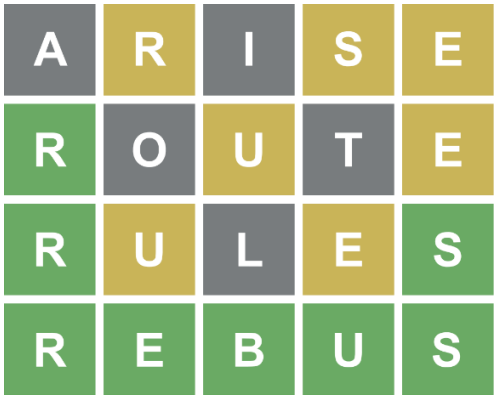
Wordle

Wordle je jednoduchá webová hra, ktorá sa stala populárnou v posledných mesiacoch. Cieľom hry je uhádnuť päťpísmenové slovo na šesť pokusov, pričom hra po každom pokuse prezradí, či náš tip obsahuje správne písmená, a či tie sú na správnom mieste. Na základe tejto spätnej väzby vieme eliminovať možné riešenia, a tak sa dopracovať ku skrytému slovu. Ak s hrou ešte nemáte skúsenosti, môžete si ju vyskúšať na https://www.powerlanguage.co.uk/wordle/.



Typický priebeh hry #196
(Zdroj: https://en.wikipedia.org/wiki/Wordle#/media/File:Wordle_196_example.svg)

Príklad hry môžete vidieť na obrázku vyššie. Skryté slovo je *rebus*. Prvý pokus hráča bol *arise*. Farby jednotlivých písmen reprezentuje informácie o správnosti písmen a ich pozícií. Napríklad pri slove *arise* sa písmená *r*, *s* a *e* nachádzajú aj v správnom riešení *rebus*, nie sú ale na správnej pozícii. Zvyšné sivé písmená sa v správnom riešení nenachádzajú. V ďalšom pokuse hráč zadal slovo *route*, v ktorom písmeno *r* je už na správnej pozícii a ďalej sa dozvedel aj to, že riešenie obsahuje aj písmeno *u*. Takýmto štýlom hráč pokračuje, až kým nenájde správne slovo, alebo nevyužije všetkých šesť pokusov.

Hru môžeme hrať aj s vyššou obťažnosťou, kde musíme brať do úvahy všetky pred tým získané vedomosti o skrytom slove. Napríklad v príklade vyššie by sme nemohli zadať slovo *route*, keďže už vieme, že písmeno *e* určite nebude na poslednej pozícii. V štandardnej verzii hry ale hráč môže zadať ľubovoľné slovo zo sady známych slov. Ak hráč zadá neplatné slovo, môže pokračovať bez toho, aby prišiel o pokus.

V tomto zadaní najprv doimplementujete textovú verziu hry a následne vytvoríte dvoch botov, ktorí budú schopní hrať hru s vysokou úspešnosťou.

Úloha 1 – 0,25 bodov

Základom hry bude súbor so sadou prijateľných slov. Príklad pre takýto súbor nájdete v priečinku `1b_samples`: každý riadok obsahuje jedno slovo a na konci súboru sa nachádza prázdny riadok.

V prvom kroku implementujete funkciu `load_words`, ktorá načíta zoznam slov akceptovaných hrou. Funkcia má jeden parameter – cestu k súboru, v ktorom sa nachádzajú podporované slová. Funkcia vracia zoznam slov, pričom jednotlivé prvky zoznamu sú reťazce s dĺžkou 5.

Úloha 2 – 0,25 bodov

K samotnej hre potrebujeme ďalšie dve jednoduché pomocné funkcie. Najprv implementujte funkciu `get_puzzle`, ktorá zo zoznamu slov ktorý dostane ako parameter `word_list` vyberie náhodné slovo, ktoré bude slúžiť ako hľadané riešenie. Funkcia má jednu návratovú hodnotu: slovo, teda `string` s dĺžkou päť. Pre náhodný výber použite štandardný modul `random`, ktorý už máte naimportovaný.

