## 8. Programování - seznamy, n-tice, slovníky

### *Seznamy*

Seznam (list) je v Pythonu seřazená měnitelná kolekce hodnot. Podobný typu pole, array v jiných pr. jazycích.

**SYNTAX**

- uvádějí se v **[ ]**, jednotlivé hodnoty se oddělují čárkou
- mohou obsahovat libovolná data i další seznamy

**VYUŽITÍ**

Využívají se pro uložení více hodnot do jedné proměnné, do které se dá v průběhu programu vkládat další hodnoty, mazat atd.

**INDEXOVÁNÍ**

K jednotlivým prvkům lze přistoupit pomocí [ ]. Kladné hodnoty číslují prvky od začátku - začíná 0, záporné od konce.

*print(seznam[0])* výpis prvku na indexu 0 (první)

*print(seznam[-1])* výpis posledního prvku - záporné indexy od konce

**SLICE**

Se všemi seznamovými datovými typy můžeme provádět vyřezy (slice). Výřez je nový objekt.

→ s[zacatek:konec:krok] -- ale zacatek, konec i krok lze vynechat - s[:] je kopie.

*print(seznam[4:])* podseznam [4..n]

*print(seznam[1:4])* podseznam [1..3]

*print(seznam[1:6:2])* podseznam [1..5] každý 2. prvek

**PŘIŘAZENÍ HODNOTY** - *seznam[2]="Třetí"*

**SČÍTÁNÍ / SPOJOVÁNÍ** - *seznam=[1, 2] + seznam*

**NÁSOBENÍ CELÝM ČÍSLEM** - *seznam=[0]*10*

POZOR! - pokud je v seznamu objekt (seznam, slovník…), znásobí se pouze jeho reference na místo v paměti, nevytvoří se nový objekt jako kopie

**PROCHÁZENÍ CYKLEM FOR**

```
for prvek in seznam:
    print(prvek)
```

**ROZBALENÍ**

*seznam=[4, 2, 7]*

*print(*seznam) # stejné jako print(4, 2, 7)* *>4 2 7*

**ZJIŠTĚNÍ PŘÍTOMNOSTI (in)** - “a” in seznam

**ZJIŠTĚNÍ DÉLKY** - funkce len(s)

**PŘEVODNÍ FUNKCE** - list()

převádí iterovatelný datový typ na seznam

*print(list(range(10,20)))*

Převedení řetězce na seznam a zpět (s mezerou mezi):

*pismena=list(“Ahoj”)*

*ret = “ “.join(pismena)*

### *operace se seznamy*

| | |
|---|---|
| .append(hodnota) | přidá hodnotu na konec seznamu |
| .insert(i, hodnota) | vloží hodnotu na pozici v seznamu |
| .pop(i) | odstraní prvek z dané pozice a vrátí ho (návratová hodnota) |
| | (pokud neuvedeme index, odebírá z konce seznamu) |
| .remove(hodnota) | odebere první výskyt této hodnoty |
| .count(hodnota) | vrátí počet výskytů dané hodnoty |
| .index(hodnota) | vrátí index dané hodnoty v seznamu |
| .sort() | setřídí seznam podle velikosti vzestupně, řetězce abecedně |
| | parametr reverse=True pro sestupné |
| | smíšená data od Python3 nelze třídit! |
| .reverse() | otočení (přehození) položek seznamu |
| .copy() | vrátí mělkou kopii seznamu, stejné jako [:] |

### *N-tice - definice, použití*

N-tice (tuple) je v Pythonu seřazená neměnná kolekce hodnot.

**SYNTAX**

- vytvářejí se pomocí **( )**, jednotlivé hodnoty se oddělují čárkou
- jednoprvková n-tice n=(17,) - čárka před koncovou závorkou pro rozlišení od závorek určujících pořadí operací
- mohou obsahovat **libovolná data** i další n-tice
- jsou **neměnné**, obdobně jako řetězce -- POZOR! - pokud obsahují referenci na měnitelný objekt (seznam, slovník…), tak ty měnit můžeme

**VYUŽITÍ**

Používají se pro uložení většího počtu hodnot než 1 - např. souřadnice (x, y), předávání parametrů do funkcí, předávání (r, g, b).

**INDEXOVÁNÍ, SLICE**

**SČÍTÁNÍ / SPOJOVÁNÍ**

**NÁSOBENÍ CELÝM ČÍSLEM**

**PROCHÁZENÍ CYKLEM FOR**

**ROZBALENÍ**

**ZJIŠTĚNÍ PŘÍTOMNOSTI (in)**

**ZJIŠTĚNÍ DÉLKY**

stejné jako řetězce, seznamy

**PŘEVODNÍ FUNKCE** - tuple()

převádí iterovatelný datový typ na n-tici.

