Datové struktury

PVA2 Programování a vývoj aplikací

Adam Fišer

Obsah

Datové struktury

- **Datová struktura** je způsob, jakým jsou data organizována, uložena a zpracována v počítači.
- **Datová struktura** je základní stavební prvek programovacích jazyků.

Typy datových struktur

- Seznam `list`
- Tuple `tuple`
- Slovník `dict`


Seznam

Seznam `list`

- Nejuniverzálnější datová struktura Pythonu
- Zápis čárkou oddělených hodnot v hranatých závorkách
- Mohou obsahovat položky různých typů, ale obvykle jsou všechny položky v seznamu stejného typu.
- Stejně jako text, jsou položky indexovány.
- První položka má index 0, druhá 1 atd.
- Položky mohou být měněny, přidávány nebo odebírány.

Vytvoření seznamu

```
# Vytvoření seznamu
nazevSeznamu = [prvek, druhyPrvek]

# Přístup na konkrétní prvek seznamu
# V hranaté závorce se uvádí index prvku
nazevSeznamu[0]


list = [1, 2, 3, 4, 5]
```

```
txtVariable = 'var'
intVariable = 1918

# Vytvoření seznamu - kombinace hodnot a proměnnými
listFromVar = ['text', 2021, txtVariable, intVariable]

print(listFromVar)
#['text', 2021, 'var', 1918]
```

Operace se seznamem

Přístup k prvkům

- Stejně jako práce s textem, lze vrátit jen část prvků
- Prvky se indexují od 0
- Na konkrétní prvek se přistupuje pomocí hranatých závorek a čísla indexu
- Lze použít i záporné indexy, které počítají od konce seznamu
- Lze použít i rozsah prvků `[start:stop:step]` Index start je obsažen ve výstupu, ale index stop už ne `<start:stop)`.
- Pokud není uveden start, bere se od začátku seznamu, pokud není uveden stop, bere se do konce seznamu.

```
1   squares = [1, 4, 9, 16, 25]
2
3   print( squares[0] )     # první prvek - 1
4   print( squares[-1] )    # poslední prvek - 25
5   print( squares[3] )     # čtvrtý prvek - 16
6   print( squares[1:4] )     # rozsah prvků [4, 9, 16]
7   print( squares[1:4:2] )   # každý druhý prvek v rozsahu 1-4 tj [4,16]
8   print( squares[:3] )      # první tři prvky [1, 4, 9]
```

Přidání a změna prvků

- Datová struktura list je typu `mutable` tj. lze měnit její obsah.
- Změna prvku na konkrétním indexu `list[index] = novyPrvek`
- Přidání prvku na konec seznamu `list.append(prvek)`
- Přiřazení je možné k řezům seznamů, stejně jako k jednotlivým prvkům seznamu. Tímto způsobem lze dokonce měnit velikost seznamu nebo jej zcela vymazat.

```
1  cubes = [1, 8, 27, 65, 125]  # hups, chyba
2  # 4 ** 3  tzn. 4 na 3 je 64, ne 65!
3
4  # nahrazení chybné hodnoty
5  cubes[3] = 64
6  cubes[4] = 4**4
7
8  # přidání dalšího prvku
9  cubes.append( 4**5 )
```

```
1  animals = ["elephant", "lion", "tiger", "giraffe", "monkey", "dog"]
2  print(animals)
3
4  # Nahrazení dvou položek "lion" a "tiger" jednou "cat"
5  animals[1:3] = ["cat"]
6  print(animals)
```

Odstranění prvků

- Odstranění prvku
  - na konkrétním indexu `del list[index]`
  - podle hodnoty `list.remove(hodnota)`
  - na konkrétním indexu a vrácení hodnoty `list.pop(index)`
- Odstranění všech prvků `list.clear()`

Odstranění prvků

```python
animals = ["elephant", "lion", "tiger", "giraffe", "monkey", "dog"]

# Odstranění dvou položek dle indexu 1 a 2
# Podle indexu odpovídá prvkům s hodnotou "lion" a "tiger"
del animals[1:3]
print(animals) # ['elephant', 'giraffe', 'monkey', 'dog‘]

# Smazání všech položek
animals = []
print(animals)
```