Dôležitou funkcionalitou je zabezpečiť, aby sme zistili, či hráč už uhádol správne riešenie. K tomu slúži funkcia `is_game_finished` s dvoma parametrami:

- `guess` – reťazec reprezentujúci tip hráča;
- `puzzle` – reťazec reprezentujúci správne riešenie.

Funkcia vracia hodnotu `True`, ak hráč slovo uhádol, v opačnom prípade vráti `False`.

Poznámka: Nezabudnite, že hráč môže svoj tip napísať aj veľkými písmenami, v takomto prípade ale musíte jeho tip vyhodnotiť takisto správne.

Úloha 3 – 1 bod

Poslednou funkciou k jednoduchej implementácii je `evaluate_guess`, ktorá vyhodnotí tip hráča a vráti mu informácie o správnosti písmen a ich pozícií. Funkcia má dva rovnaké parametre ako `is_game_finished`, teda tip hráča a správne riešenie. Návratová hodnote je ale trošku zložitejšia – bude to zoznam n-tíc (`list of tuples`), ktoré reprezentujú spätnú väzbu po vyhodnotení tipu a to tak, že pre každé písmeno sa vytvorí trojica hodnôt:

- `písmeno` – reťazec s dĺžkou 1
- informácia o tom, či sa dané písmeno nachádza v riešení – booleovská hodnota `True`/`False`
- informácia o tom, či dané písmeno je na správnej pozícii – booleovská hodnota `True`/`False`;

ak sa písmeno v riešení nenachádza, bude `False`, keďže písmeno je na nesprávnej pozícii.

V našom príklade teda tip *route* by sme vyhodnotili nasledovne:

```
[('r', True, True), ('o', False, False), ('u', True, False), ('t', False, False), ('e', True, False)]
```

Ak hráč uhádne správne slovo, všetky booleovské hodnoty budú `True`, ak zadá slovo, ktoré nemá žiadne písmeno spoločné s riešením, všetky hodnoty budú `False`.

Poznámka: Pri príprave vyhodnotenia musíte dodržiavať poradie písmen v slove.

Ak ste funkcie v prvých troch úlohách implementovali správne, môžete si zahrať hru Wordle pomocou funkcie `human_game`, ktorá pre vás bola pripravená na účely testovania. Funkcia má jeden parameter, cestu k súboru so sadou podporovaných slov. Vo funkcii môžete vidieť aj spôsob spolupráce jednotlivých funkcií. Momentálne sa na začiatku hry vypíše správne riešenie (`print(puzzle)`) aby ste vedeli otestovať správnosť vyhodnocovania vašich tipov. Ak hru chcete hrať na ostro, tento riadok môže zakomentovať alebo vymazať.

Úloha 4 – 1,5 bodov

Je známym faktom, že programátori sú leniví, a aj keď je Wordle zábavná hra a skvelý spôsob na prokrastináciu, skôr či neskôr im napadne implementovať inteligentného bota, ktorý by túto hru zahral za nich a omnoho efektívnejšie. V tejto úlohe vytvoríte jedného takéhoto bota.

Prvým krokom je implementovať funkciu `get_player_guess`, ktorá zo zoznamu slov `word_list` (formát rovnaký ako po načítaní funkciou `load_words`) ktorý dostane ako parameter, vyberie zoznam možných riešení a náhodný tip. Funkcia má teda dve návratové hodnoty:

- zoznam možných riešení – zoznam reťazcov, kde každý reťazec je päťpísmenové slovo; nepracujte priamo s parametrom `word_list`, ale vlastným zoznamom alebo kópiou `word_listu`.
- tip – reťazec, teda náhodné slovo zo zoznamu možných riešení.

Bot eliminuje možnosti na základe získaných vedomostí o správnom riešení, ktoré sú reprezentované v zozname `knowledge`, ktorý dostane ako parameter. Tento zoznam je inicializovaný na začiatku skriptu (neskôr sa pracuje s jeho kópiou) a je to zoznam trojíc, kde každá trojica reprezentuje informáciu o jednotlivých možných písmenách podobne ako pri funkcii `evaluate_guess`.

