МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

**Звіт**

До лабораторної роботи №4

**Розробка ASCII ART генератора для візуалізації 2D-фігур**

Виконав:

ст. гр. ІТ-21сп

**Олександр БІЗЯЄВ**

Прийняв:

доц. каф. ІСМ

**Сергій ЩЕРБАК**

Львів ‒ 2023

**Лабораторна робота №4**

**Мета:** створення додатка 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.

Файл **art.py**

from labwork\_4 import fonts  
  
  
class Art:  
 def \_\_init\_\_(self):  
 self.font = fonts.banner  
  
 def create(self, direction\_message):  
 max\_lines = max(len(self.font.get(letter, '').split('\n')) for letter in direction\_message)  
 art = ['' for \_ in range(max\_lines)]  
  
 for letter in direction\_message:  
 letter\_lines = self.font.get(letter, '').split('\n')  
 art = [''.join([line1, line2]) for line1, line2 in zip(art, letter\_lines)]  
  
 return '\n'.join(art)

Файл **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',  
}

Файл **modifiedArt.py**

from labwork\_4 import variables  
from labwork\_4.art import Art  
from labwork\_4.functions import Function  
  
  
class CustomArt(Art):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.alignment = 'center'  
 self.width = 50  
 self.character = '\*'  
 self.art = Art()  
 self.functions = Function()  
  
 def default\_character(self, character):  
 if character is None or character.strip() == "":  
 character = self.character  
 return character  
 else:  
 return character  
  
 def default\_alignment(self, alignment):  
 if (alignment is None or alignment.strip() == ""  
 or (alignment != 'left' and alignment != 'right')):  
 alignment = self.alignment  
 return alignment  
 else:  
 return alignment  
  
 def default\_width(self, width):  
 if width is None or width <= 0:  
 width = self.width  
 return width  
 else:  
 return width  
  
 def \_check\_parameters(self, width, alignment, character):  
 width = self.default\_width(width)  
 alignment = self.default\_alignment(alignment)  
 character = self.default\_character(character)  
 return width, alignment, character  
  
 def create\_all(self, direction\_message, width, alignment, character):  
 width, alignment, character = self.\_check\_parameters(width, alignment, character)  
 art\_with\_width = self.\_create\_width(direction\_message, width)  
 art\_with\_alignment = self.\_create\_alignment(art\_with\_width, width, alignment)  
 art\_with\_character = self.\_create\_character(art\_with\_alignment, character)  
 return art\_with\_character  
  
 def \_create\_width(self, direction\_message, max\_width):  
 max\_lines = max(len(self.font.get(letter, '').split('\n')) for letter in direction\_message)  
 alignment\_func = str.center  
  
 line = [""] \* max\_lines  
 line\_width = 0  
 art = []  
  
 for word in direction\_message:  
 letter\_lines = self.font.get(word, '').split('\n')  
 word\_width = len(letter\_lines[0]) + 1  
  
 if line\_width + word\_width <= max\_width:  
 for i in range(max\_lines):  
 line[i] += alignment\_func(letter\_lines[i], word\_width)  
 line\_width += word\_width  
 else:  
 art.append('\n'.join(line))  
 line = [""] \* max\_lines  
 for i in range(max\_lines):  
 line[i] += alignment\_func(letter\_lines[i], word\_width)  
 line\_width = word\_width  
  
 art.append('\n'.join(line))  
 output = '\n'.join(art)  
 return output  
  
 def \_create\_alignment(self, direction\_message, width, alignment):  
  
 output\_lines = direction\_message.split('\n')  
 aligned\_output\_lines = []  
  
 for line in output\_lines:  
 if alignment == variables.alignment\_left:  
 aligned\_line = f"{' '}{line}"  
 elif alignment == variables.alignment\_right:  
 aligned\_line = f"{line.rjust(width)}"  
 elif alignment == variables.alignment\_center:  
 aligned\_line = f"{line.center(width)}"  
 else:  
 aligned\_line = line  
  
 aligned\_output\_lines.append(aligned\_line)  
  
 aligned\_output = '\n'.join(aligned\_output\_lines)  
  
 return aligned\_output  
  
 def \_create\_character(self, direction\_message, character):  
  
 lines = direction\_message.split('\n')  
  
 for i in range(len(lines)):  
 line = lines[i]  
 new\_line = ''  
 for char in line:  
 if char != ' ':  
 new\_line += character  
 else:  
 new\_line += ' '  
 lines[i] = new\_line  
  
 modified\_ascii\_art = '\n'.join(lines)  
  
 return modified\_ascii\_art

Файл **functions.py**

from colorama import Fore, Back  
  
  
class Function:  
  
 def \_choose\_color(self, user\_color):  
 if user\_color == '1':  
 return Fore.LIGHTBLACK\_EX + Back.WHITE  
 elif user\_color == '2':  
 return Fore.BLACK + Back.WHITE  
 elif user\_color == '3':  
 return Fore.WHITE + Back.BLACK  
 elif user\_color == '4':  
 return Fore.WHITE + Back.LIGHTBLACK\_EX

