IB113 Úvod do programování a algoritmizace

Cvičení 11

Jaromír Plhák

Kódy z minula (fronta, seznam)

Úkol 4 - vyhodnocení

- Pep8
- Efektivita kódu vs. styl kódování
  - if, if, if, if … místo cyklu
  - Nepoužívání pomocných funkcí
    - Dlouhé funkce (omezte se třeba 30 řádky)
    - Přílišné zanoření (maximálně 4 úrovně)
  - Ne globálním proměnným
  - Hlavičky funkcí vás mají navést na správné řešení, neupravujte!
- Funkčnost
  - Detekce remízy
  - Detekce výhry v řádku, sloupci a diagonále
  - Nekorektní správa hry

Objektově orientované programování (OOP)

- OOP je paradigma (= myšlenkový vzor) programování postavené na 3 základných principech:
  - Abstraktní datový typ třída (class)
  - Dědičnost
  - Polymorfizmus
- Python podporuje OOP:
  - Budeme pracovat jen s jednoduchými třídami
  - Dědičnost a polymorfizmus nad rámec kurzu

Třída

- **Třída** = abstraktní datový typ (ADT) v OOP
  - = Šablona pro nové datové struktury
  - = „Výrobní vzor“ pro tzv. **objekty**

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
```

- Konvence: názvy tříd začínají velkým písmenem
- Podobnost se strukturou (struct) v jazycích C, Pascal

Objekt

- **Objekt** = instance třídy; má svůj stav a chování
  - Stav = uchovávaná data; aktuální hodnoty *atributů*
  - Chování = funkce; tzv. *metody* patřící objektu
- Objekt uchovává dohromady atributy a metody
  - Zapouzdření (encapsulation)
- Tyto atributy a metody předpisuje třída

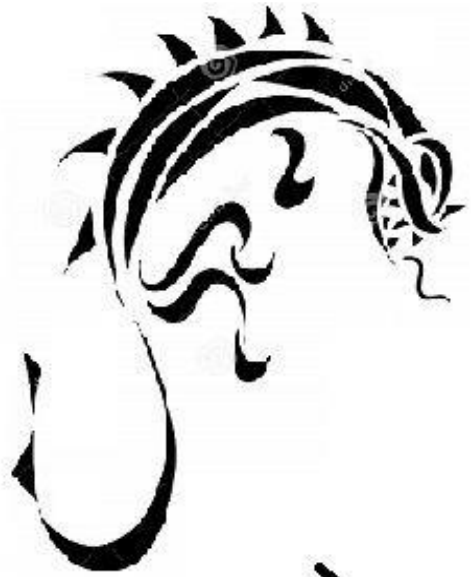
```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
hero = Creature ('Augur', 31, 4130)
monster = Creature ('Goblin', 5, 220)
```

Objekt jako instance třídy

hero = Creature ('Augur ', 31, 4130)

monster = Creature ('Goblin ', 5, 220)

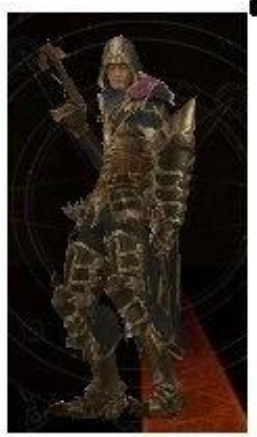
- Třídu Creature budeme používat jako nový datový typ (podobně jako int, string, bool, ...)
- Vytvořili jsme dva objekty, hero a monster, typu Creature
- Oba jsou ze stejné šablony, liší se jen hodnotami atributů



Trieda Creature

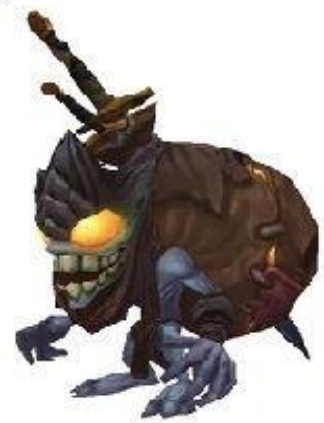
Atribúty:
name
level
life

Metódy:
__init__()



Objekt

Hodnoty atribútov:
"Augur"
31
4130



Objekt

Hodnoty atribútov:
"Goblin"
5
220

http://www.dreamstime.com/stock-images-abstract-silhouette-lizard-image16690644
http://diablo.wikia.com/wiki/Treasure_Goblin

Třída - kód

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
```

- class Creature:
  - Příkaz pro definici nové třídy, tak jako def foo(): definuje novou funkci
- def __init__(self, n, l, h):
  - Definice metody s názvem __init__ a danými parametry
    - Speciální metoda, tzv. inicializátor

Konstruktor a inicializátor

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
```

- Volání názvu třídy jako metody, t.j. Creature() zavolá **konstruktor -** metodu tvořící nový prázdny objekt
- Inicializátor nastaví počáteční hodnoty jeho atributů

```
hero = Creature ('Augur ', 31, 4130)
monster = Creature ('Goblin ', 5, 220)
```

- Vždy obsahuje alespoň jeden parametr, self (obvykle první v pořadí), který odkazuje na vytvářený objekt

Konstruktor a inicializátor - volání

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
hero = Creature ('Augur ', 31, 4130)
```

- Uvnitř metody __init__():

```
        self.name = 'Augur ,
        self.level = 31
        self.life = 4130
```

- self odkazuje na vytvářený objekt; používá se jen „skrytě“ uvnitř inicializátora, nemusíme ho psát při volaní

Atribut

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
```

- Datová složka (proměnná) patřící objektu
- self.name odlišuje atribut objektu třídy Creature od lokální proměnné n funkce
- Přístup k atributu: object.attribute

```
>>> hero = Creature ('Augur ', 31, 4130)
>>> print (hero.name)
Augur
```

Metoda

- Funkce, která ovlivní samotný objekt, kterému patří

```
class Creature :
        def __init__ (self, n, l, h):
                self.name = n
                self.level = l
                self.life = h
        def take_damage (self, dmg ):
                self.life -= dmg
                if self.life <= 0:
                        print (self.name, 'is dead!')
>>> hero = Creature ('Augur ', 31, 4130)
>>> hero.take_damage (10)
>>> print(hero.life)
4120
```

Úkoly

- 10.2.1. Kruh
  - Napište třídu pro reprezentaci kruhu s daným středem a poloměrem. Implementujte metody pro výpočet obvodu, obsahu a vrácení informačního řetězce (__str__).
- 10.2.3. Příšera
  - Doplňte následující kostru třídy a vyzkoušejte její funkčnost. Poté přidejte příšeře další metody, např. is_defeated, která vrací True, pokud už příšera nemá žádné životy.

```
from math import pi

class Circle(object):
        def __init__(self, center, radius):
                self._center = center
                self._radius = radius

        def get_perimeter(self):
                return 2 * pi * self._radius

        def get_area(self):
                return pi * (self._radius ** 2)

        def __str__(self):
                 return 'Circle at {c} with radius {r}'.format(c=self._center, r=self._radius)
```

Zadání domácí úlohy 5 – Objekty

- V ISu v odevzdávárně
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006396/72006443/hw05_zadani.py (skupina 1)
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006400/72006457/hw05_zadani.py (skupina 2)
- Veškeré informace v souboru
- Odevzdejte pouze tento soubor do odevzdávárny
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006396/72006433/ (skupina 1)
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006400/72006454/ (skupina 2)
- Do **neděle** 10. 12. 2017, 23:59