Python: Struktury Danych w języku Python.

W języku Python istnieją cztery wbudowane struktury danych: lista, krotka, słownik i zbiór (list, tuple, dictionary, set).

Lista

Jest to kolekcja uporządkowanych elementów (elementy zachowują swoją kolejność). Listy możemy modyfikować: dodawać i usuwać elementy bez konieczności tworzenia nowych instancji listy (tak jak to jest np. w przypadku zmiennych typu string czy krotek) – są mutable. Do elementów listy możemy dostać się z użyciem nawiasów kwadratowych. Elementy w liście mogą się powtarzać.

```
shoplist = ['apple', 'mango', 'carrot', 'banana']

print('I have', len(shoplist), 'items to purchase.')

print('These items are:', end=' ')
for item in shoplist:
        print(item, end=' ')

print('\nI also have to buy rice.')
shoplist.append('rice')
print('My shopping list is now', shoplist)

print('I will sort my list now')
shoplist.sort()
print('Sorted shopping list is', shoplist)

print('The first item I will buy is', shoplist[0])
olditem = shoplist[0]
del shoplist[0]
print('I bought the', olditem)
print('My shopping list is now', shoplist)
```

Krotka

Jest to kolekcja uporządkowanych elementów. Krotka jest niezmienna (immutable) – jeśli zachodzi konieczność jej modyfikacji, musimy stworzyć jej kopię. Krotki tworzymy z uzyciem nawiasów zwykłych a do elementów dostajemy się za pomocą kwadratowych. Może zawierać duplikaty.

```
zoo = ('python', 'elephant', 'penguin')
print('Number of animals in the zoo is', len(zoo))

new_zoo = 'monkey', 'camel', zoo
print('Number of cages in the new zoo is', len(new_zoo))
print('All animals in new zoo are', new_zoo)
print('Animals brought from old zoo are', new_zoo[2])
print('Last animal brought from old zoo is', new_zoo[2][2])
print('Number of animals in the new zoo is', len(new_zoo)-1+len(new_zoo[2]))
```

Słownik

Jest to kolekcja nieuporządkowanych elementów. W przeciwieństwie do poprzednich struktur kluczem (indeksem) w słowniku nie musi być liczba naturalna – koniecznie jednak musi być to zmienna typu immutable (najczęściej, choć niekoniecznie - string). Klucze muszą być unikalne w obrębie słownika, wartości natomiast mogą się powtarzać. Słownik intuicyjnie można sobie wyobrazić jako… słownik gdzie kluczem jest dane słowo a wartością jego tłumaczenie lub definicja. Innym przykładem może być książka telefoniczna (kluczem jest nazwisko, a wartością nr telefonu). Do operacji na słowniku używamy nawiasów klamrowych i dwukropków. Przy odczytywaniu elementów korzystamy z kwadratowych.

```
ab = {
        'Swaroop': 'swaroop@swaroopch.com',
        'Larry': 'larry@wall.org',
        'Matsumoto': 'matz@ruby-lang.org',
        'Spammer': 'spammer@hotmail.com'
}

print("Swaroop's address is", ab['Swaroop'])

del ab['Spammer']

print('\nThere are {} contacts in the address-book\n'.format(len(ab)))

for name, address in ab.items():
        print('Contact {} at {}'.format(name, address))

ab['Guido'] = 'guido@python.org'

if 'Guido' in ab:
        print("\nGuido's address is", ab['Guido'])
```

Zbiór

Jest to kolekcja nieuporządkowanych i **unikalnych** elementów (nie da się dodać drugiego, takiego samego elementu). Zbiory stosuje się tam gdzie istnienie elementów w jest istotniejsze niż ich kolejność albo liczba wystąpień. Do definiowania zbiorów służy słowo kluczowe set() lub nawiasy

klamrowe, a operuję się na nim za pomocą dedykowanych metod i nawiasów kwadratowych. Słownik można modyfikować.

```
bri = set(['brazil', 'russia', 'india'])
print('india' in bri)
print('usa' in bri)
bric = bri.copy()
bric.add('china')
bric.issuperset(bri)
bri.remove('russia')
print(bri & {'brazil', 'india'})
```

Alternatywnie:

```
bri = {'brazil', 'russia', 'india'}
```