Завдання 2. Шибениця

Це завдання стосуватиметься теми створення функцій та циклів, що дозволяють повторювати обчислення доки не буде виконана певна умова, у Python.

Про співпрацю:
Ви можете працювати над цим завданням разом з іншими студентами. Однак, кожен має написати і здати власну версію виконаного завдання. *Переконайтеся, що ви вказали з ким виконували це завдання у коментарях до вашого коду.*

Задача 1: Шибениця

Ви реалізуєте варіант класичної гри у слова під назвою Шибениця. Якщо вам не знайомі правила цієї гри, ви можете прочитати про них у Інтернеті (укр, en). Нехай складність цього завдання не деморалізує вас — воно простіше, ніж здається! Ми закладемо основу для вирішення, разом підготувавши допоміжні функції перед тим, як ви реалізуєте саму гру.

A) Початок

Завантажте файли `hangman.py` та `words.txt` і збережіть їх у одну й ту саму папку. Запустіть файл `hangman.py` до того, як напишете будь-який код, аби переконатися, що файли збереглися коректно. Цей код завантажує слова з файлу. Ви маєте побачити у консолі наступний текст:

```
Loading word list from file...
55900 words loaded.
```

Якщо ви побачили його, переходьте до наступного кроку. Якщо ні — перевірте чи обидва файли збережено в одному місці.

B) Вимоги до Шибениці

Ви маєте реалізувати функцію `hangman`, що дозволить користувачу грати у Шибеницю проти комп'ютера. Комп'ютер буде вибирати слово, а користувач намагатиметься відгадати його по буквах.

Ось загальний опис поведінки, яку ми хочемо реалізувати. Не впадайте у відчай! Це просто опис; ми розіб'ємо його на окремі кроки і додаткові функціональні вимоги пізніше.

1. Комп'ютер має вибрати слово випадковим чином зі списку доступних слів, поданих у файлі.
   Помітьте, що всі слова у файлі вказані у нижньому регістрі.
2. Користувач на початку має певну кількість спроб.

3. Гра є інтерактивною; користувач вводить свій варіант, а комп'ютер:
    a. відображає усі входження цієї літери у слові, якщо вона там є;
    b. зменшує кількість доступних спроб, якщо ні.
4. Гра закінчується або коли гравець використає всі свої спроби, або коли вгадає слово повністю.

Задача 2:
Шибениця. Частина 1: Три допоміжні функції

Перш ніж ми перейдемо до написання коду для організації власне гри у шибеницю, ми маємо розбити її на логічні підпроблеми, створивши три допоміжні функції, які знадобляться вам для того, щоб реалізувати гру. Це типовий підхід до розв'язання обчислювальних задач; підхід, застосування якого ми очікуємо від вас.

Файл `hangman.py` має ряд вже написаних функцій, які ви можете використовувати при написанні вашого розв'язку. Ви можете ігнорувати код у двох функціях на початку файлу, але ви маєте розуміти як використовувати кожну допоміжну функцію, прочитавши коментарі до них.

1A) Визначте, чи слово вже вгадано

Спершу, реалізуйте функцію `is_word_guessed`, яка приймає два параметри — рядок `secret_word` та список букв (рядків) `letters_guessed`. Ця функція повертає булеве значення: `True`, якщо `secret_word` було вгадано, та `False` інакше. Ця функція буде корисною при визначенні чи була конкретна партія успішно завершена та буде перевіркою завершення для будь-якого циклу, що перевіряє букви у загаданому слові.

Для цієї функції ви можете припустити, що всі букви у `secret_word` та `letters_guessed` вже подано у нижньому регістрі.

Приклад використання:

```
>>> secret_word = 'apple'
>>> letters_guessed = ['e', 'i', 'k', 'p', 'r', 's' ]
>>> print(is_word_guessed(secret_word, letters_guessed))
False
```

1B) Отримання користувацької здогадки

Тепер, реалізуйте функцію `get_guessed_word`, яка приймає два параметри: рядок `secret_word` та список букв `letters_guessed`. Ця функція повертає рядок, що складається з літер та підкреслень на основі того, які літери з `letters_guessed` містяться у `secret_word`. Вона не має сильно відрізнятися від `is_word_guessed`.

Ми будемо використовувати підкреслення та пробіл (_) для представлення невідомих літер. Ми могли б вибрати інші символи, але комбінація з підкреслення та пробілу є достатньо помітною, її легко вирізнити. Помітьте, що пробіл є дуже важливим, оскільки інакше було би дуже складно визначити, наприклад, складається ____ з чотирьох чи з трьох прочерків. Це називається *usability*: дуже важливо при програмуванні звертати увагу на те, чи легкою у користуванні є ваша програма. Якщо користувачу складно керувати програмою, він не буде нею користуватися. Ми заохочуємо вас думати про це при розробці ваших програм.

