5. domácí úkol

- zadaný: 20. 11. 2023
- deadline: 4. 12. 2023, **23:55**
- maximální počet bodů: 40 (za funkcionalitu, řešení bez duplikovaného kódu, EduLint)

Pokyny

- Odevzdejte jediný soubor pojmenovaný prijmeni_du5.py (“prijmeni” nahraďte svým příjmením) s definicemi požadovaných funkcí. Odevzdávárnu najdete v ISu: Student / FI:IB113 / Odevzdávárny / Skupina 10 / DU5
- Šablona řešení je dostupná ve studijních materiálech.
- Jména odevzdávaných funkcí neměňte; ať zůstanou pojmenované stejně, jako jsou pojmenované v zadání.
- Případné dotazy pokládejte v diskuzním fóru této skupiny ve vláknu k tomuto úkolu.
- Příklad vypracujte samostatně. Nepovolená spolupráce bude potrestána dle pravidel předmětu.
- Funkce, které napíšete, můžete používat v rámci jiných funkcí. Můžete si implementovat i další pomocné funkce.

Zadání

Implementujte hru piškvorky ve variantě člověk proti počítači a počítač proti počítači.

Pravidla hry

Klasické piškvorky pravděpodobně znáte. Pravidla jsou následující:

- Hraje se na mřížce obdélníkových rozměrů.
- Na začátku je herní plán volný. Hráči se postupně střídají a nakreslí vždy do jednoho volného políčka mřížky svůj znak.
- Vyhrává hráč, jehož znaky vytvoří rovně nebo vertikálně nepřerušenou čtveřici.
- Pokud jsou zaplněna všechna pole a nikdo nevyhrál, nastává remíza.
- Souřadnice polí jsou indexovány od 0, první souřadnice odpovídá horizontálnímu a druhá vertikálnímu směru. Pole se souřadnicemi 0 0 je v levém horním rohu.
- Začíná hráč s kolečky.

Úkol

Implementujte jak hru člověka proti počítači, tak hru dvou počítačových strategií proti sobě. Pro plný bodový zisk implementujte alespoň dvě ze tří následujících počítačových strategií:

- *Náhodná strategie*: počítač hraje na náhodné volné pole.
- *Naivní strategie*: počítač hraje tak, aby vždy přiložil další symbol k pokud co možno nejdelší řadě svých symbolů.
- *Blokovací strategie*: pokud by protihráč mohl v příštím tahu vyhrát (tzn. má vhodně umístěny tři symboly a prostor na čtvrtý), blokuje ho. Jinak hraje libovolně.

Fantazii se meze nekladou, za komplexnější strategie vlastní tvorby nad rámec příkladů můžete dostat až pět bonusových bodů.

Konkrétně bude vaším úkolem implementovat následující funkce:

Strategie

Vaším úkolem bude vytvořit dvě ze tří funkcí `strategy_random(plan)`, `strategy_naive(plan)` a `strategy_block(plan)`, které vybírají nejlepší pole pro tah na základě výše zmíněných postupů. Jediným argumentem každé z těchto funkcí bude reprezentace herního plánu, funkce by měla vracet dvojici souřadnic pole vybraného pro tah.

Hra

Dále implementujte funkce `human_vs_pc(plan_width, plan_height, strategy)`a `pc_vs_pc(plan_width, plan_height, strategy1, strategy2)`. První z těchto funkcí zprostředkuje hru člověka proti počítači, druhá dvou počítačů proti sobě. Argumenty `plan_width` a `plan_height` označují šířku (vodorovně), respektive výšku (vertikálně) herního plánu, argumenty `strategy`, `strategy1` a `strategy2` jsou název funkce pro strategii, kterou bude používat počítač, respektive jeden z počítačů. (K předávání funkcí viz Rada 2 níže.)

Pokud hraje člověk, je vždy začínající hráč (a dle pravidel tedy hraje za kolečka). V případě hry dvou počítačů hraje za kolečka `strategy1` a za křížky `strategy2`.