Sjednocení

```python
# Sjednocení seznamu
list1 = [1, 2, 3]
list2 = [4, 5, 6]

sjednocenySeznam = list1 + list2
print(sjednocenySeznam)             # [1, 2, 3, 4, 5, 6]

print(sjednocenySeznam + [7, 8, 9]) # [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Funkce

- `len()` - vrátí délku seznamu
- `max()` - vrátí největší prvek seznamu
- `min()` - vrátí nejmenší prvek seznamu
- `sum()` - vrátí součet prvků seznamu
- `count()` - vrátí počet výskytů prvku v seznamu
- `sorted()` - vrátí seřazený seznam

```
1  list = [1, 8, 3, 3, 4, 5]
2
3  len(list)       # 6
4  max(list)       # 8
5  min(list)       # 1
6  sum(list)       # 24
7  list.count(3)   # 2
8  sorted(list)    # [1, 3, 3, 4, 5, 8]
```

Slovník

Slovník `dict`

- Slovník je datová struktura Pythonu, která je podobná seznamu.
- Slovník je typu `mutable` tj. lze měnit její obsah a `key-value` tj. obsahuje klíče a hodnoty.
- Zápis klíčů a hodnot oddělených dvojtečkou a jednotlivé položky oddělené čárkou v složených závorkách.
- Klíče mohou být různého typu, ale hodnoty obvykle stejného typu.
- Klíčem může být libovolný neměnný typ.
- Klíčem mohou být vždy řetězce a čísla; tuply lze použít jako klíče, pokud obsahují pouze neměnné objekty.
- Jako klíče nelze použít seznamy.
- Stejně jako seznamy, jsou položky indexovány.

Deklarace slovníku

```
# Vytvoření slovníku
nazevSlovniku = {klic: hodnota, klic2: hodnota2}
osobaV1 = {'jmeno': 'John', 'vek': 36, 'zeme': 'Norway'}

Slovník lze vytvářet i pomocí construktoru dict
osobaV2 = dict(jmeno='John', vek=36, zeme='Norway')

osobaV3 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Přístup k prvkům

```
# Přístup na konkrétní prvek slovníku
# V hranaté závorce se uvádí klíč prvku
nazevSlovniku[klic]

print(osobaV1['jmeno'])     # John
print(osobav1['vek'])       # 36
print(osobaV1['zeme'])      # Norway

print(osobaV2['vek'])       # 36
print(osobaV3['zeme'])      # Norway
```

Metody

- `keys()` vrátí seznam klíčů slovníku
- `values()` vrátí seznam hodnot slovníku
- `items()` vrátí seznam klíčů a hodnot slovníku
- `get()` vrátí hodnotu klíče, pokud klíč existuje, jinak vrátí `None`

```
1  osoba = {'jmeno': 'John', 'vek': 36, 'zeme': 'Norway'}
2
3  print(osoba.keys())             # ['jmeno', 'vek', 'zeme']
4  print(osoba.values())           # ['John', 36, 'Norway']
5  print(osoba.items())            # [('jmeno', 'John'), ('vek', 36), ('zeme', 'Norway')]
6  print(osoba.get('jmeno'))       # John
7  print(osoba.get('prijmeni'))    # None
```

Operátor `in`

- Operátor `in` se používá k zjištění, zda klíč nebo hodnota existuje v daném slovníku.
- Vrací `True` pokud klíč nebo hodnota existuje, jinak `False`

```
1  osoba = {'jmeno': 'John', 'vek': 36, 'zeme': 'Norway'}
2
3  print('jmeno' in osoba)         # True
4  print('prijmeni' in osoba)      # False
5  print('John' in osoba)          # False
```


n-tice (Tuple)

Tuple `tuple`

- Tuple je datová struktura Pythonu, která je podobná seznamu.
- Tuple je typu `immutable` tj. **nelze měnit** její obsah.
- Zápis čárkou oddělených hodnot v kulatých závorkách.
- Mohou obsahovat položky různých typů, ale obvykle jsou všechny položky v tuple stejného typu.
- Stejně jako seznamy, jsou položky indexovány.

```
# Vytvoření tuple
nazevTuple = (prvek, druhyPrvek)

# Přístup na konkrétní prvek tuple
# V hranaté závorce se uvádí index prvku
nazevTuple[index]
```

Děkuji za pozornost

Otázky?

Repository / Prezentace