Struktury danych

Piotr Pawlik

Wykład 2019

Struktury sterujące

Wprowadzenie standardowych „struktur sterujących”:

- sekwencja
- selekcja (wybór, if-else, switch)
- pętla
- podprogram (jedno wejście, JEDNO wyjście)

Instrukcje sterujące

Instrukcje powtarzania (pętle)

Przykład - Pętla 'while'

```
i=1
while i < 10:
    print(i**2)
    i+=1
```

Przykład - Pętla 'for each'

```
for i in range(1,10):
    print(i**2)
```

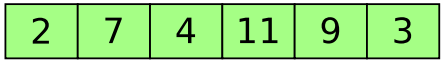

Struktury danych

Wprowadzenie standardowych „struktur danych”:

- tablica
- rekord
- lista, stos, kolejka,
- mapa (tablica asocjacyjna)
- zbiór
- drzewo, graf

Tablice

Tablica jest indeksowaną kolekcją elementów tego samego typu (elementy są dostępne poprzez numer/indeks)



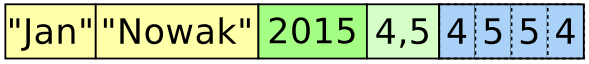
Tablice

Operacje wykonywane na elementach tablicy:

- odwołanie do i-tego elementu - tab[i]
- rekord
- lista, stos, kolejka,
- mapa (tablica asocjacyjna)
- zbiór
- drzewo, graf

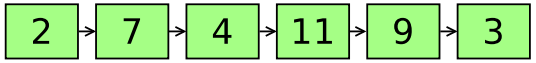
Rekordy

Rekord jest kolekcją elementów **różnych** typów (pól) dostępnych poprzez nazwę.



Listy

Lista jest kolekcją elementów tego samego typu o dostępie sekwencyjnym (element po elemencie)



Mapy

Mapa jest kolekcją uporządkowanych par (unikatowy_klucz, wartość). Dostęp do wartości w mapie następuje poprzez klucze.

"Prezydent"	"Duda"
"Premier"	"Morawiecki"
"Marszałek"	"Witek"

Zbiory

Zbiór jest nieuporządkowaną kolekcją różnych (unikatowych) elementów tego samego typu. Brak bezpośredniego dostępu do elementów zbioru.

Stosy, kolejki

Struktury, podobnie jak listy, o dostępie sekwencyjnym różniące się sposobem wstawiania i przechodzenia do kolejnych elementów.

Drzewa, grafy

Struktury nieliniowe - każdy z elementów może być powiązany z kilkoma 'następnikami' i 'poprzednikami' (tylko grafy). Wymusza to stosowanie bardziej zaawansowanych algorytmów przeglądania.

Struktury danych w Pythonie

- tablica - BRAK - substytut: typ List
- rekord - BRAK - substytut: typy List, Tuple (ew. Dictionary)
- lista, stos, kolejka, - typ List
- mapa (tablica asocjacyjna) - typ Dictionary
- zbiór - typ Set
- drzewo, graf - BRAK

Listy

Listy tworzymy używając nawiasów kwadratowych, rozdzielając elementy przecinkami.

X=[1,2,3,4,5]
Osoba=['Piotr','Pawlik', 2018]

Lista jest typem zmienialnym:

Osoba[0]='Jan'

Struktury danych oparte na typie List

Stos

```
stos = [7]
stos.append(6)
stos.append(5)
print(stos)      # [7,6,5]
el = stos.pop()
print(el, stos) # 5 [7,6]
el = stos.pop()
print(el, stos) # 6 [7]
el = stos.pop()
print(el, stos) # 7 [ ]
```

Kolejka

```
kolejka = [7]
kolejka.append(6)
kolejka.append(5)
print(kolejka) # [7,6,5]
el = kolejka.pop(0)
print(el, kolejka)
                  # 7 [6,5]
el = kolejka.pop(0)
print(el, kolejka) # 6 [5]
el = kolejka.pop(0)
print(el, kolejka) # 5 [ ]
```


Inne metody tworzenia listy

Przez append

```
lista = [ ]
for x in range(10)
   lista.append(x*2)
```

To samo za pomocą 'wytwornika listy' (tzw. list comprehension):

Przez 'wytwornik'

```
lista = [x*2 for x in range(10)]
```

Krotki (touples)

Krotki (ang. tuples) tworzymy jako sekwencje elementów rozdzielonych przecinkami, zazwyczaj objęte nawiasami okrągłymi.

X=(1,2,3,4,5)
Osoba=('Piotr','Pawlik', 2018)

Lista jest typem **niezmienialnym**! Nie zadziała:
Osoba[0]='Jan'

Krotki (a więc obiekty niezmienialne) mogą zawierać np. listy (czyli obiekty zmienialne).

Inne metody tworzenia krotki

Można wykorzystać notację zbliżoną do 'wytwornika listy' (list comprehension):

Przez 'wytwornik'

```
g = (x*2 for x in range(10))
```

ale wynikiem jest tzw. generator, a nie krotka. Aby uzyskać krotkę trzeba wykonać:

Rzutowanie

```
krotka = tuple(g)
```


Krotki jednolemenentowe

Do utworzenia krotki jednoelementowej konieczna jest notacja z przecinkiem, np.:

Krotka jednolemenentowa

```
krotka_jednoelementowa = (3,)
```

W przeciwnym wypadku (bez przecinka) 'zmienna' krotka_jednoelementowa będzie zwykłą liczbą typu int

Słowniki (mapy)

Słowniki tworzymy używając nawiasów klamrowych { }, rozdzielając pary elementów przecinkami, a elementy w parze dwukropkiem.

