 ІТ-32

Моляща Ю.А.

Прийняв:

Щербак С.С.

Львів - 2023

## **Лабораторна робота №4**

### **«Розробка ASCII ART генератора для візуалізації 2D-фігур »**

**Мета роботи:** створення Генератора ASCII-арту без використання зовнішніх бібліотек

#### **Завдання на лабораторну роботу.**

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.

#### **Хід роботи.**

##### `ascii_art_generator.py`

```
# Include the parent directory in the system's import path
import sys
import os

current_dir = os.path.dirname(os.path.abspath(__file__))
parent_dir = os.path.abspath(os.path.join(current_dir, '..'))
sys.path.append(parent_dir)

# Imports
from lab3.ascii_art_generator import get_phrase
from lab3.ascii_art_generator import set_size, set_symbols, set_color,
set_alignment, set_3d_option
from lab3.ascii_art_generator import check_size, preview_art
from lab4.font8x8 import font8x8

# Constants
SPACE = 0
SYMBOL = 1
SHADOW = 2

def settings(settings_obj):
    while True:
        print('Options:')
        print('0. Show current settings')
        print('1. Change size')
        print('2. Change symbol')
        print('3. Change color')
```

```

print('4. Change alignment')
print('5. Change 3D option')
print('6. Reset settings')
print('7. Back')

user_input = input('Enter option number: ')
if user_input == '0':
    settings_obj.show_settings()
elif user_input == '1':
    settings_obj.set_size(*set_size())
elif user_input == '2':
    settings_obj.set_symbols(*set_symbols())
elif user_input == '3':
    settings_obj.set_color(set_color())
elif user_input == '4':
    settings_obj.set_alignment(set_alignment())
elif user_input == '5':
    settings_obj.set_3d_option(set_3d_option())
elif user_input == '6':
    settings_obj.default_settings()
elif user_input == '7':
    break

def str_to_ascii_list(char_str):
    """Convert string to list of ASCII codes."""
    char_list = []
    for char in char_str:
        char_list.append(ord(char))
    return char_list

def row_to_string(row, regular_symbol, shadow_symbol):
    """Convert row items (SPACE, SYMBOL, SHADOW) to requested symbols."""
    row_string = ""
    for item in row:

```

```

        if item == SPACE:
            row_string += " "
        elif item == SYMBOL:
            row_string += regular_symbol
        elif item == SHADOW:
            row_string += shadow_symbol
    return row_string

def add_shadow(set_bit_list):
    """Add shadow to ASCII-art.

    Check if there is a SYMBOL and SPACE next to each other.
    If so, replace SPACE with SHADOW.

    """
    for i in range(len(set_bit_list)):
        if set_bit_list[i] == SYMBOL and set_bit_list[i + 1] == SPACE:
            set_bit_list[i] = SHADOW

    return set_bit_list

def twod_to_threed(set_bit_list, column_item, char_item):
    """Add 3D effect to ASCII-art.

    For the first character in the row, add SPACE to the beginning of the row.

    In ascending order, add SPACE to the beginning of the rows that represent a
    character.

    """
    if char_item == 0:
        for _ in range(column_item, 8):
            set_bit_list.insert(0, SPACE)
    return set_bit_list

def set_lines(char_list, width):
    """Get number of lines for ASCII-art.

    If string is longer than width, split string into multiple lines.

    """

```

```

char_width = 8
if len(char_list) * char_width > width:
    lines_list = []
    chars_in_line = width // 8
    for i in range(len(char_list) // chars_in_line + 1):
        lines_list.append(char_list[i * chars_in_line: (i + 1) *
chars_in_line])
    else:
        lines_list = [char_list]
    return lines_list

def align(alignment, width, row_string):
    if alignment == 'left':
        left_padding = 0
    elif alignment == 'center':
        left_padding = (width - len(row_string)) // 2
    elif alignment == 'right':
        left_padding = (width - len(row_string))

    return left_padding

def render(char_str, color, regular_symbol, shadow_symbol, width, height,
alignment, threaded):
    """Render ASCII-art."""
    char_width = 8
    char_height = 8
    art = ""

    ascii_list = str_to_ascii_list(char_str)
    lines = set_lines(ascii_list, width)

    for line in lines:
        for column_item in range(char_height):
            row = ""
            for char_item in range(len(line)):

