

Zložitejšie dátové štruktúry

Poznámky k téme

Zoznamy, n-tice, množiny a slovníky sa dajú do seba **vnárať**: zoznam zoznamov vytvorí tabuľku, zoznam slovníkov databázu triedy, slovník zoznamov rozvrh. V tejto téme sa naučíš pracovať s dvojrozmernými zoznamami (mriežkami, maticami), s vnorenými štruktúrami a s dvoma dôležitými spôsobmi ukladania údajov – **zásobníkom** a **radom**.

Čo sa naučíš

- vytvoriť dvojrozmerný zoznam, pristupovať k prvkom `m[riadok][stĺpec]` a prechádzať ho,
- vyhnúť sa chybe so zdieľanými riadkami,
- pracovať so zoznamom slovníkov a slovníkom zoznamov,
- použiť zásobník (LIFO) a rad (FIFO) a vedieť, kde sa hodia.

1 Dvojrozmerný zoznam

Tabuľku (maticu, mriežku) uložíme ako **zoznam riadkov**, kde každý riadok je zoznam:

```
m = [
    [1, 2, 3],
    [4, 5, 6],
]
print(m[0])           # prvý riadok
print(m[1][2])        # riadok 1, stĺpec 2
print("riadkov:", len(m), "stĺpcov:", len(m[0]))
m[0][1] = 20
print(m)
```

Výstup:

```
[1, 2, 3]
6
riadkov: 2 stĺpcov: 3
[[1, 20, 3], [4, 5, 6]]
```

	stĺpec 0	stĺpec 1	stĺpec 2
riadok 0	<code>m[0][0] = 1</code>	<code>m[0][1] = 2</code>	<code>m[0][2] = 3</code>
riadok 1	<code>m[1][0] = 4</code>	<code>m[1][1] = 5</code>	<code>m[1][2] = 6</code>

Prvý index vyberá **riadok**, druhý **stĺpec** v ňom: `m[riadok][stĺpec]`.

1.1 Vytvorenie mriežky

Mriežku $r \times s$ vyplnenú nulami vytvoríme generátorovým zápisom – **každý riadok musí byť samostatný zoznam**:

```
r, s = 3, 4
mriezka = [[0] * s for _ in range(r)]
mriezka[1][2] = 7
for riadok in mriezka:
    print(riadok)
```

Výstup:

```
[0, 0, 0, 0]
[0, 0, 7, 0]
[0, 0, 0, 0]
```

Pozor: Zápis `[[0] * s] * r` vyzerá podobne, ale vytvorí `r`-krát **ten istý** riadok. Zmena jedného „riadku“ sa prejaví vo všetkých:

```
zle = [[0] * 4] * 3
zle[1][2] = 7
for riadok in zle:
    print(riadok)
```

Výstup:

```
[0, 0, 7, 0]
[0, 0, 7, 0]
[0, 0, 7, 0]
```

(Je to rovnaký jav ako `b = a` pri zoznamoch – viac mien pre jeden zoznam.)

1.2 Prechádzanie a výpis

Dva vnorené cykly: vonkajší cez riadky, vnútorný cez stĺpce (prvky riadku):

```
m = [[3, 8, 1], [9, 4, 7], [2, 6, 5]]
for riadok in m:
    for prvok in riadok:
        print(f"{prvok:3}", end=" ")
    print()
```

Výstup:

```
3  8  1
9  4  7
2  6  5
```

Ak potrebujeme indexy (napríklad pri susedoch či diagonále), prechádzame `range`:

```
m = [[3, 8, 1], [9, 4, 7], [2, 6, 5]]
n = len(m)
diagonala = [m[i][i] for i in range(n)]
print("hlavná diagonála:", diagonala, "súčet:", sum(diagonala))
for r in range(n):
    for s in range(n):
        if m[r][s] > 5:
            print(f"m[{r}][{s}] = {m[r][s]}")
```

Výstup:

```
hlavná diagonála: [3, 4, 5] súčet: 12
m[0][1] = 8
m[1][0] = 9
m[1][2] = 7
m[2][1] = 6
```