V průběhu každého tahu (člověka i počítače) by měl být vypisován herní stav následující formou (zde dva tahy):

```
Turn of player o
move: 3 0
...o..x...
...o......
..........
..........

Turn of player x
move: 7 1
...o..x...
...o...x..
..........
..........
```

Na začátku tahu vypíše program, zda je na tahu hráč `o` nebo `x`, načež se buď zeptá na souřadnice tahu hráče nebo vypíše souřadnice tahu počítače. Pokud tah hráče není validní (nejde o čísla, která by byla validními indexy do hracího plánu), vyzývá hráče k zadání tahu

opakovaně, dokud nezadá validní tah. Následně vypíše podobu herního plánu po tahu. Pokud tahem hráč/počítač vyhrál, vypíše hra informaci, kdo vyhrál, a skončí.

Rady

Rada 1: Otestujte si, že program korektně funguje na obdélníkových plánech. Častá chyba u podobných úloh je, že testování proběhne pouze na čtvercových plánech.

Rada 2: V Pythonu můžete předávat jako parametr i jméno funkce. Můžeme tedy mít:
`def play_game(width, height, strategy1, strategy2): …`
kterou následně voláme například:
`play_game(6, 4, strategy_random, strategy_maxblock)`
kde `strategy_random`, `strategy_maxblock` jsou jména funkcí implementujících strategie.
Tyto aspekty Pythonu není nezbytné používat, úlohu můžete řešit i jinými způsoby (předáváním pomocné hodnoty pro výběr strategie).

Rada 3: Pro náhodnou strategii se mohou hodit další funkce z knihovny `random` (https://docs.python.org/3/library/random.html), konkrétně třeba `choice`.

Rada 4: Směr jako parametr
V tomto příkladu se vám může hodit kontrolovat/počítat nějaké vlastnosti hracího plánu v osmi (nebo možná jenom čtyřech?) směrech.

Směr je možné reprezentovat jako požadovanou změru na ose x a ose y. V naší “indexové” soustavě souřadnic, která má souřadnice (0, 0) v levém horním rohu a souřadnice rostou směrem doprava a dolů, pak směr doprava odpovídá změně (1, 0) (pro krok ve směru doprava se xová souřadnice musí zvýšit o jedna a yová souřadnice zůstane stejná). Směr nahoru odpovídá dvojici změn (0, -1), diagonální směr doleva dolů pak odpovídá změnám (-1, 1) atd.

Pokud mám souřadnice v proměnných `x` a `y` a posuny v proměnných `x_shift` a `y_shift`, pak nové souřadnice získám pomocí `new_x = x + x_shift` a `new_y = y + y_shift`.
Hodnoty `x_shift` a `y_shift` můžu předat funkci, kterou pak můžu napsat tak, že podle posunu předaného v parametrech dovede provést svou kontrolu v jakémkoli směru.

Občas se může hodit provést nějaký kód / zavolat funkci pro několik směrů po sobě. To je možné provést tak, že si vyrobím seznam obsahující směry (jako tuples – dvojice), přes který pak budu iterovat.

```
for x_shift, y_shift in [(1, 0), (1, 1), (0, 1), (-1, 1)]:
    # do stuff
```

Dvojice můžete samozřejmě použít i pro předání pozice/směru do funkce, pokud si s tím ale nevíte rady, stačí předat parametry zvlášť.

Pro zisk plného počtu bodů bude nutné implementovat řešení bez většího množství duplicitního kódu, k čemuž je úspěšné následování této rady nezbytné.

EduLint

V tomto úkolu část bodů dostanete za to, když váš kód čistě projde kontrolou nástrojem EduLint. EduLint je volně dostupný na edulint.com. Vaším cílem je, aby se poté, co do webu zkopírujete svůj kód a kliknete na Check, v bloku What to improve objevil pouze zelený text “No problems found in the code :)”. Abyste dostali pro vás zamýšlenou zpětnou vazbu, je nezbytné, aby kontrolovaný kód obsahoval řádek `# edulint: config=cs0` (dodaná šablona tento řádek obsahuje).

EduLint můžete nainstalovat i přímo jako plugin do Thonny. Pro to je třeba v Tools → Manage package vyhledat "edulint" a kliknout na Install, a poté v Tools → Manage plug-ins vyhledat "thonny-edulint" a kliknout na Install. Nakonec je třeba restartovat Thonny. Kontrolu pak provedete kliknutím na ikonku smetáku (Lint with EduLint).