

Лабораторна робота №2

Тема: Робота зі списками, функціями та створення модулів в Python. Створення файлів з даними.

Мета роботи: навчитися працювати з розширеною структурою додатку, яка включатиме модулі та файли з даними. Реалізувати додаток-гру "HANGMAN".

ТЕОРЕТИЧНА ЧАСТИНА

Для глибшого розуміння принципу роботи зі списками в Python, ознайомтеся з наступною документацією: <https://developers.google.com/edu/python/lists#for-and-in>

В середовищі Python виконайте наступні дії:

```
import random #import random module
colors = ['red', 'blue', 'green'] #create list
new_list = colors #duplicate list to new variable
random.shuffle(colors)
print(new_list) #what is the expected outcome of this output? Why?
```

Поясніть яким буде результат виконання останньої операції та чому (більше ніж одним реченням).

ПОРЯДОК ВИКОНАННЯ

Частина 1. Завдання складністю до 75 балів.

1. Зіграйте в гру: <https://hangmanwordgame.com/?fca=1&success=0#/>. Ознайомтеся з логікою гри представленої в блок-схемі на рис. 1.
2. Зробіть форк репозиторію: <https://replit.com/@MikolaSich/2-Lab-HANGMAN#main.py>.
3. Ознайомтеся з лістингом скрипту main.py та виконайте його. В даній програмі реалізовано наступні елементи:
 - створено масив **word_list** з трьох слів ;
 - імпортовано модуль **random**, методом **random.choice()** з якого обирається випадкове слово з масива та зберігається у змінну **chosen_word**;
 - користувач вводить літеру в консолі, яка зберігається у змінну **guess** (відбувається її перетворення на літеру нижнього регістра, зверніть увагу, усі літери в списку - в нижньому регістрі);
 - цикл **for letter in chosen_word** перевіряє відповідність літери введеної користувачем до кожного елемента (літери) слова, обраного випадково;

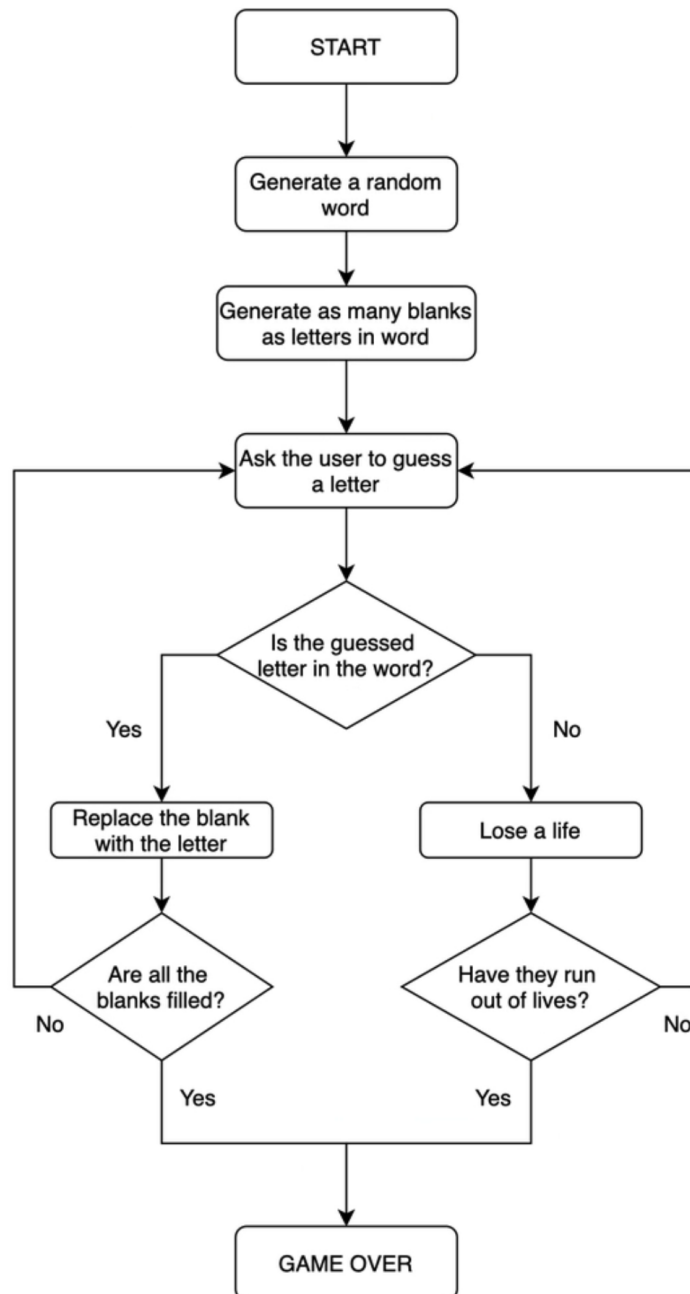


Рис. 1 - Блок-схема

4. Для реалізації логіки (block1) роботи гри виконайте наступні кроки:

- створіть список **display**. У цей список повинно додаватися стільки ж символів “_” скільки літер у випадково обраному слові. Наприклад, якщо **chosen_word** має значення “apple” то **display** повинно мати ["_", "_", "_", "_", "_"]
- після того, як користувач введе літеру, використовуючи цикл пройдіть по кожному елементу масиву **display** та замініть “_” на літеру, якщо вона співпала. Наприклад, якщо користувач ввів літеру “p” то в масиви стане ["_", "p", "p", "_", "_"]. Необхідні методи для роботи з масивом:

<https://developers.google.com/edu/python/lists#list-methods>

- виведіть змінну **display** та перевірте виконання написаного блоку програми;

5. Уважно прочитайте як працює конструкція for x in y за посиланням


<https://developers.google.com/edu/python/lists#range>

6. Наступний блок програми (block2) має визначати чи виграв гравець, вгадавши всі літери в слові **chosen_word**. Як варіант, змінна **end_of_game** буде визначати кінець гри за допомогою циклу **while**.

Частина 2. Завдання складністю до 90 балів.

7. У файлі **plus.py** знаходяться зображення для кожної невдалої спроби, які з'являються після невдалої спроби відгадати слово. Після 6 спроб гра закінчується відповідним зображенням та словами **GAME OVER**.
8. Додайте список **attempts**, який не дасть користувачу ще раз ввести літеру, яку він вже намагався відгадати.

Частина 3. Завдання складністю до 100 балів.

9. За допомогою модуля **pickle** створіть файл **.dat** який міститиме 100 слів, з яких і обиратиметься слово, яке необхідно вгадати. Модуль добре описаний в підручнику на ст. 146:  Python For Kids.pdf
10. Розфарбуйте інтерфейс за допомогою модуля **colorama**.
11. З модулем **from replit import clear** оновлюйте інтерфейс вікна, кожного разу очищаючи попередню інформацію.
12. Створіть свої модулі, підключіть їх та зменшіть кількість коду у файлі **main.py**, використовуючи свої функції.