```

```

        set_bit_list = []
        for row_item in range(char_width):
            bitmap = font8x8[line[char_item]]
            set_bit = (1 << row_item) & bitmap[column_item]
            if set_bit:
                set_bit_list.append(SYMBOL)
            else:
                set_bit_list.append(SPACE)

        if shadow_symbol:
            set_bit_list = add_shadow(set_bit_list)

        if threed:
            set_bit_list = twod_to_threed(set_bit_list, column_item,
char_item)

            row_string = row_to_string(set_bit_list, regular_symbol,
shadow_symbol)

            row += row_string
            left_padding = align(alignment, width, row_string)
            art += " " * left_padding + row + "\n"
            art = color + art

        return art

def create_ascii_art(FOLDER_PATH, settings_obj):
    """Ask user for phrase and show ASCII-art.

    Args:
        FOLDER_PATH (str): Path to folder with ASCII-arts.
        settings_obj (AsciiArtSettings): Object with settings.
    """
    char_str = get_phrase()
    width, height = settings_obj.size

    try:

```

```

        check_size(char_str, width, height)
except ValueError as e:
    print(f"An error occurred: {e}")
    return None

while True:
    art = render(char_str,
                  settings_obj.color,
                  settings_obj.symbols[0],
                  settings_obj.symbols[1],
                  settings_obj.size[0],
                  settings_obj.size[1],
                  settings_obj.alignment,
                  settings_obj.is_3d)

    try:
        preview_art(FOLDER_PATH, art)
    except Exception as e:
        print(f"An error occurred while previewing the ASCII art:
{str(e)}")

    change_settings = input('Do you want to change settings? (y/n): ')

    if change_settings == 'y':
        settings(settings_obj)
    else:
        break

```

### font8x8.py

```

# Source: https://github.com/dhepper/font8x8
# 8x8 pixel bitmap
# 128x8

# The character 'A' (0x41 / 65) is encoded as
# { 0x0C, 0x1E, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x00}

```



```
#      0x0C => 0000 1100 => ..XX...
#      0x1E => 0001 1110 => .XXXX...
#      0x33 => 0011 0011 => XX..XX..
#      0x33 => 0011 0011 => XX..XX..
#      0x3F => 0011 1111 => xxxxxxx..
#      0x33 => 0011 0011 => XX..XX..
#      0x33 => 0011 0011 => XX..XX..
#      0x00 => 0000 0000 => .....
```

```
font8x8 = [
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0000 (nul)
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0001
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0002
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0003
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0004
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0005
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0006
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0007
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0008
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0009
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000A
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000B
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000C
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000D
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000E
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+000F
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0010
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0011
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0012
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0013
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0014
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0015
    [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0016
```

```
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0017
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0018
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0019
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001A
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001B
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001C
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001D
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001E
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+001F
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0020 (space)
[ 0x18, 0x3C, 0x3C, 0x18, 0x18, 0x00, 0x18, 0x00], # U+0021 (!)
[ 0x36, 0x36, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0022 (")
[ 0x36, 0x36, 0x7F, 0x36, 0x7F, 0x36, 0x36, 0x00], # U+0023 (#)
[ 0x0C, 0x3E, 0x03, 0x1E, 0x30, 0x1F, 0x0C, 0x00], # U+0024 ($)
[ 0x00, 0x63, 0x33, 0x18, 0x0C, 0x66, 0x63, 0x00], # U+0025 (%)
[ 0x1C, 0x36, 0x1C, 0x6E, 0x3B, 0x33, 0x6E, 0x00], # U+0026 (&)
[ 0x06, 0x06, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0027 (' )
[ 0x18, 0x0C, 0x06, 0x06, 0x06, 0x0C, 0x18, 0x00], # U+0028 (( )
[ 0x06, 0x0C, 0x18, 0x18, 0x18, 0x0C, 0x06, 0x00], # U+0029 ( ) )
[ 0x00, 0x66, 0x3C, 0xFF, 0x3C, 0x66, 0x00, 0x00], # U+002A (*)
[ 0x00, 0x0C, 0x0C, 0x3F, 0x0C, 0x0C, 0x00, 0x00], # U+002B (+)
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C, 0x06], # U+002C (,)
[ 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00], # U+002D (-)
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C, 0x00], # U+002E (.)
[ 0x60, 0x30, 0x18, 0x0C, 0x06, 0x03, 0x01, 0x00], # U+002F (/)
[ 0x3E, 0x63, 0x73, 0x7B, 0x6F, 0x67, 0x3E, 0x00], # U+0030 (0)
[ 0x0C, 0x0E, 0x0C, 0x0C, 0x0C, 0x0C, 0x3F, 0x00], # U+0031 (1)
[ 0x1E, 0x33, 0x30, 0x1C, 0x06, 0x33, 0x3F, 0x00], # U+0032 (2)
[ 0x1E, 0x33, 0x30, 0x1C, 0x30, 0x33, 0x1E, 0x00], # U+0033 (3)
[ 0x38, 0x3C, 0x36, 0x33, 0x7F, 0x30, 0x78, 0x00], # U+0034 (4)
[ 0x3F, 0x03, 0x1F, 0x30, 0x30, 0x33, 0x1E, 0x00], # U+0035 (5)
[ 0x1C, 0x06, 0x03, 0x1F, 0x33, 0x33, 0x1E, 0x00], # U+0036 (6)
[ 0x3F, 0x33, 0x30, 0x18, 0x0C, 0x0C, 0x0C, 0x00], # U+0037 (7)
[ 0x1E, 0x33, 0x33, 0x1E, 0x33, 0x33, 0x1E, 0x00], # U+0038 (8)
```