Файл **variables.py**

user\_word\_input = "Please enter a word or phrase: "  
user\_color\_input = "Colors: \n 1. 'gray and white'\n 2. 'black and white'\n 3. 'white and black'\n 4. 'white and gray'\nPlease choose color : "  
user\_alignment\_input = "Please enter the alignment(left, right, center): "  
width\_input = "Please enter the width (optimal 50): "  
user\_character\_input = "What character you want use?: "  
  
error\_color = "Error: You have selected the wrong color number"  
error\_width = "Error: Cannot be less than 50"  
error\_alignment = "Error: Must be string"  
  
choose\_exit = "Do you want to make another art? (Yes/No): "  
  
ask\_change\_art = "Do you want to change the art? (Yes/No): "  
file\_name = "Please enter file name: "  
  
alignment\_left = "left"  
alignment\_right = 'right'  
alignment\_center = "center"  
  
choice\_yes = "yes"

Файл **errors.py**

class Error:  
 def check\_widht(self, parameter):  
 try:  
 parameter = int(parameter)  
 if parameter < 50:  
 raise ValueError  
 except ValueError:  
 return ValueError()  
  
 def check\_color(self, number):  
 try:  
 number = int(number)  
 if number <= 0 or number > 4:  
 raise ValueError  
 except ValueError:  
 return ValueError()

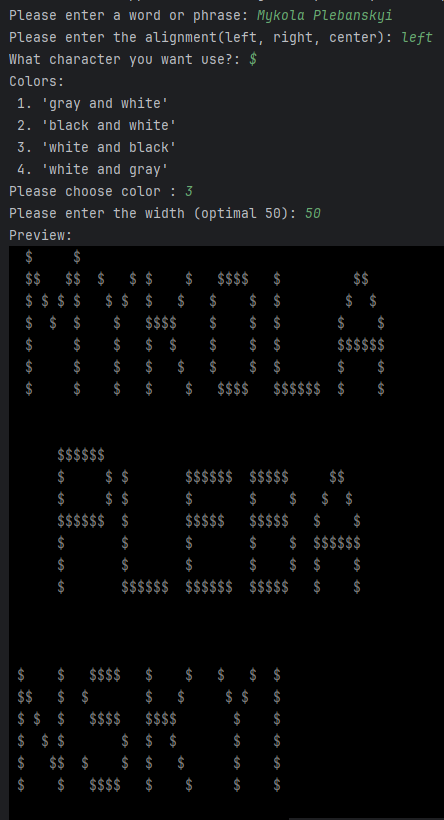
Файл **interface.py**

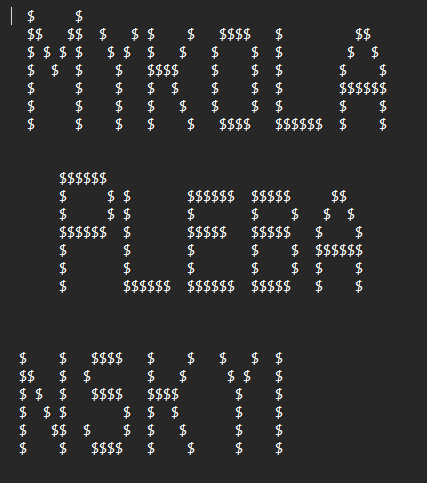
import re  
  
from colorama import Fore, Style  
  
from labwork\_5 import variables  
from labwork\_5.errors import Error  
from labwork\_5.parallelepiped import Parallelepiped  
  
  
class Interface:  
 def \_\_init\_\_(self):  
 self.error = Error()  
  
 def \_choice\_exit(self, choice):  
 if choice.lower() not in [variables.choice\_yes, variables.choice\_y]:  
 return True  
  
 def \_choice(self, choice):  
 if choice.lower() in [variables.choice\_yes, variables.choice\_y]:  
 return True  
  
