Kontejnery

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

Kontejnery

Hešovatelné objekty

- Objekty
  - mutabilní – proměnné,
  - imutabilní – neměnné,
  - hešovatlené – opravdu neměnné objekty.

- Hešovatelné objekty
  - je-li objekt hešovatelný, je považován opravdu za neměnný,
  - objekt je hešovatelný, má-li definovanou heš-hodnotu, která se během života objektu neměnní,
  - heš-hodnota je celé číslo, které získáme zavoláním funkce:

```
hash()
```

Druhy kontejnerů

- **Posloupnosti (sequence)**
  - kontejnery, u kterých je definováno pořadí prvků,
  - na prvky se můžeme odvolávat indexem
    - Řetězce – neměnné posloupnosti znaků (instance třídy str).
    - Seznamy – proměnné posloupnosti (instance třídy list).
    - N-tice – neměnné posloupnosti obecných objektů (instance třídy tuple).
- **Množiny**
  - obecné proměnné kontejnery, v nichž může být každá hodnota jen jednou,
  - instance třídy set.
- **Zmrazené množiny**
  - neměnné množiny,
  - instance třídy frozenset.
- **Slovníky**
  - množiny uspořádaných dvojic (klíč, hodnota),
  - instance třídy dict.

Vytvoření kontejneru

- Máme tři základní možnosti
  - prostřednictvím literálů,
  - prostřednictvím konstruktorů,
  - prostřednictvím generované notace.

Vytváření kontejnerů pomocí literálů

- Všechny kontejnery, kromě zmrazených množin mají definovány svoje literály:
  - „“ – řetězce,
  - () – n-tice,
  - [] – seznamy,
  - {} – množiny
  - {} – slovníky, vkládaná dvojice se zapíše tak, že se vždy nedříve napíše klíč, za ním dvojtečka za ní hodnota.

- Příklad:

```
seznam=[1,5,6]
```

Vytváření kontejnerů pomocí konstruktorů

- konstruktory požadují jako argument nějaký zdroj hodnot (iterovatelný objekt),
- hodnoty z tohoto zdroje se pak stanou prvky vytvářeného kontejneru,
- jedinou výjimkou je řetězec, kde lze jako argument zadat libovolný objekt, který je převeden na řetězec.

- Příklad:

```
seznam=list(range(10))
```

Vytváření kontejnerů pomocí generátorové notace

- požadované hodnoty jsou vygenerovány,
- vygenerované hodnoty potom můžeme:
  - uzavřít do iterátorů a vytvořit tak příslušný kontejner,
  - použít jako argument konstruktoru.
- Syntaxe generátoru:

```
výraz for proměnná in zdroj if podmínka
```

- Příklad:

```
cisla=range(10)
seznam=[n*n for n in cisla]
```

Příklad

- Vytvoříme seznam všech násobků pěti od 1 do 100:

```
seznam=[x for x in range(1,101) if x%5==0]
print(seznam)
```

- Vytvoříme seznam s malou násobilkou:

```
seznam=[[x*y for x in range(1,11)] for y in range(1,11)]
print(seznam)
```

- Vytvoříme seznam všech možných PINů:

```
seznam=[str(a)+str(b)+str(c)+str(d) for a in range(10) for b in range(10)
for c in range(10) for d in range(10)]
print(seznam)
```

Řetězce

- jsou neměnné posloupnosti znaků
- Vytvoření:
  - literály:

```
retezec="Python"
retezec='Python'
```

  - konstruktor:

```
retezec=str(123)
```

  - generovaná notace:

```
prevedeny=''.join(n.upper() if n.islower() else n.lower() for n in retezec)
```

    - nelze použít:

```
prevedeny="n.upper() if n.islower() else n.lower() for n in retezec"
prevedeny=str(n.upper() if n.islower() else n.lower() for n in retezec)
```

Seznamy

- jsou proměnné posloupnosti (instance třídy list)
- Vytvoření:
  - literály:

```
seznam=[1,2,3,4]
```

  - konstruktor:

```
seznam=list(range(1,5))
```

  - generovaná notace:

```
seznam=[1,2,3,4]
mocniny=[n**2 for n in seznam]
mocniny=list(n**2 for n in seznam)
```

n-tice

- jsou proměnné posloupnosti (instance třídy list)
- Vytvoření:
  - literály:

```
ntice=1,2
ntice=(1,2)
```