[ 0x1E, 0x33, 0x33, 0x3E, 0x30, 0x18, 0x0E, 0x00], # U+0039 (9)  
[ 0x00, 0x0C, 0x0C, 0x00, 0x00, 0x0C, 0x0C, 0x00], # U+003A (:)  
[ 0x00, 0x0C, 0x0C, 0x00, 0x00, 0x0C, 0x0C, 0x06], # U+003B (;)  
[ 0x18, 0x0C, 0x06, 0x03, 0x06, 0x0C, 0x18, 0x00], # U+003C (<)  
[ 0x00, 0x00, 0x3F, 0x00, 0x00, 0x3F, 0x00, 0x00], # U+003D (=)  
[ 0x06, 0x0C, 0x18, 0x30, 0x18, 0x0C, 0x06, 0x00], # U+003E (>)  
[ 0x1E, 0x33, 0x30, 0x18, 0x0C, 0x00, 0x0C, 0x00], # U+003F (?)  
[ 0x3E, 0x63, 0x7B, 0x7B, 0x7B, 0x03, 0x1E, 0x00], # U+0040 (@)  
[ 0x0C, 0x1E, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x00], # U+0041 (A)  
[ 0x3F, 0x66, 0x66, 0x3E, 0x66, 0x66, 0x3F, 0x00], # U+0042 (B)  
[ 0x3C, 0x66, 0x03, 0x03, 0x03, 0x66, 0x3C, 0x00], # U+0043 (C)  
[ 0x1F, 0x36, 0x66, 0x66, 0x66, 0x36, 0x1F, 0x00], # U+0044 (D)  
[ 0x7F, 0x46, 0x16, 0x1E, 0x16, 0x46, 0x7F, 0x00], # U+0045 (E)  
[ 0x7F, 0x46, 0x16, 0x1E, 0x16, 0x06, 0x0F, 0x00], # U+0046 (F)  
[ 0x3C, 0x66, 0x03, 0x03, 0x73, 0x66, 0x7C, 0x00], # U+0047 (G)  
[ 0x33, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x33, 0x00], # U+0048 (H)  
[ 0x1E, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x1E, 0x00], # U+0049 (I)  
[ 0x78, 0x30, 0x30, 0x30, 0x33, 0x33, 0x1E, 0x00], # U+004A (J)  
[ 0x67, 0x66, 0x36, 0x1E, 0x36, 0x66, 0x67, 0x00], # U+004B (K)  
[ 0x0F, 0x06, 0x06, 0x06, 0x46, 0x66, 0x7F, 0x00], # U+004C (L)  
[ 0x63, 0x77, 0x7F, 0x7F, 0x6B, 0x63, 0x63, 0x00], # U+004D (M)  
[ 0x63, 0x67, 0x6F, 0x7B, 0x73, 0x63, 0x63, 0x00], # U+004E (N)  
[ 0x1C, 0x36, 0x63, 0x63, 0x63, 0x36, 0x1C, 0x00], # U+004F (O)  
[ 0x3F, 0x66, 0x66, 0x3E, 0x06, 0x06, 0x0F, 0x00], # U+0050 (P)  
[ 0x1E, 0x33, 0x33, 0x33, 0x3B, 0x1E, 0x38, 0x00], # U+0051 (Q)  
[ 0x3F, 0x66, 0x66, 0x3E, 0x36, 0x66, 0x67, 0x00], # U+0052 (R)  
[ 0x1E, 0x33, 0x07, 0x0E, 0x38, 0x33, 0x1E, 0x00], # U+0053 (S)  
[ 0x3F, 0x2D, 0x0C, 0x0C, 0x0C, 0x0C, 0x1E, 0x00], # U+0054 (T)  
[ 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x3F, 0x00], # U+0055 (U)  
[ 0x33, 0x33, 0x33, 0x33, 0x33, 0x1E, 0x0C, 0x00], # U+0056 (V)  
[ 0x63, 0x63, 0x63, 0x6B, 0x7F, 0x77, 0x63, 0x00], # U+0057 (W)  
[ 0x63, 0x63, 0x36, 0x1C, 0x1C, 0x36, 0x63, 0x00], # U+0058 (X)  
[ 0x33, 0x33, 0x33, 0x1E, 0x0C, 0x0C, 0x1E, 0x00], # U+0059 (Y)  
[ 0x7F, 0x63, 0x31, 0x18, 0x4C, 0x66, 0x7F, 0x00], # U+005A (Z)