Підказка: при розробці ваших функцій, подумайте про те, яку інформацію ви хотіли б отримати у результаті, чи потрібне вам місце для збереження цієї інформації, коли ви проходите циклом по структурі даних, і як ви би хотіли додавати інформацію до вашого накопичуваного результату.

Приклад використання:

```
>>> secret_word = 'apple'
>>> letters_guessed = ['e', 'i', 'k', 'p', 'r', 's']
>>> print(get_guessed_word(secret_word, letters_guessed))
'_ pp_ e'
```

1C) Отримання усіх доступних букв

Тепер, реалізуйте функцію `get_available_letters`, яка приймає один параметр — список букв `letters_guessed`. Ця функція повертає рядок, який містить літери англійського алфавіту у нижньому регістрі — усі літери, що не входять у `letters_guessed`.

Ця функція має повертати літери у алфавітному порядку. Для цієї функції ви можете припустити, що всі літери у `letters_guessed` вже є у нижньому регістрі.

Підказка: Ви можете розглянути використання `string.ascii_lowercase`, що є рядком складеним із усіх латинських літер у нижньому регістрі:

```
>>> import string
>>> print(string.ascii_lowercase)
abcdefghijklmnopqrstuvwxyz
```

Приклад використання:

```
>>> letters_guessed = ['e', 'i', 'k', 'p', 'r', 's']
>>> print(get_available_letters(letters_guessed))
abcdfghjlmnoqtuvwxyz
```

Задача 3:
Шибениця. Частина 2: Гра

Тепер, коли у вас є якісь корисні функції, ви можете перейти до реалізації функції `hangman`, яка має приймати один параметр — `secret_word` — слово, що має вгадати користувач. Для початку, ви можете (і маєте!) встановити це загадане слово вручну, коли запускаєте функцію: це дозволить легко протестувати ваш код. Але у кінці ви захочете, аби це слово випадковим чином вибирав комп'ютер, перш ніж він запропонує вам чи іншому гравцю зіграти, запустивши цю функцію.

Виклик функції починає інтерактивну гру у Шибеницю між користувачем і комп'ютером. При розробці свого коду переконайтеся, що ви користуєтеся трьома допоміжними функціями — `is_word_guessed`, `get_guessed_word` та `get_available_letters` — які ви визначили у попередній частині.

Нижче наведено вимоги до гри, розділені на окремі категорії. Переконайтеся, що ваша реалізація відповідає усім цим вимогам!

Вимоги до гри

A. Архітектура гри:

1. Комп'ютер має вибирати слово випадковим чином зі списку доступних слів, що надані у файлі `words.txt`. Функції завантаження списку слів та вибору випадкового слова уже реалізовані для вас у файлі `hangman.py`.
2. Користувач починає з 6-ма спробами.
3. На початку гри дайте користувачу зрозуміти зі скількох букв складається слово комп'ютера і як багато спроб у нього є.
4. Комп'ютер слідкує за усіма невикористаними буквами і перед кожним кроком показує користувачу літери, що залишилися.

Приклад реалізації гри:

```
Loading word list from file...
55900 words loaded.
Welcome to the game Hangman!
I am thinking of a word that is 4 letters long.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
```

B. Взаємодія користувача і комп'ютера:

Гра має бути інтерактивною і відбуватися наступним чином:

1. Перед кожною спробою вгадати букву ви маєте показати користувачу:
    a. Нагадування про те, скільки спроб у нього/неї залишилося.
    b. Усі літери, що користувач ще не використовував.
2. Запитайте у користувача одне припущення. (Перегляньте вимоги до користувацького вводу нижче аби зрозуміти, що ви можете очікувати від користувача.)
3. Відразу після кожного вводу, користувачу необхідно повідомити чи є його літера у загаданому слові.
4. Після кожної спроби ви маєте також вивести користувачу загадане слово, замінивши невідгадані букви на підкреслення та пробіл (_).
5. У кінці спроби, виведіть кілька прочерків (-----) аби відділити окремі спроби одна від одної.

