Datové struktury

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZADS - Základní algoritmy a datové struktury

Konzultace

- v pracovně 5.071
- každou středu (14.00 – 15.00)
- jindy po vzájemné domluvě,
- email: jiri.zacpal@upol.cz,
- MS Teams

Kontejnery

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)
```

n-tice

- jsou proměnné posloupnosti (instance třídy tuple)
- 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)
```

Seznamy

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)
```

Vytvoření seznamu

- Syntaxe:

```
[první_položka_seznamu, druhá_položka_seznamu, ..., poslední_položka_seznamu]
```

- Položky seznamu jsou reference na (libovolné) objekty, které reprezentují tyto položky.
- Seznam je mutabilní.
- Příklady:

  L = [1, 2, 3, 4]

  L = ["s", "p", "a", "m"]

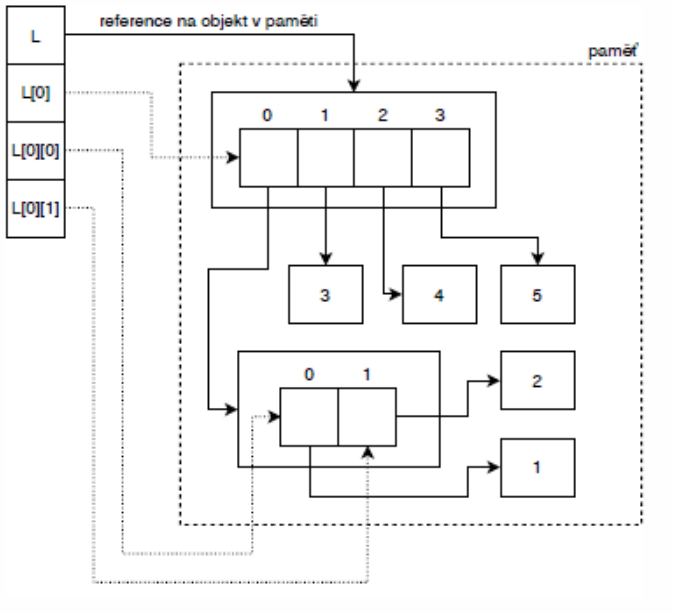
  L = ["spam", "ham"]

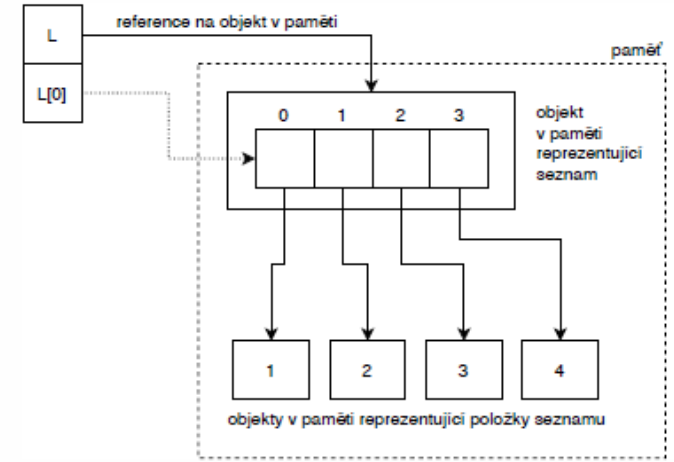
  L = ["spam", 2, 3, 4]

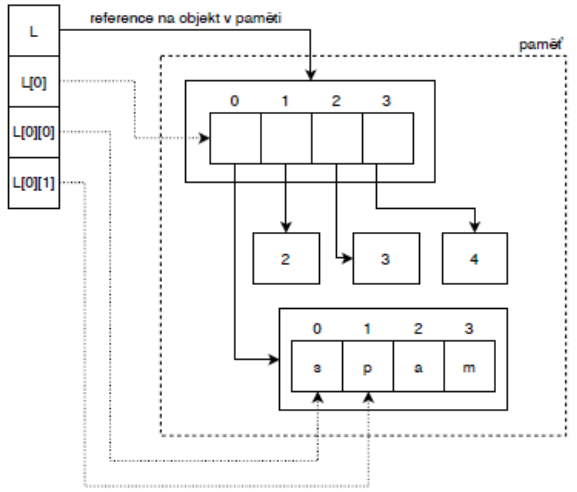
Práce se seznamy

- Pro práci se seznamy lze použít stejné nástroje jako pro práci s číselnými sekvencemi.
- Příklad:

```
L = [1, 2, 3, 4]
print(L[1]**L[2]) # vypíše: 8
L = ["spam", 2, 3, 4]
L = [[1, 2], 3, 4, 5]
```







Mutabilita seznamu

- Seznamy jsou na rozdíl od číselných sekvencí a řetězců mutabilní.
- Jejich jednotlivé položky lze měnit.
- Příklad:

```
L = [1, 2, 3, 4, 5]
L[0] = 42
print(L) # vypíše: [42, 2, 3, 4, 5]
```

- Mutabilita přináší drobnou komplikaci. Ke změně objektu může dojít skrze různé reference.
- Příklad:

```
L = [1, 2, 3, 4, 5]
M = L # M nyní obsahuje referenci na L
M[0] = 42 # změní referenci v L
print(L) # vypíše: [42, 2, 3, 4, 5]
```

- V následujícím kódu výše uvedené nenastává:

```
L = [1, 2, 3]
M = [4, 5]
N = [L, M]
M = 42
print(N) # vypíše: [[1, 2, 3], [4, 5]]
```

Metody pro práci se seznamy

- .append(objekt) přidá objekt na konec seznamu,
- .extend(sekvence) přidá všechny položky sekvence na konec seznamu,
- .remove(hodnota) odstraní ze seznamu první výskyt objektu s hodnotou hodnota,
- .pop() odstraní poslední prvek seznamu,
- .pop(index) odstraní prvek seznamu na daném indexu,
- .clear() odstraní všechny položky seznamu,
- .copy() vrací kopii seznamu (nový objekt),
- .reverse() vrací seznam s prvky v obráceném pořadí.
- .count(hodnota) vrací počet prvků seznamu s hodnotou hodnota

Příklad

```
s=[]
for i in range(0,10):
    s.append(int(input("Zadej teplotu:")))

soucet=0
for i in s:
    soucet+=i

print(f"Průměrná teplota je {soucet/len(s):.2f}")

max=s[0]
for i in range(1,len(s)):
    if s[i]>max:
        max=s[i]

print(f"Nejvyšší teplota je {max}")
```

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 správu knihovny. O každé knize ukládejte tyto informace: název, autora, informaci o tom, zda je vypůjčena nebo ne.
- Implementujte tyto funkce:
  - pridej_knihu(nazev,autor_jmeno,autor_prijmeni) – která přidá do seznamu knih danou knihu,
  - vypujceni(nazev) – která nastaví stav vypůjčení na True u dané knihy,
  - vraceni(nazev) – která nastaví stav vypůjčení na False u dané knihy,
  - vypujcene_knihy() – která vrátí seznam vypůjčených knih.
- Například:

```
pridej_knihu("Algoritmy v C","Robert","Sedgewick")
pridej_knihu("The Art of Computer Programming, Volume 1","Donald","Knuth")
pridej_knihu("The Art of Computer Programming, Volume 2","Donald","Knuth")

vypujceni("The Art of Computer Programming, Volume 1")
vypujceni("The Art of Computer Programming, Volume 2")

vk=vypujcene_knihy()
print("Seznam vypůjčených knih:")
for k in vk:
    print(k["nazev"])
```

- Vytiskne

```
Seznam vypůjčených knih:
The Art of Computer Programming, Volume 1
The Art of Computer Programming, Volume 2
```