[ 0x1E, 0x06, 0x06, 0x06, 0x06, 0x06, 0x1E, 0x00], # U+005B ([)  
[ 0x03, 0x06, 0x0C, 0x18, 0x30, 0x60, 0x40, 0x00], # U+005C (\)  
[ 0x1E, 0x18, 0x18, 0x18, 0x18, 0x18, 0x1E, 0x00], # U+005D (])  
[ 0x08, 0x1C, 0x36, 0x63, 0x00, 0x00, 0x00, 0x00], # U+005E (^)  
[ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF], # U+005F (\_)  
[ 0x0C, 0x0C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00], # U+0060 (`)  
[ 0x00, 0x00, 0x1E, 0x30, 0x3E, 0x33, 0x6E, 0x00], # U+0061 (a)  
[ 0x07, 0x06, 0x06, 0x3E, 0x66, 0x66, 0x3B, 0x00], # U+0062 (b)  
[ 0x00, 0x00, 0x1E, 0x33, 0x03, 0x33, 0x1E, 0x00], # U+0063 (c)  
[ 0x38, 0x30, 0x30, 0x3e, 0x33, 0x33, 0x6E, 0x00], # U+0064 (d)  
[ 0x00, 0x00, 0x1E, 0x33, 0x3f, 0x03, 0x1E, 0x00], # U+0065 (e)  
[ 0x1C, 0x36, 0x06, 0x0f, 0x06, 0x06, 0x0F, 0x00], # U+0066 (f)  
[ 0x00, 0x00, 0x6E, 0x33, 0x33, 0x3E, 0x30, 0x1F], # U+0067 (g)  
[ 0x07, 0x06, 0x36, 0x6E, 0x66, 0x66, 0x67, 0x00], # U+0068 (h)  
[ 0x0C, 0x00, 0x0E, 0x0C, 0x0C, 0x0C, 0x1E, 0x00], # U+0069 (i)  
[ 0x30, 0x00, 0x30, 0x30, 0x30, 0x33, 0x33, 0x1E], # U+006A (j)  
[ 0x07, 0x06, 0x66, 0x36, 0x1E, 0x36, 0x67, 0x00], # U+006B (k)  
[ 0x0E, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x1E, 0x00], # U+006C (l)  
[ 0x00, 0x00, 0x33, 0x7F, 0x7F, 0x6B, 0x63, 0x00], # U+006D (m)  
[ 0x00, 0x00, 0x1F, 0x33, 0x33, 0x33, 0x33, 0x00], # U+006E (n)  
[ 0x00, 0x00, 0x1E, 0x33, 0x33, 0x33, 0x1E, 0x00], # U+006F (o)  
[ 0x00, 0x00, 0x3B, 0x66, 0x66, 0x3E, 0x06, 0x0F], # U+0070 (p)  
[ 0x00, 0x00, 0x6E, 0x33, 0x33, 0x3E, 0x30, 0x78], # U+0071 (q)  
[ 0x00, 0x00, 0x3B, 0x6E, 0x66, 0x06, 0x0F, 0x00], # U+0072 (r)  
[ 0x00, 0x00, 0x3E, 0x03, 0x1E, 0x30, 0x1F, 0x00], # U+0073 (s)  
[ 0x08, 0x0C, 0x3E, 0x0C, 0x0C, 0x2C, 0x18, 0x00], # U+0074 (t)  
[ 0x00, 0x00, 0x33, 0x33, 0x33, 0x33, 0x6E, 0x00], # U+0075 (u)  
[ 0x00, 0x00, 0x33, 0x33, 0x33, 0x1E, 0x0C, 0x00], # U+0076 (v)  
[ 0x00, 0x00, 0x63, 0x6B, 0x7F, 0x7F, 0x36, 0x00], # U+0077 (w)  
[ 0x00, 0x00, 0x63, 0x36, 0x1C, 0x36, 0x63, 0x00], # U+0078 (x)  
[ 0x00, 0x00, 0x33, 0x33, 0x33, 0x3E, 0x30, 0x1F], # U+0079 (y)  
[ 0x00, 0x00, 0x3F, 0x19, 0x0C, 0x26, 0x3F, 0x00], # U+007A (z)  
[ 0x38, 0x0C, 0x0C, 0x07, 0x0C, 0x0C, 0x38, 0x00], # U+007B ({)  
[ 0x18, 0x18, 0x18, 0x00, 0x18, 0x18, 0x18, 0x00], # U+007C (|)

```

        [ 0x07, 0x0C, 0x0C, 0x38, 0x0C, 0x0C, 0x07, 0x00],    # U+007D (})
        [ 0x6E, 0x3B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00],    # U+007E (~)
        [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]    # U+007F
    ]