Trojica má nasledovnú štruktúru:

- písmeno – `string` s dĺžkou 1
- informácia o tom, či sa písmeno nachádza v slove – inicializovaná na `None`, neskôr nahradíte hodnotu booleovskými hodnotami `True`/`False`
- informácia o pozícii písmena v správnom riešení – inicializovaná na -1, neskôr nahradíte platným indexom 0 až 4. **Poznámka:** tento bot nebude rátať s možnosťou viacnásobného výskytu písmena v slove, teda tretia hodnota bude vždy iba jedno číslo.

Na základe týchto vedomostí by mala funkcia `get_player_guess` eliminovať možnosti nasledovne:

1. vymazať zo zoznamu možných riešení slová, ktoré neobsahujú písmená, o ktorých vie, že sa nachádzajú v správnom riešení na základe zoznamu `knowledge`
2. vymazať zo zoznamu možných riešení slová, ktoré obsahujú písmená, o ktorých vie, že sa nenachádzajú v správnom riešení na základe zoznamu `knowledge`
3. vymazať zo zoznamu možných riešení slová, ktoré nemajú správne písmeno na niektorej pozícii, o ktorej už vie, aké tam bude písmeno.

Pre jednoduchosť uvedieme aj jeden príklad. Po prvých dvoch pokusoch vyššie hráč už vie, že riešenie určite obsahuje písmená *r*, *s*, *e*, *u* a na prvej pozícii je písmeno *r*. Na základe jednotlivých pravidiel by teda eliminoval napríklad:

1. slovo *table*, ktoré neobsahuje potrebné písmená *r*, *s*, *u*
2. slovo *braid*, ktoré obsahuje písmeno *a*, o ktorom už hráč vie, že sa v riešení nenachádza
3. slovo *trout*, keďže hráč už vie, že na prvej pozícii musí byť písmeno *r*

Po eliminácii niektorých možných riešení, bot vyberie náhodné slovo zo zoznamu a vráti hodnoty podľa špecifikácie funkcie.

Samozrejme bot má šancu hru vyhrať iba, ak bude postupne aktualizovať reprezentáciu svojich vedomostí o hľadanom slove. K tomu slúži funkcia `process_result` s parametrom `result`, ktorý obsahuje spätnú väzbu od hry, ktorá je vygenerovaná funkciou `evaluate_guess` (formát sa nemení). Funkcia postupne spracuje informácie o všetkých písmenách z posledného tipu a to nasledovne:

1. aktualizuje informáciu o tom, či sa písmeno nachádza alebo nenachádza v správnom riešení
2. ak je písmeno aj na správnej pozícii, aktualizuje tento údaj v zozname `knowledge`.

Funkcia `process_result` nemá návratovú hodnotu, aktualizujte priamo zoznam `knowledge`.

Poznámka: Nezabudnite, že zoznam `knowledge` obsahuje n-tice, ktoré sú nemenné. Práve preto pri spracovaní spätnej väzby musíte vytvoriť novú n-ticu a uložiť ju na správne miesto v zozname `knowledge`. Poradie informácií o písmenách v zozname má pritom ostať rovnaké, t.j. najprv budete mať vedomosti o výskyte písmena *a*, písmena *b*, atď.

Po implementovaní týchto dvoch funkcií máte bota hotového, môžete jeho funkčnosť vyskúšať pomocou metódy `main()`. Funkcia je veľmi podobná funkcii `human_game` – vstup od hráča je ale nahradený vstupom od bota. Počas hry sa takisto vypíše zoznam možných riešení, ktorý bude stále kratší ako bot spracuje informácie, ktoré sa dozvie o hľadanom slove.

Úloha 5 – 1 bod

V poslednej časti zadania implementujete inteligentného bota, ktorý bude efektívnejšie využívať vedomosti o hľadanom slove. Tohto bota reprezentujú funkcie `get_smart_player_guess` a `smart_process_result`. Na reprezentáciu znalostí použije iný spôsob, konkrétne zoznam `SMART_PLAYER_KNOWLEDGE`, resp. jeho kópiu.