Приклад реалізації гри:
(Синім кольором позначено ввід користувача, аби відділити його від виводу комп'ютера.)

```
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Good guess: _ a_ _
-----------
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: b
Oops! That letter is not in my word: _ a_ _
```

C. Вимоги до вводу користувача:

1. Ви можете припустити, що користувач уводитиме тільки один символ за раз, але користувач може обрати будь-який символ, цифру чи букву. Ваш код має приймати як великі так і малі літери як валідні припущення!
2. Якщо користувач вводить щось окрім символів алфавіту (інші символи, цифри), повідомте йому, що можна вводити лише символи алфавіту. Оскільки користувачі можуть зробити це випадково, вони мають мати 3 попередження на початку гри. Кожного разу, коли вони вводять невалідне значення або літеру, яку вже вгадували, вони мають втрачати одне з цих попереджень. Якщо у користувача не залишається жодного попередження і введене значення невалідне, користувач втрачає одну спробу.

Підказка №1: Використовуйте функцію `input` для отримання вводу користувача.

1. Перевірте, що введене користувачем значення є символом алфавіту.

2. Якщо користувач ввів не велику чи малу літери алфавіту, відніміть одне попередження чи одну спробу.

Підказка №2: Ви можете використовувати рядкові функції `str.isalpha('your string')` та `str.lower('Your string')`! Якщо ви не знаєте, що ці функції роблять, ви можете спробувати ввести `help(str.isalpha)` чи `help(str.lower)` у інтерактивну консоль пайтону аби побачити їх документацію.

Підказка №3: Оскільки слова у списку містяться у нижньому регістрі, може бути простіше переводити користувацький ввід до нижнього регістру і маніпулювати лише ним у вашій грі.

Приклад реалізації гри:

```
You have 3 warnings left.
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: s
Oops! That letter is not in my word: _ a_ _
------------
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: $
Oops! That is not a valid letter. You have 2 warnings left: _ a_ _
```

D. Правила гри:

1. Гравець починає з 3-ма попередженнями.
2. Якщо користувач вводить щось окрім символу алфавіту (символи, цифри), повідомте йому, що він може вводити тільки символи алфавіту.
    a. Якщо у користувача залишилося 1 або більше попереджень, він втрачає одне з них. Повідомте йому, скільки попереджень у нього залишилося.
    b. Якщо у користувача не залишилося попереджень, він втрачає одну спробу.
3. Якщо користувач вводить літеру, яку він уже вводив, виведіть користувачу повідомлення, що цю літеру вже було спробувано.
4. Якщо користувач вводить літеру, яку раніше не вводив, і ця літера є у загаданому слові, користувач **не втрачає** спроб.
5. **Приголосні**: Якщо користувач вводить приголосну, яку раніше не вводив, і ця приголосна не міститься у загаданому слові, користувач втрачає **одну** спробу.
6. **Голосні**: Якщо користувач вводить голосну, яку раніше не вводив, і ця голосна не міститься у загаданому слові, він втрачає **дві** спроби. Голосними є *a*, *e*, *i*, *o* та *u*. *y* не рахується за голосну.

Приклад реалізації:

```
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: t
Good guess: ta_ t
------------
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: e
Oops! That letter is not in my word: ta_ t
------------
You have 3 guesses left.
Available letters: bcdfghijklmnopqrtuvwxyz
Please guess a letter: e
Oops! You've already guessed that letter. You now have 2 warnings:
ta_ t
```

E. Закінчення гри:

1. Гра має завершитися, коли користувач складе повне слово чи закінчаться усі його спроби.
2. Якщо користувач витратить усі свої спроби до того, як завершить слово, повідомте йому, що він чи вона програли та відобразьте загадане слово.
3. Якщо користувач виграє, виведіть поздоровлення та повідомте користувачу його рахунок.
4. Загальний рахунок є кількістю `guesses_remaning` у кінці гри помноженою на число унікальних літер у `secret_word`.

Total score = `guesses_remaining` * number of unique letters in `secret_word`

Приклад реалізації:

```
You have 3 guesses left.
Available letters: bcdfghijklnopquvwxyz
Please guess a letter: c
Good guess: tact
------------
Congratulations, you won!
Your total score for this game is: 9
```

Приклад реалізації:

```
You have 3 guesses left.
Available letters: bcdfghijklnopquvwxyz
Please guess a letter: n
Good guess: dolphin
------------
Congratulations, you won! Your total score for this game is: 21
```

F. Загальні підказки:

1. Розгляньте можливість написання додаткових допоміжних функцій, якщо вони вам необхідні.
2. Є чотири важливих частини інформації, які ви можете захотіти зберігати:
    a. `secret_word`: Загадане слово. Воно уже використовується як ім'я параметра у функції `hangman`.
    b. `letters_guessed`: Літери, які вже було використано. Якщо користувач вводить літеру, яку вже було використано, повідомте йому про це та застосуйте відповідний штраф.
    c. `guesses_remaining`: Кількість сброб, що залишилися у користувача. Зверніть увагу, що в нашому варіанті гри ціна помилки за невірно вгадану голосну відрізняється від ціни за невірно вгадану приголосну.
    d. `warnings_remaining`: Кількість попереджень, що залишилися у користувача. Зверніть увагу, що користувач втрачає попередження лише за введення символу чи літери, яку вже було вгадано.