```

## main.py

```

# Include the parent directory in the system's import path
import sys
import os

current_dir = os.path.dirname(os.path.abspath(__file__))
parent_dir = os.path.abspath(os.path.join(current_dir, '..'))
sys.path.append(parent_dir)

# Imports
from lab3.ascii_art_settings import AsciiArtSettings
from lab3.ascii_art_generator import show_art
from lab4.ascii_art_generator import settings, create_ascii_art

# Constants
FOLDER_PATH = 'source/lab4/ASCII-arts/'
SETTINGS_FILE_PATH = 'source/lab4/settings.json'

def main():
    try:
        settings_obj = AsciiArtSettings()
        settings_obj.set_settings_file_path(SETTINGS_FILE_PATH)
        settings_obj.load_settings()

    while True:
        print('Options (1/2/3):')
        print('1. Create ASCII-art')
        print('2. Show ASCII-art')
        print('3. Settings')
        print('4. Exit')

```

```

        user_input = input('Enter option number: ')

        if user_input == '1':
            create_ascii_art(FOLDER_PATH, settings_obj)
        elif user_input == '2':
            show_art(FOLDER_PATH)
        elif user_input == '3':
            settings(settings_obj)
        elif user_input == '4':
            break

    except ValueError as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()

```

settings.json

```

{"font": "clb6x10", "size": [80, 25], "symbols": ["$", ""], "color":
"\u001b[32m", "alignment": "left", "is_3d": true}

```

**Висновок:** під час виконання лабораторної роботи було створено генератор ASCII-арту без використання зовнішніх бібліотек