Przykłady

```
Osoby={'Prezydent':'Duda', 'Premier':'Morawiecki'}

print Osoby['Premier'];

for osoba in Osoby :
   print 'Aktualnie ', osoba , ' to ', Osoby[osoba]

for osoba, nazwisko in Osoby.items() :
   print ('Aktualnie ', osoba , ' to ', nazwisko);
```

Słowniki(mapy)

Kluczem słownika może być dowolny typ, np. krotka:

Przykład

```
książka = {('Adam', 'Nowak') : '601234567',  ('Jan',
     'Kowalski') : '600778899'}

for osoba, numer in książka.items() :
   print (osoba[0], '␣␣', osoba[1], '␣ma␣numer␣',
       numer);
```


Słowniki składane (dictionary comprehension)

Przykład

```
>>> znak2cyfra = {str(i):i for i in [1,2,3,4,5]}
# znak2cyfra == {'1': 1, '3': 3, '2': 2, '5': 5,
    '4': 4}

>>> znaki = ['1', '2', '3', '4', '5' ]
>>> znak2cyfra = {z:c for c,z in enumerate(znaki,1)}
# znak2cyfra == {'1': 1, '3': 3, '2': 2, '5': 5,
    '4': 4}

>>> cyfra2znak = {c:z for c,z in enumerate(znaki,1)}
# cyfra2znak == {1: '1', 2: '2', 3: '3', 4: '4', 5:
    '5'}
```


Zbiory

Zbiory tworzymy używając nawiasów klamrowych { }, rozdzielając elementy przecinkami.

Przykłady

```
zbior={'Adam', 'Ewa', 'Piotr', 'Paweł'}
zbior_compr = { x*2 for x in  range(3) }
```


Tworzenie zbiorów przez rzutowanie

Zbiory można tworzyć z innych typów sekwencyjnych przez rzutowanie.

Przykład

```
zbior=set(['Adam', 'Ewa', 'Piotr', 'Paweł', 'Ewa'])
```

'Ewa' w zbiorze zbior wystąpi tylko raz!

Operacje na strukturach danych

Dla struktur danych określa się czy i jak dane:

- odczytać
- wstawić
- usunąć
- wyszukać
- sortować

Operacje na strukturach danych

Operacje odczytu na strukturach danych:

- tablica - []
- rekord - .
- lista - head, tail
- stos, kolejka - peek
- mapa - []
- zbiór - brak

Operacje na strukturach danych

Operacje wstawiania do struktur danych:

- tablica - []/nadpisanie
- rekord - ./nadpisanie
- lista - append, at
- stos, kolejka - push, inqueue
- mapa - []/nadpisanie
- zbiór - add

Operacje na strukturach danych

Operacje usuwania ze struktur danych:

- tablica - brak/nadpisanie
- rekord - brak/nadpisanie
- lista - remove
- stos, kolejka - pop, dequeue
- mapa - brak/nadpisanie
- zbiór - remove

Struktury danych w Pythonie

Wybrane operacje dostępne na sekwencjach

- Zawieranie: x in a, x not in a
- Konkatenacja: +
- Powielanie: *
- Wybór: a[i] ; możliwe a[-1]
- Wykrawanie: a[i:j] ;Rozszerzone : a[i:j:k]
- Liczebność: len(a)
- Indeks pierwszego wystąpienia: a.index(x)
- Zliczanie wystąpień: a.count(x)
- Usuwanie wszystkiego: a.clear( )

Struktury danych w Pythonie

Wybrane operacje dostępne na listach

- Usuwanie elementów: del a[i:j:k]
- Dodawanie elementu: a.append(x)
- Rozszerzanie: a.extend([x])
- Wstawiane elementu: a.insert(i, x)
- Pobieranie elementu: a.pop(i) / a.pop( )
- Usuwanie elementu: a.remove(x)
- Odwracanie (in situ): a.reverse( )
- Sortowanie (in situ): a.sort( )

Struktury danych w Pythonie

Wybrane operacje dostępne na napisach

- Dzielenie napisu na wyrazy (może mieć parametr - separator): a.split( )
- Łączenie napisów (parametrem jest napis lub lista napisów, s - separator): s.join([x])
- Szukanie fragmentu: a.find(x)
- Zastępowanie fragmentu: a.replace(x, x)
- Usuwanie spacji: a.strip( );lstrip( ), rstrip( )
- Wielkość liter: a.upper( ); lower( )
- Wyrównywanie a.center(d); ljust(d), rjust(d)

Struktury danych w Pythonie

Wybrane operacje dostępne na zbiorach

- len(a), in, not in, a.add(x), a.pop(), a.clear(), a.remove(x)/a.discard(x)
- Jest podzbiorem?: a.issubset(b); a <= b
- Jest nadzbiorem?: a.issuperset(b); a >= b
- Suma zbiorów: a.union(b); a | b
- Różnica zbiorów: a.difference(b); a – b
- Przecięcie zbiorów: a.intersection(b); a & b
- Różnica symetryczna zbiorów: a.symetric_diference(b); a ^ b