G. Приклад гри:

Уважно перегляньте приклади виклику функції `hangman` подані нижче, в них наведено приклади інформації, яку ви маєте вивести після кожної спроби вгадати букву.

Примітка: Спробуйте зробити ваш вивід максимально наближеним до прикладів!

Вивід **виграшної** гри має виглядати так. (Синім кольором позначено користувацький ввід, аби відрізнити його від виводу комп'ютера.)

```
Loading word list from file...
55900 words loaded.
Welcome to the game Hangman!
I am thinking of a word that is 4 letters long.
You have 3 warnings left.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Good guess: _ a_ _
------------
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Oops! You've already guessed that letter. You have 2 warnings
left:  _ a_ _
------------
You have 6 guesses left.
```

```
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: s
Oops! That letter is not in my word.
Please guess a letter: _ a_ _
------------
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: $
Oops! That is not a valid letter. You have 1 warnings left: _ a_ _
------------
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: t
Good guess: ta_ t
------------
You have 5 guesses left.
Available letters: bcdefghijklmnopqrtuvwxyz
Please guess a letter: e
Oops! That letter is not in my word: ta_ t
------------
You have 3 guesses left.
Available letters: bcdfghijklmnopqrtuvwxyz
Please guess a letter: e
Oops! You've already guessed that letter. You have 0 warnings
left: ta_ t
------------
You have 3 guesses left.
Available letters: bcdfghijklmnopqrtuvwxyz
Please guess a letter: e
Oops! You've already guessed that letter. You have no warnings
left         so you lose one guess: ta_ t
------------
You have 2 guesses left.
Available letters: bcdfghijklnopquvwxyz
Please guess a letter: c
Good guess: tact
------------
Congratulations, you won! Your total score for this game is: 6
```

А вивід **програної** гри має виглядати так...

```
Loading word list from file...
55900 words loaded.
Welcome to the game Hangman!
I am thinking of a word that is 4 letters long
You have 3 warnings left.
-----------
```

```
You have 6 guesses left.
Available Letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Oops! That letter is not in my word: _ _ _ _
-----------
You have 4 guesses left.
Available Letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: b
Oops! That letter is not in my word: _ _ _ _
-----------
You have 3 guesses left.
Available Letters: cdefghijklmnopqrstuvwxyz
Please guess a letter: c
Oops! That letter is not in my word: _ _ _ _
-----------
You have 2 guesses left.
Available Letters: defghijklmnopqrstuvwxyz
Please guess a letter: 2
Oops! That is not a valid letter. You have 2 warnings left: _ _ _
_
-----------
You have 2 guesses left.
Available Letters: defghijklmnopqrstuvwxyz
Please guess a letter: d
Oops! That letter is not in my word: _ _ _ _
-----------
You have 1 guesses left.
Available Letters: efghijklmnopqrstuvwxyz
Please guess a letter: e
Good guess: e_ _ e
-----------
You have 1 guesses left.
Available Letters: fghijklmnopqrstuvwxyz
Please guess a letter: f
Oops! That letter is not in my word: e_ _  e
-----------
Sorry, you ran out of guesses. The word was else
```

Після того, як ви закінчили і протестували ваш код (в якому ви вручну задавали загадане слово, оскільки знання про нього дає вам змогу краще відлагоджувати ваш код), ви можете спробувати запустити його проти комп'ютера. Якщо ви догортаєте до кінця наданого файлу, ви побачите два закоментовані рядки після тексту `if __name__ == "__main__"`:

```
#secret_word = choose_word(worldlist)
#hangman(secret_word)
```

Ці рядки використовують функції, які ми реалізували (майже на початку `hangman.py`), які ви можете захотіти розібрати. Спробуйте розкоментувати ці рядки і перезапустити вашу програму. Це дасть вам змогу спробувати свої сили у грі проти комп'ютера, який використовує наші функції для завантаження великого масиву слів та вибору одного випадковим чином.

Задача 4:
Шибениця. Частина 3: Гра з підказками

Якщо ви спробували пограти у Шибеницю з комп'ютером, ви могли помітити, що виграти у нього може бути непросто, особливо коли він вибирає езотеричні слова (наприклад, "esoteric"!). Було б непогано, якби ви могли попросити у комп'ютера підказку, наприклад список усіх слів, які підходять під вгадані літери.

