Warteschlange mit Tupel

Realisierung einer
Warteschlange
mit Tupeln

Warteschlange mit Tupel

- Problem: Tupel sind statisch
- Sie lassen sich aber neu definieren
- Es muss tief geschachtelt werden
  - **`('Hund', ('Katze', ('Vogel', None)))`**
  - Rekursion einsetzen
  - Verzicht auf Verwaltung des Endes

Warteschlange mit Tupel

Anstellen

- Fallunterscheidung
  - Warteschlange ist leer, also kopf==None
    → neues Tupel ('Hund', None) bildet den Kopf
  - Warteschlange ist nicht leer
    → am Ende neues Tupel rekursiv einfügen
    → Hilfsfunktion zum Anhängen

Warteschlange mit Tupel

```
def Anhaengen(self, aktuell, objekt):
    if aktuell[1]==None:
        return (aktuell[0], (objekt, None))
    else:
        return (aktuell[0],
self.Anhaengen(aktuell[1], objekt))
```

kein Umbruch

Selbstaufruf
→ Rekursion

Warteschlange mit Tupel

- Aufrufen ist einfacher

```
def Aufrufen (self) :
    '''setzt den Kopf der Warteschlange weiter
    und gibt den Inhalt des bisherigen Kopfes
    zurück'''
    if self.__kopf == None:
        return None
    kopfObjekt = self.__kopf[0]
    self.__kopf = self.__kopf[1]
    return kopfObjekt
```

Reihenfolge beachten!

Warteschlange mit Tupel

- "Rekursiv nachklappernd" wird das tief geschachtelte Tupel zusammengefügt:
- 'Vogel' an ('Hund', ('Katze', None))
  - 'Vogel' an ('Katze', None)
    - → ('Katze', ('Vogel', None))
  - → ('Hund', ('Katze', ('Vogel', None)))

Warteschlange mit Tupel

- Iterator

```
def __iter__(self):
    '''initialisiert den Iterator'''
    self.aktuell=self.__kopf
    return self

def __next__(self):
    '''definiert den Iterationsschritt'''
    if self.aktuell==None:
        raise StopIteration
    else:
        wert=self.aktuell[0]
        self.aktuell=self.aktuell[1]
        return wert
```

Warteschlange mit Tupel

- Anwenden bei GibLaenge()

```
def GibLaenge (self) :
    '''gibt die aktuelle Länge der
    Warteschlange zurück'''
    laenge = 0
    # Lösung mit Iterator
    for element in self:
        laenge += 1
    return laenge
```

Warteschlange mit Tupel

- Anwenden bei Letzter()

```
def Letzter(self):
    '''gibt den Inhalt des letzten Elements
    der Warteschlange zurück'''
    ergebnis=None
    for element in self:
        if element!=None:
            ergebnis=element
    return ergebnis
```