  - konstruktor:

```
ntice=tuple(range(1,3))
```

  - generovaná notace:

```
seznam=[1,2,3,4]
ntice=(n**2 for n in seznam) - nelze
ntice=tuple(n**2 for n in seznam)
```

Množina

Množina

- obecné proměnné kontejnery, v nichž může být každá hodnota jen jednou,
- v množině mohou být pouze hešovatelné objekty,
- instance třídy set.
- Vytvoření:
  - literály:

```
mnozina={1,2,3,4}
```

  - konstruktor:

```
mnozina=set(range(1,5))
```

  - generovaná notace:

```
seznam=[1,1,2,2,3,4]
mnozina={n**2 for n in seznam}
mnozina=set(n**2 for n in seznam)
```

Metody

- přidávání prvků do množiny:

```
.add(prvek)
.update(prvky)
```

- odstranění prvku z množiny:

```
.discard(prvek)
.remove(prvek) – pokud není prvek v množině, vyvolá metoda chybu
.clear() – odstraní z množiny všechny prvky
```

Příklad – počet znaků v řetězci

- Chceme zjistit počet jednotlivých znaků v řetězci.

```
retezec="v množině mohou být pouze hešovatelné objekty"
znaky=set(retezec)
for i in znaky:
    print(f"V řetězci je {retezec.count(i)}x znak {i}.")
```

Množinové operace

- operátory

```
a = {1,2,3,4,5}
b = {4,5,6,7,8}

print("Sjednocení", a | b)
print("Průnik", a & b)
print("Rozdíl", a - b)
print("Symetrický rozdíl", a ^ b)
```

- metody

```
print("Sjednocení", a.union(b))
print("Průnik", a.intersection(b))
print("Rozdíl", a.difference(b))
print("Symetrický rozdíl", a.symmetric_difference(b))
```

Příklad

- Mějme množiny účastníků kurzů:

```
prvni = {"Adam","Eduard","Pavel","Jana","Lucie","Lukáš"}
druhy = {"Adam","Pavla","Petr","Kryštof","Jana"}
treti={"Adam","Lucie","Kryštof"}
```

- Chceme vytvořit seznam poplatků za kurzy. Přitom platí, že kdo se zúčastnil všech tří kurzů, platí 150 Kč, kdo dvou 125 Kč a kdo jednoho 100 Kč.

- Nejdříve si vytvoříme seznam všech účastníků:

```
seznam=[[x,0] for x in prvni|druhy|treti]
```

- Vytvoříme funkci pro nastavení poplatku vybraným účastníkům:

```
def nastav_poplatek(komu,cena):
    for i in komu:
        for s in seznam:
            if s[0]==i:
                s[1]=cena
```

Příklad

- Vytvoříme funkci pro nastavení poplatků všem účastníkům:

```
def poplatek(p,d,t):
    s=p&d&t
    nastav_poplatek(s,150)
    s=((p&d)|(p&t)|(d&t))-(p&d&t)
    nastav_poplatek(s,125)
    s=((prvni|druhy|treti))-((p&d)|(p&t)|(d&t))
    nastav_poplatek(s,100)
```

- Funkci zavoláme:

```
poplatek(prvni,druhy,treti)
print(seznam)
```

Slovníky

Slovníky

- množiny uspořádaných dvojic (klíč, hodnota),
- instance třídy dict.

- Vytvoření:
  - literály:

```
slovnik = {"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}
```

  - konstruktor:

```
slovnik=dict([["Jablko", "Apple"], ["Knoflík", "Button"], ["Myš",
"Mouse"]])
slovnik=dict(Jablko="Apple", Knoflík="Button", Myš="Mouse")
```

  - generovaná notace:

```
ascii={x:chr(x) for x in range(128)}
```

Vlatnosti

- klíč
  - klíčem může být jakýkoliv hašovatelný typ
- hodnota
  - hodnotou může být jakýkoliv typ
- Příklad:

```
student={"jmeno":"Karel","kredity":20,"adresa":{"ulice":"Palackého 26","město":"Olomouc"},"předměty":["ZPPC1","DATAB"]}
```