Наприклад, якщо загадане слово це "tact", а ви вгадали літеру "t", то ви знаєте, що рішення це "t_ _ t", де ви маєте вгадати дві літери, було б непогано знати, що множина підходящих слів (принаймні на основі того, що комп'ютер завантажив на початку) це:

tact tart taut teat tent test text thet tilt tint toot tort tout trot turf twit

Ми хочемо, аби ви написали версію Шибениці (ми називатимемо її `hangman_with_hints` і ми вже підготували початок її реалізації) з властивістю, що якщо ви пропонуєте спеціальний символ `*`, комп'ютер знайде всі слова, з тих, що він завантажив, які можуть відповідати поточному вгаданому слову, і надрукує кожне з них. Звичайно, ми не рекомендуємо пробувати це на першому кроці, оскільки це виведе усі 55900 слів, що були завантажені! Але якщо ви достатньо близько до відповіді і не маєте права на помилку, це може допомогти.

Аби виконати це, спочатку ми хочемо попросити вас закінчити дві допоміжні функції:

3A) Підбір за поточним вгаданим словом

`match_with_gaps` приймає два параметри: `my_word` і `other_word`. `my_word` є вгаданим словом, тобто може мати певну кількість підкреслень (_) в деяких позиціях (наприклад, "t_ _ t"). `other_word` є звичайним словом.

Ця функція має повертати `True`, якщо вгадані літери з `my_word` підходять до відповідних літер з `other_word`. Вона має повертати `False`, якщо два слова мають різну довжину чи вгадані літери з `my_word` не відповідають літерам з `other_word`.

Пам'ятайте, що якщо літеру вгадано, ваш код має відобразити всі позиції, в яких вона знаходиться у загаданому слові. Таким чином, прихована літера (_) не може бути однією з літер у слові, яку вже було вгадано.

Приклад використання:

```
>>> match_with_gaps("te_ t", "tact")
```

```
False
>>> match_with_gaps("a_ _ le", "banana")
False
>>> match_with_gaps("a_ _ le", "apple")
True
>>> match_with_gaps("a_ ple", "apple")
False
```

Підказка: Ви можете використовувати `strip()` аби позбутися пробілів у слові при порівнянні довжини.

3B) Виведення усіх можливих співпадінь

`show_possible_matches` приймає єдиний параметр: `my_word`, який є загаданим словом, тобто може містити певну кількість підкреслень (_) в деяких позиціях (нарпиклад, “t_ _ t“).

Ця функція має виводити усі слова з `wordlist` (зверніть увагу, що ми вже визначили його на початку, рядок 51), що підходять під `my_word`. Вона має виводити “No matches found”, якщо співпадінь немає.

Приклад використання:

```
>>> show_possible_matches("t_ _ t")
tact tart taut teat tent test text that tilt tint toot tort tout
trot tuft twit
>>> show_possible_matches("abbbb_ ")
No matches found
>>> show_possible_matches("a_ pl_ ")
ample amply
```

3C) Шибениця з підказками

Тепер ви маєте бути у змозі відтворити код, що написали для `hangman` як тіло `hangman_with_hints`, а потім зробити незначні зміни аби дозволити користувачу використовувати астериск (`*`) для того, щоб комп’ютер вивів усі слова, що підходять під вгадані букви.

Користувач не може програти, якщо його спробою був астериск.

Закоментуйте рядки коду, які ви використали, аби програти у оригінальну версію Шибениці:

```
secret_word = choose_word(wordlist)
hangman(secret_word)
```

І розкоментуйте ці рядки коду, що ми навели у кінці файлу, аби програти у вашу нову версію Шибениці з підказками:

```
#secret_word = choose_word(wordlist)
#hangman_with_hints(secret_word)
```

Приклад виводу:
Вивід від введеного астериску має виглядати як на прикладі нижче. Усі інші виводи такі ж самі як і у Шибениці з частини 2 вище.

```
Loading word list from file...
55900 words loaded.
Welcome to the game Hangman!
I am thinking of a word that is 5 letters long.
--------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter: a
Good guess: a_ _ _ _
--------
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter: l
Good guess: a_ _ l_
--------
You have 6 guesses left.
Available letters: bcdefghijkmnopqrstuvwxyz
Please guess a letter: *
Possible word matches are: addle adult agile aisle amble ample
amply amyls angle ankle apple apply aptly arils atilt
--------
You have 6 guesses left.
Available letters: bcdefghijkmnopqrstuvwxyz
Please guess a letter: e
Good guess: a_ _ le
--------
```

На цьому це завдання завершено!