1.3 Súčty riadkov a stĺpcov

Súčet riadku je jednoduchý (`sum(riadok)`). Pre stĺpec musíme prejsť všetky riadky a z každého vziať prvok s rovnakým indexom:

```
predaj = [          # riadky: predajne, stĺpce: mesiace
    [120, 135, 110],
    [ 90, 100, 130],
    [150, 140, 160],
]
for i, riadok in enumerate(predaj):
    print(f"predajňa {i}: {sum(riadok)}")
for s in range(len(predaj[0])):
    stlpec = sum(predaj[r][s] for r in range(len(predaj)))
    print(f"mesiac {s}: {stlpec}")
```

Výstup:

```
predajňa 0: 365
predajňa 1: 320
predajňa 2: 450
mesiac 0: 360
mesiac 1: 375
mesiac 2: 400
```

2 Vnorené štruktúry

2.1 Zoznam slovníkov

Každý **záznam** (osoba, výrobok, kniha) je slovník s rovnakými kľúčmi; všetky záznamy tvoria zoznam. Je to v Pythone najbežnejší spôsob, ako uchovávať „tabuľkové“ údaje:

```
zaznamy = [
    {"meno": "Ema", "trieda": "3.A", "body": 47},
    {"meno": "Ján", "trieda": "3.B", "body": 38},
    {"meno": "Sofia", "trieda": "3.A", "body": 50},
]
for z in zaznamy:
    print(z["meno"], z["body"])

najlepsi = max(zaznamy, key=lambda z: z["body"])
print("Najviac bodov:", najlepsi["meno"])
trieda_a = [z["meno"] for z in zaznamy if z["trieda"] == "3.A"]
print("3.A:", trieda_a)
```

Výstup:

```
Ema 47
Ján 38
Sofia 50
Najviac bodov: Sofia
3.A: ['Ema', 'Sofia']
```

2.2 Slovník zoznamov

Ak k jednému kľúču patrí viac hodnôt, hodnotou je zoznam:

```
rozvrh = {
    "pondelok": ["matematika", "informatika", "fyzika"],
    "utorok": ["slovenčina", "informatika"],
}
rozvrh["utorok"].append("telesná")
for den, predmety in rozvrh.items():
    print(f"{den}: {' '.join(predmety)} ({len(predmety)} h)")
print("Informatika v pondelok:", "informatika" in rozvrh["pondelok"])
```

Výstup:

```
pondelok: matematika, informatika, fyzika (3 h)
utorok: slovenčina, informatika, telesná (3 h)
Informatika v pondelok: True
```

Prístup k vnoreným údajom sa číta zľava doprava: `rozvrh["utorok"]` je zoznam, `rozvrh["utorok"][0]` jeho prvý prvok.

Tip: V takýchto štruktúrach sa údaje často ukladajú do súborov vo formáte JSON. Modul `json` vie zoznamy a slovníky zapísať do textu (`json.dumps`) aj z neho prečítať (`json.loads`).

3 Zásobník

Zásobník (*stack*) je štruktúra, do ktorej pridávame aj z nej odoberáme **na tom istom konci** – ako kopa tanierov. Prvok, ktorý prišiel posledný, odchádza prvý: **LIFO** (*last in, first out*).

V Pythone je zásobníkom obyčajný zoznam: `append` pridá na vrch, `pop()` vráti a odoberie prvok z vrchu.

```
zasobnik = []
for x in ["A", "B", "C"]:
    zasobnik.append(x)
    print("vložené", x, "→", zasobnik)
while zasobnik:
    print("vybraté", zasobnik.pop(), "→", zasobnik)
```

Výstup:

```
vložené A → ['A']
vložené B → ['A', 'B']
vložené C → ['A', 'B', 'C']
vybraté C → ['A', 'B']
vybraté B → ['A']
vybraté A → []
```

Zásobník sa používa napríklad pri funkcii **Späť** (*undo*) v editoroch, pri návrate na predchádzajúcu stránku v prehliadači a pri volaniach funkcií (zásobník volaní – téma Rekurzia).