- základní operace:
  - hodnotu získáme pomocí klíče: slovnik[klic]

```
student["jmeno"]="Václav"
```

  - prvek do slovníku přidáme odkazem na nový klíč:

```
slovnik["papír"]="paper"
```

  - prvek smažeme pomocí del:

```
del slovnik["papír"]
```

Metody

- `.keys()` - vrátí seznam klíčů,
- `.values()` - vrátí seznam hodnot,
- `.items()` - vrátí seznam prvků,

- `.update(slovnik)` - přidá prvky slovnik do slovníku

Procházení slovníků

- Pro procházení slovníků je nejlepší použít cyklus for.
- Vytiskneme hodnoty:

```
slovnik = {"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}

for slovo in slovnik:
    print (slovnik[slovo])
```

- Vytiskneme klíče:

```
slovnik = {"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}

for slovo in slovnik:
    print (slovo)
```

- Pracujeme s klíčem i hodnotou:

```
slovnik = {"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}

for k,v in slovnik.items():
    print (k,v)
```

Příklad

- Vytvoříme program pro kódování a dekódování textu. Kódovací klíč bude uložen ve slovníku:

```
kodovaciKlic = {"A": "f","B": "c","C": "z","D": "r","E": "u",
"F": "k","G": "j","H": "a","I": "y","J": "w","K": "b","L": "q",
"M": "d","N": "e","O": "x","P": "m","Q": "p","R": "v","S": "g",
"T": "t","U": "s","V": "h","W": "o","X": "n","Y": "i","Z": "l"}
```

- Funkce pro zakódování textu:

```
def zakoduj(text):
    kodovany=""
    for z in text:
        kodovany+=kodovaciKlic[z]
    return kodovany
```

- Funkci lze zjednodušit pomocí generované notace:

```
def zakoduj(text):
    return "".join([kodovaciKlic[x] for x in text])
```

Příklad

- Funkce pro dekódování:

```
def dekoduj(text):
    dekodovany=""
    for z in text:
        for k,v in kodovaciKlic.items():
            if v==z:
                dekodovany+=k
                break
    return dekodovany
```

- I tuto funkci lze zjednodušit:

```
def dekoduj(text):
    dekodovaciKlic={kodovaciKlic[x]:x for x in kodovaciKlic}
    return "".join([dekodovaciKlic[x] for x in text])
```

Příklad

- Napíšeme program pro sledování zapsaných předmětů pro jednotlivé studenty.
- Pro seznam studentů použijeme seznam, který bude obsahovat slovník s klíči jmeno a predmety.

- Napíšeme funkci pro vytvoření osoby:

```
def pridej_osobu(jmeno):
    seznam.append({"jmeno":jmeno,"predmety":set()})
```

- Funkce pro zápis předmětu:

```
def zapis_predmet(predmet,jmeno):
    for i in seznam:
        if i["jmeno"]==jmeno:
            i["predmety"].add(predmet)
```

Příklad

- Funkce pro odepsání předmětu:

```
def odepis_predmet(predmet,jmeno):
    for i in seznam:
        if i["jmeno"]==jmeno:
            i["predmety"].discard(predmet)
```

- Funkce pro vytvoření seznamu studentů, kteří mají zapsaný daný předmět:

```
def seznam_zapsanych(predmet):
    s=set()
    for i in seznam:
        if predmet in i["predmety"]:
            s.add(i["jmeno"])
    return s
```

Úkol

- Napište program pro sledování příjmů rodinných příjmů a výdajů.
- Implementujte tyto funkce:
  - `pridej_polozku(den,mesic,rok,castka,kategorie)` – která přidá do seznamu položek danou položku,
  - `prehled(mesic,rok)` – která vypíše celkový součet výdajů za jednotlivé kategorie pro daný měsíc a rok.
- Například:

```
pridej_polozku(15,10,2022,150,"jídlo")
pridej_polozku(16,11,2022,250,"jídlo")
pridej_polozku(17,11,2022,300,"bydlení")
pridej_polozku(18,11,2022,500,"jídlo")
pridej_polozku(19,11,2022,150,"domáctnost")

print(prehled(11,2022))
```

- Vytiskne

```
{'jídlo': 750, 'domácnost': 150, 'bydlení': 300}
```