

Domácí projekty 6

Dnešní projekty jsou součástí projektu 1D Piškvorky. Dělej je jeden po druhém.

o. Rozděl 1D Piškvorky na čtyři moduly:

- `ai.py`, kde bude funkce `tah_pocitace`,
- `piskvorky.py`, kde budou ostatní funkce,
- `hra.py`, kde bude import a volání hlavní funkce z `piskvorky.py` (a nic jiného),
- `test_piskvorky.py`, kde budou testy.

Připomínám: Až to bude fungovat, dej to do Gitu! A kdybys něco nedopatřením rozbila, `git diff HEAD` ukáže změny od poslední revize.

1. Napiš aspoň dva testy na každou funkci z Piškvorek, ke které testy napsat umíš.

Testy pravděpodobně neumíš napsat na funkci `input` a všechny funkce, které ji (byť nepřímo) volají. A taky asi neumíš otestovat účinek funkce `print`.

2. Zkus program přepsat tak, aby část, která není pokrytá automatickými testy, byla co nejmenší.

Místo volání `print` můžeš často vrátit řetězec a `print` zavolat až na návratovou hodnotu. Podobně `input` můžeš zavolat před zavoláním funkce, která pak může příkaz od uživatele brát jako argument.

3. Doplně funkci `tah_pocitace` tak, aby brala jako argument symbol, za který má počítač hrát – buď `'x'`, nebo `'o'`.

Dopiš testy.

Nezapomínáš na Git?

4. Ověř, že se funkce `tah_pocitace` se umí vyrovnat s jinou délkou hracího pole než 20.

„Ověření“ znamená napsání příslušného testu.

5. Ověř si, že se `tah_pocitace` chová rozumně když dostane plné hrací pole, nebo pole s délkou 0.

Rozumné chování v tomto případě znamená vyvolání rozumné výjimky.

Podle toho, jak jste se na sraze domluvili, pošli soubor `ai.py` (s funkcí `tah_pocitace`) e-mailem (např. organizátorům, koučovi, nebo vůbec). Posílej ho jako přílohu, nekopíruj ho do textu e-mailu.

Jestli procházíš-li kurz sama a můžeš programování konzultovat s někým zkušenějším, je tento úkol na takovou konzultaci ideální téma.

Dozvíš se, jak si tvá strategie stojí proti ostatním – a proti testům, které napíše organizátor. :) Strategii je možné i vylepšovat a poslat několik různých verzí. Neboj se poslat i první nástřel!

Stejně tak se neboj funkci poslat, i kdyby ještě nebyla úplně dodělaná.

Na některém dalším srazu bude turnaj o ceny!

6. Vytvoř hru sibenice podle následujícího popisu. Snaž se projekt rozdělit do funkcí a modulů s hezkými jmény, piš testy a dokumentační řetězce, funkční kousky dávej postupně do Gitu.

- Počítač náhodně zvolí slovo (zatím třeba ze tří možností). Pro jednoduchost používej malá písmena a nepoužívej slova, ve kterých se stejné písmeno opakuje dvakrát (třeba čokoláda).
- Nastav výchozí stav: řetězec s tolika podtržítka, kolik je ve vybraném slově písmen.
- Nastav počet neúspěšných pokusů na nulu.
- Každý tah:
 - Zeptej se hráče na písmeno.
 - Pokud je to písmeno ve vybraném slově, zaměň příslušná podtržítka za ono písmeno.
Bude se hodit řetězcová metoda `index` a funkce `zamen` z `piškvorek`.
 - Pokud dané písmeno *není* ve vybraném slově, započítej neúspěšný pokus.
 - Vypiš stav (slovo s případnými podtržítka).
 - Pokud už ve slově nejsou podtržítka, pograttuluj hráči a ukonči hru.
 - Podle počtu neúspěšných pokusů vypiš „obrázek“. Obrázky jsou ke stažení na <http://pyladies.cz/v1/s005-modules/obrazky.txt> a můžeš je dát do víceřádkových řetězců (s trojitými uvozovkami).
 - Pokud jsi právě vypsala poslední obrázek, hráč prohrál. Dej mu to najevo a ukonči program.

```
+---.  
|  |  
|  0  
| --|--  
|  /  
|  
~~~~~
```

p_t__n

7. Funguje-li ti předchozí hra, můžeš ji vylepšit.

- Zařiď, aby fungovala slova s více stejnými písmeny.
- Když hráč nezadá písmeno (zadá např. ABC nebo !), nepovažuj to za tah.
- Po skončení dej hráči možnost hru opakovat.