 def user\_input(self):  
 while True:  
 width, height, length = self.input\_parameters()  
 user\_parallelepiped = Parallelepiped(width, height, length)  
 print(variables.colors)  
 colors = self.input\_colors(user\_parallelepiped)  
 user\_parallelepiped = user\_parallelepiped.display\_parallelepiped(number=1, colors=colors)  
 print(user\_parallelepiped)  
 choose\_zoom = input(variables.choose\_zoom)  
 if self.\_choice(choose\_zoom):  
 parallelepiped\_zoom = Parallelepiped(width, height, length)  
 user\_parallelepiped = parallelepiped\_zoom.display\_parallelepiped(number=2, colors=colors)  
 print(user\_parallelepiped)  
 change\_selection = input(variables.ask\_save\_art)  
 if self.\_choice(change\_selection):  
 name\_file = input(variables.file\_name)  
 self.save\_to\_file(name\_file, user\_parallelepiped)  
 choose\_exit = input(variables.choose\_exit)  
 if self.\_choice\_exit(choose\_exit):  
 break  
  
 def input\_parameters(self):  
 width = self.get\_valid\_input(variables.user\_width, self.error.check\_width\_and\_height,  
 variables.error\_width\_and\_height)  
 height = self.get\_valid\_input(variables.user\_height, self.error.check\_width\_and\_height,  
 variables.error\_width\_and\_height)  
 length = self.get\_valid\_length(variables.user\_length, height, self.error.check\_length, variables.error\_length)  
  
 return width, height, length  
  
 def get\_valid\_input(self, prompt, error\_check, error\_message):  
 while True:  
 value = input(prompt).strip()  
 if not self.error.check\_empty(value):  
 return None  
 if error\_check(value):  
 print(Fore.RED + Style.BRIGHT + error\_message + Style.RESET\_ALL)  
 else:  
 return int(value)  
  
 def get\_valid\_length(self, prompt, height, error\_check, error\_message):  
 while True:  
 length = input(prompt).strip()  
 if not self.error.check\_empty(length):  
 return None  
 if error\_check(height, length):  
 print(Fore.RED + Style.BRIGHT + error\_message + Style.RESET\_ALL)  
 else:  
 return int(length)  
  
 def input\_colors(self, parallelepiped):  
 custom\_color1 = self.get\_input\_color(variables.color\_left, self.is\_valid\_color,  
 parallelepiped.default\_colors['color1'])  
 custom\_color2 = self.get\_input\_color(variables.color\_top, self.is\_valid\_color,  
 parallelepiped.default\_colors['color2'])  
 custom\_color3 = self.get\_input\_color(variables.color\_front, self.is\_valid\_color,  
 parallelepiped.default\_colors['color3'])  
 custom\_color4 = self.get\_input\_color(variables.color\_shadow, self.is\_valid\_color,  
 parallelepiped.default\_colors['color4'])  
  
 parallelepiped\_colors = {  
 'color1': custom\_color1,  
 'color2': custom\_color2,  
 'color3': custom\_color3,  
 'color4': custom\_color4  
 }  
 return parallelepiped\_colors  
  
 def is\_valid\_color(self, color):  
 colors = {  
 'black': '\033[40m',  
 'red': '\033[41m',  
 'green': '\033[42m',  
 'yellow': '\033[43m',  
 'blue': '\033[44m',  
 'magenta': '\033[45m',  
 'cyan': '\033[46m',  
 'white': '\033[47m',  
 }  
 return color.lower() in colors  
  
 def get\_input\_color(self, entered\_value, error\_check, default\_value):  
 while True:  
 value = input(entered\_value).strip().lower()  
 if value in {'', ' ', None} or not error\_check(value):  
 return default\_value  
 return value  
  
 def save\_to\_file(self, file\_name, user\_parallelepiped):  
 file\_name = f"{file\_name}.txt"  
 stripped\_text = self.strip\_ansi(user\_parallelepiped)  
  
 with open(file\_name, "w") as file:  
 file.write(stripped\_text)  
  
 def strip\_ansi(self, text):  
 ansi\_escape = re.compile(r'\x1B(?:[@-Z\\-\_]|\[[0-?]\*[ -/]\*[@-~])')  
 return ansi\_escape.sub('', text)

Файл **main.py**

from labwork\_4.interface import Interface  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 interface = Interface()  
 interface.user\_input()

Результат виконання всіх завдань:





**Рис.1** Результат виконання

**Висновок:** Виконавши ці завдання, я розробив додаток Генератора ASCII-арту без сторонніх бібліотек.