3.1 Kontrola zátvoriek

Klasická úloha so zásobníkom: sú zátvorky vo výraze správne spárované? Každú otváraciu zátvorku vložíme na zásobník. Pri zatváracej musí byť na vrchu zodpovedajúca otváracia – vyberieme ju. Na konci musí byť zásobník prázdny.

```
def spravne_zatvorky(vyraz):
    pary = {"(": ")", "[": "]", "{": "}"
    zasobnik = []
    for znak in vyraz:
        if znak in "([{":
            zasobnik.append(znak)
        elif znak in ")]}":
            if not zasobnik or zasobnik.pop() != pary[znak]:
                return False
    return len(zasobnik) == 0

for v in ["(a + b) * [c - d]", "((a + b)", "(a + b))", "[a + b]", "{[()]}"
    print(f"v:<20} {spravne_zatvorky(v)}")
```

Výstup:

```
(a + b) * [c - d]    True
((a + b)             False
(a + b))             False
[(a + b)]            False
{[()]}
```

4 Rad

Rad (fronta, *queue*) funguje ako rad pri pokladni: pridáva sa na **koniec**, odoberá sa zo **začiatku**. Čo prišlo prvé, odchádza prvé: **FIFO** (*first in, first out*).

So zoznamom by sme odoberali `pop(0)` – to je však pri dlhom zozname pomalé, lebo všetky ostatné prvky sa musia posunúť. Python má na rad štruktúru `deque` z modulu `collections` s rýchlymi operáciami na oboch koncoch:

```
from collections import deque

rad = deque()
rad.append("Ema")
rad.append("Ján")
rad.append("Sofia")
print("v rade:", list(rad))
prvy = rad.popleft()
print("obslúžený:", prvý, "→ v rade:", list(rad))
```

Výstup:

```
v rade: ['Ema', 'Ján', 'Sofia']
obslúžený: Ema → v rade: ['Ján', 'Sofia']
```

4.1 Simulácia radu pri pokladni

Zadanie: Ľudia prichádzajú k pokladni v zadaných minútach a obsluha jednej osoby trvá 3 minúty. Pre každú osobu vypíš, kedy bola obslúžená a koľko minút čakala.

Riešenie: Osoby vkladáme do radu v poradí príchodu. Pokladňa obsluhuje vždy osobu, ktorá je v rade prvá; obsluha začne, keď je pokladňa voľná a osoba už prišla.

```
from collections import deque

prichody = deque([("Ema", 0), ("Ján", 1), ("Sofia", 2), ("Tomáš", 9)])
cas = 0 # kedy je pokladňa voľná
while prichody:
    meno, prisiel = prichody.popleft()
    zaciatok = max(cas, prisiel)
    cas = zaciatok + 3
    print(f"{meno:<6} príchod {prisiel:>2}, obsluha {zaciatok:>2}–{cas:>2},",
          f"čakanie {zaciatok - prisiel} min")
```

Výstup:

```
Ema   príchod  0, obsluha  0– 3, čakanie 0 min
Ján   príchod  1, obsluha  3– 6, čakanie 2 min
Sofia príchod  2, obsluha  6– 9, čakanie 4 min
Tomáš príchod  9, obsluha  9–12, čakanie 0 min
```

Zásobník a rad sú základom mnohých algoritmov – napríklad prehľadávania bludiska alebo grafu.

5 Riešené úlohy

5.1 Piškvorky: kto vyhral?

Zadanie: Hracia plocha 3×3 je dvojrozmerný zoznam so znakmi "X", "O" a ".". Napíš funkciu, ktorá zistí, či má niektorý znak tri v rade – v riadku, stĺpci alebo na diagonále.

```
def vitaz(p):
    ciary = []
    for i in range(3):
        ciary.append(p[i])                # riadok i
        ciary.append([p[r][i] for r in range(3)]) # stĺpec i
        ciary.append([p[i][i] for i in range(3)]) # diagonála
        ciary.append([p[i][2 - i] for i in range(3)]) # vedľajšia diagonála
    for c in ciary:
        if c[0] != "." and c[0] == c[1] == c[2]:
            return c[0]
    return None

plocha = [
    ["X", "O", "X"],
    [".", "X", "O"],
    ["O", ".", "X"],
]
print("Tri v rade má:", vitaz(plocha))
```

Výstup:

```
Tri v rade má: X
```

Funkcia najprv zostaví všetkých 8 „čiar“ (3 riadky, 3 stĺpce, 2 diagonály) a potom každú skontroluje rovnakým spôsobom – vyhneme sa tak opakovaniu kódu.

