Poznámka k magickým funkcím

které byly minule v podobě konstruktoru...

- Magické funkce mají kolem názvu z obou stran dvě podtržítka,
- příklady: `__init__`, `__eq__`, `__add__`
- Volají se implicitně, reprezentují konstruktor, přetěžování operatorů, implicitní konverze,...
- vizte dokumentaci: `https://docs.python.org/3/reference/datamodel.html`
- sekce 3.3

Aplikace objektového programování

v podobě datové struktury

- Dynamické datové struktury jsou takové, do kterých lze prvky bez omezení přidávat a odebírat.
- Příklad: Spojové seznamy. Každý prvek nese hodnotu (hodnoty) a ukazatel na následníka.
- Ačkoliv existují předpřipravená řešení, my si ukážeme, jak spojové seznamy vytvořit sami.
- Využijeme objektové programování.

Aplikace objektového programování

v podobě datové struktury

- Data budou v objektu v atributu k tomu určeném (hod, data,....),
- jeden kontejner bude ukazovat na následníka (next, dalsi,...),
- známe (například z fronty v menze).

Příklad

spojového seznamu

```
class vagon:
    def __init__(self, hod, next=0):
        self.hod=hod
        self.next=next
```

- Vlastnosti datové struktury?
- Operace (s ní)?
- Parametrizace?
- Modifikace

Datové struktury

nám umožňují ukládat data

Typické operace:

- Přidej (`insert, append, add,...`)
- uber (`delete, erase, remove,...`)
- najdi (`member, find,...`)
- zvláštní (`extract_min, merge, find_nearest,...`)
- pozor na okrajové případy!

Příklad

spojového seznamu

```
class vagon:
    def __init__(self, hod, next=0):
        self.hod=hod
        self.next=next
class spojak:
    def __init__(self):
        self.zacatek=None
        self.konec=None
    def pridejZ(self,co):
        self.zacatek=vagon(co,self.zacatek)
        if self.konec==None: self.konec=self.zacatek
```

Příklad

pokračování

```
class spojak:
    ...
    def pridejK(self,co):
        if self.zacatek!=None:
            self.konec.next=vagon(co,None)
            self.konec=self.konec.next
        else: self.pridejZ(co)
    def uber(self):
        if self.zacatek==None:
            return None
        vratit=self.zacatek.hod
        self.zacatek=self.zacatek.next
        if self.zacatek==None: self.konec=None
        return vratit
```

Aplikace spojových seznamů

anebo k čemu se tato hračka hodí...

- Fronta,
- zásobník,
- dequeue (STL),
- ...

Složitosti operací (při různých variantách).

Objektová dekompozice

je důležitá

- Funkce pracující s objekty obsahujícími data můžeme napsat do samostatného souboru,
- anebo do samostatné třídy.
- Výhody?
- s1=seznam() s2=seznam() s3=seznam()
- ... a máme tři různé seznamy.

Typologie spojových seznamů

alias zvěřinec a jeho využití

- Jak odebírat z konce? Těžko. $\Rightarrow$
- Obousměrný spojový seznam (prvky ukazují i na předchůdce),
- cyklický (z konce ukazuje next na začátek),
- s hlavou (v prvním prvku není hodnota $\Rightarrow$ nenarazíme tam na None),
- s ocasem ("hlava"na konci) – lze využít jako zarážku.
- Hledání se zarážkou: Na konec dáme, co hledáme (pak se nemusíme bát, že to nenajdeme).
- Uspořádaný.