Zoznam `SMART_PLAYER_KNOWLEDGE` je veľmi podobný `PLAYER_KNOWLEDGE`, avšak namiesto n-tice sa používa zoznam na reprezentáciu znalostí o výskyte písmena v hľadanom slove. Dôvodom je to, aby sa jednoduchšie aktualizovali údaje jednotlivých členoch a aby sme poukázali na rozdielnosť medzi prácou s n-ticou a zoznamom. Zoznam popisujúci jedno písmeno má však stále tri hodnoty:

- písmeno – `string` s dĺžkou 1
- informácia o tom, či sa písmeno nachádza v slove – inicializovaná na `None`, neskôr nahradíte hodnotu booleovskými hodnotami `True`/`False`
- zoznam možných pozícií písmena v hľadanom slove – inicializovaný na [0, 1, 2, 3, 4], neskôr ho budete aktualizovať elimináciou možných pozícií písmena v slove. **Poznámka:** vďaka takejto reprezentácii dokáže inteligentný bot pracovať lepšie so slovami, v ktorých sa to isté písmeno vyskytuje viackrát.

Funkcia `get_smart_player_guess` má dva parametre: `word_list` (zoznam možných riešení) a `knowledge` (zoznam reprezentujúci znalosti o hľadanom slove – štruktúra rovnako ako

v `SMART_PLAYER_KNOWLEDGE`). Bude fungovať veľmi podobne, ako `get_player_guess`, eliminácia bude prebiehať rovnako v troch fázach:
1. vymazať zo zoznamu možných riešení slová, ktoré neobsahujú písmená, o ktorých vie, že sa nachádzajú v správnom riešení na základe `knowledge`
2. vymazať zo zoznamu možných riešení slová, ktoré obsahujú písmená, o ktorých vie, že sa nenachádzajú v správnom riešení na základe `knowledge`
3. vymazať zo zoznamu možných riešení slová, ktoré majú niektoré písmeno na nedovolenom mieste (zoznam dovolených miest nájdete v zozname `knowledge`).

Napríklad: vzhľadom na bod 3 by bot v našom príklade nepoužil ako druhý pokus slovo *route*, keďže po zadaní slova *arise* už vie, že písmeno *e* určite nie je na poslednej pozícii.

Po eliminácii niektorých možných riešení bot vyberie náhodné slovo zo zoznamu a vráti hodnoty podľa špecifikácie funkcie. Rovnako ako pri `get_player_guess`, aj tu použite vlastný zoznam na ukladanie možných riešení a neopravujte vstupný parameter `word_list`.

Poznámka: Pri implementácii funkcie `get_smart_player_guess` môžete použiť časti kódu z funkcie `get_player_guess` – kód nakopírujte alebo volajte vhodným spôsobom funkciu `get_player_guess`.

Takisto ako v prípade prvého bota, potrebujete implementovať aj metódu na spracovanie spätnej väzby. Na tento účel slúži funkcia `smart_process_result`. Má rovnaké parametre ako `process_result`, t.j. zoznam popisujúci spätnú väzbu o tipe (`result`) a zoznam so získanými znalosťami o hľadanom slove (`knowledge`), ktorý ale teraz má formát podľa `SMART_PLAYER_KNOWLEDGE`.

Funkcia výsledky spracuje podobne ako `process_result`, teda:
1. aktualizuje hodnotu výskytu písmena v hľadanom slove (na hodnoty `True`/`False`)
2. aktualizuje zoznam možných pozícií daného písmena v hľadanom slove – počas hry sa postupne mažú pozície. Nesprávnu pozíciu zo zoznamu vymažte iba v prípade, že sa písmeno nachádza v hľadanom slove, t.j. pre písmená, ktoré nie sú v hľadanom slove, zoznam ostane `[0, 1, 2, 3, 4]` až do konca behu programu.

Funkcia `smart_process_result` nemá návratovú hodnotu, aktualizujte priamo zoznam `knowledge`.

Po implementácii týchto dvoch funkcií môžete otestovať funkčnosť inteligentného bota pomocou metódy `smart_main()`. Metódy `human_game()`, `main()` a `smart_main()` sú pomocné, môžete ich pokojne meniť podľa potreby testovania.

Vaše riešenia môžete otestovať aj pomocou sady testov v súbore `sample_tests_1b.py`. Pri hodnotení vášho riešenia použijeme podobné testy, avšak ich bude viac.

Približná dĺžka riešenia: *cca. 100 riadkov kódu bez komentárov a už implementovaných pomocných funkcií*.