5.2 Počet mín v susedstve

Zadanie: V mriežke sú míny (*) a prázdne políčka (.). Pre každé prázdne políčko vypíš počet mín na 8 susedných políčkach (ako v hre Míny).

Riešenie: Susedia políčka `pole[r][s]` majú indexy `r + dr`, `s + ds`, kde `dr` a `ds` sú `-1`, `0` alebo `1` (okrem dvojice `0, 0`). Pri okrajoch treba overiť, že index neleží mimo mriežky – inak by záporný index „pretiekol“ na druhú stranu a praveľký by spôsobil chybu.

```
pole = [
    "*... ",
    ".*. ",
    ".....",
]
riadkov, stlpcov = len(pole), len(pole[0])
for r in range(riadkov):
    vysledok = ""
    for s in range(stlpcov):
        if pole[r][s] == "*":
            vysledok += "*"
            continue
        miny = 0
        for dr in (-1, 0, 1):
            for ds in (-1, 0, 1):
                rr, ss = r + dr, s + ds
                if (dr, ds) != (0, 0) and 0 <= rr < riadkov and 0 <= ss < stlpcov:
                    if pole[rr][ss] == "*":
                        miny += 1
            vysledok += str(miny)
    print(vysledok)
```

Výstup:

```
*211
12*1
0111
```

Mriežku sme tu uložili ako zoznam reťazcov – reťazce sa tiež dajú indexovať `pole[r][s]`, iba sa nedajú meniť.

5.3 Transponovaná matica

Zadanie: Vymeň v matici riadky a stĺpce (prvý riadok sa stane prvým stĺpcom).

```
m = [[1, 2, 3], [4, 5, 6]]
t = [[m[r][s] for r in range(len(m))] for s in range(len(m[0]))]
print(t)
```

Výstup:

```
[[1, 4], [2, 5], [3, 6]]
```

6 Časté chyby

- `[[0] * s] * r` vytvorí zdieľané riadky – použi `[[0] * s for _ in range(r)]`.
- **Zámena indexov:** `m[riadok][stĺpec]`, nie naopak; pri súradniciach `x`, `y` je to `m[y][x]`.
- **Okraje mriežky:** pri susedoch kontroluj `0 <= r < pocet_riadkov`; záporný index nespôsobí chybu, ale vráti prvok z druhého konca!
- `pop(0)` na dlhom zozname je pomalé – na rad použi `deque.popleft()`.
- **Kopírovanie vnorených zoznamov:** `m.copy()` skopíruje iba vonkajší zoznam, riadky ostanú zdieľané. Kópia riadkov: `[riadok.copy() for riadok in m]`.

Zhrnutie

- Dvojmerný zoznam = zoznam riadkov; `m[r][s]`, `len(m)` riadkov, `len(m[0])` stĺpcov.
- Mriežku vytváraj `[[hodnota] * s for _ in range(r)]`.
- Prechádzanie: vnorené cykly; stĺpec: `[m[r][s] for r in range(len(m))]`.
- Zoznam slovníkov pre záznamy, slovník zoznamov pre viac hodnôt k jednému kľúču.
- Zásobník (LIFO): `append` + `pop()`; rad (FIFO): `deque` + `append` + `popleft()`.

Úlohy na precvičenie

1. Vytvor tabuľku násobenia 10×10 ako dvojmerný zoznam a vypíš ju zarovnanú.
2. Nájdi v matici najväčší prvok a vypíš jeho riadok a stĺpec.
3. Zisti, či je štvorcová matica symetrická (`m[r][s] == m[s][r]` pre všetky `r, s`).
4. Zo zoznamu slovníkov (meno, trieda, body) vypočítaj priemer bodov pre každú triedu.
5. Pomocou zásobníka obráť poradie slov vo vete.
6. Simuluj hru „horúci zemiak“: deti v kruhu si podávajú zemiak, každé `k`-té vypadne; vypíš poradie vypadávania (pomôže `deque.rotate`).

Úlohy k tejto téme nájdeš na learnwithmente.com v časti Informatika → Python.