*pismena=tuple(“Ahoj”)*

**METODY**

| | |
|---|---|
| .count(hodnota) | vrátí počet výskytů dané hodnoty |
| .index(hodnota) | vrátí index dané hodnoty |

## *Slovníky a operace s nimi*

Slovník (dictionary, hashtable, asociativní pole) je uspořádaná (>Py3.7) kolekce dvojic klíč:hodnota.

**SYNTAX**

- vytvářejí se pomocí **{ }**, dvojice se oddělují čárkou
- klíče musejí být unikátní a neměnné (str, tuple, int) - NE list nebo slovník
- hodnoty mohou být libovolného datového typu

**VYUŽITÍ**

Používají se pro uložení informací podle unikátních klíčů - id, telefon… Oproti seznamům umožňují mnohonásobně rychlejší přístup k datům.

**INDEXOVÁNÍ**

Hodnoty můžeme získat na základě klíče, ale ne opačně.

*print(slov[“jm”])*

Pokud klíč neexistuje, nastane výjimka - můžeme použít metodu **.get(**klíč, default=None**)**

**PŘIŘAZENÍ HODNOTY** - *slov[“jm“]=“Josef“*

Pokud je při přiřazení použit klíč, který neexistuje, vytvoří se.

**SČÍTÁNÍ / SPOJOVÁNÍ**

*slov.update({'a': 123, 'b': 985})*

Dále s převodní fcí

```
slov=dict(slov, a=123, b=985)
```

… s rozbalením

```
slov=dict(slov, **slov2)
```

**PROCHÁZENÍ CYKLEM FOR**

Řídící proměnná nabývá klíčů, ne hodnot.

```
for i in slovnik:
    print (i,":",slovnik[i])
```

**ROZBALENÍ**

*slovnik={“sep”: ”,”, “end”: “\t”}*

*print(“.”, **slovnik)*

**ZJIŠTĚNÍ PŘÍTOMNOSTI (in)** - “klic” in slovnik

Hodnotu zjistíme takto:

```
if “Jana” in slov.values(): …
```

**ZJIŠTĚNÍ DÉLKY** - *c=len(slovník)*

**PŘEVODNÍ FUNKCE** - dict()

**METODY**

| | |
|---|---|
| .pop(k, vych) | odstraní dvojici podle klíče; pokud neexistuje, vrací výjimku nebo výchozí par. |
| | *# if k in slov: del slov(k)* |
| .popitem() | odstraní a vrátí poslední vloženou dvojici |
| .clear() | vymaže obsah slovníku |
| | |
| .keys() | vrátí seznam klíčů ze slovníku v podobě speciálního objektu |
| .values() | vrátí seznam hodnot v podobě speciálního objektu |
| .items() | vrátí seznam dvojic (klíč, hodnota) v podobě speciálního objektu |
| | Pro další práci je většinou převádíme na list. |

***Množiny (navíc)***

Množina je neuspořádaná měnitelná kolekce jedinečných prvků.

- Vytváří se pomocí **{ }** a/nebo pomocí funkce set(). Prázdná množina se vytváří výhradně pomocí funkce set() -- prázdné složené závorky by vedly k vytvoření prázdného slovníku.
- Přidání duplicitních prvků je bezpečné, ale bezúčelné.

**Metody:**

| | |
|---|---|
| s.add(x) | Přidá prvek x do množiny s. |
| s.update(t) | Přidá do množiny s prvky z množiny t. |
| s.pop() | Odstraní náhodný prvek a vydá ho jako svou návratovou hodnotu. |
| s.remove(x) | Odstraní z množiny s prvek x. |
| s.clear() | Odstraní všechny prvky z množiny s. |
| s.copy() | Vrátí mělkou kopii množiny s. |

**Množinové operace**

| | |
|---|---|
| a \| b | Sjednocení -- vrátí všechny prvky množiny a i b. |
| a & b | Průnik -- vrátí všechny prvky společné množině a a b. |
| a - b | Rozdíl -- vrátí množinu a zbavenou všech prvků, které byly obsaženy v b |
| | |
| == | Testuje shodnost množin |
| in | Testuje příslušnost prvku v množině |
| a < b | a je podmnožinou b. |
| a <= b | a je podmnožinou nebo ekvivalentem b. |
| a > b | a je nadmnožinou b. |
| a >= b | a je nadmnožinou nebo ekvivalentem b. |