**Lists**

Listę tworzy się w Pythonie za pomocą nawiasów kwadratowych ([, i]) i oddzielając wartości przecinkami. Oto przykładowa lista:

>>> my\_list = [1, 2, 3, 4, 5]

**Reading from Lists**

Aby uzyskać dostęp do pojedynczego elementu listy, możesz użyć indeksu, a Python używa systemu indeksu opartego na zerach:

>>> my\_list[0]

1

>>> my\_list[1]

2

Jeśli spróbujemy uzyskać dostęp do indeksu, który jest zbyt wysoki (lub zbyt niski), otrzymamy błąd:

>>> my\_list[5]

Traceback (most recent call last):

File "", line 1, in

IndexError: list index out of range

Aby upewnić się, że nie próbujemy uzyskać indeksu, który jest poza zakresem, możemy przetestować długość za pomocą funkcji len (a następnie odjąć 1):

>>> len(my\_list)

5

Dodatkowo możemy uzyskać dostęp do podsekcji listy przez "cięcie" jej. Dostarczamy indeks początkowy i indeks końcowy (obiekt w tym indeksie nie zostanie uwzględniony).

>>> my\_list[0:2]

[1, 2]

>>> my\_list[1:0]

[2, 3, 4, 5]

>>> my\_list[:3]

[1, 2, 3]

>>> my\_list[0::1]

[1, 2, 3, 4, 5]

>>> my\_list[0::2]

[1, 3, 5]

**Modifying a List**

W przeciwieństwie do ciągów, których nie można zmienić (nie można zmienić znaku w ciągu), można zmienić wartość na liście, używając operacji równości:

>>> my\_list[0] = "a"

>>> my\_list

['a', 2, 3, 4, 5]

Jeśli chcemy dodać do listy, możemy użyć metody .append. Jest to przykład metody, która modyfikuje obiekt wywołujący metodę:

>>> my\_list.append(6)

>>> my\_list.append(7)

>>> my\_list

['a', 2, 3, 4, 5, 6, 7]

Listy mogą być sumowane (łączone):

>>> my\_list + [8, 9, 10]

['a', 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> my\_list += [8, 9, 10]

>>> my\_list

['a', 2, 3, 4, 5, 6, 7, 8, 9, 10]

Pozycje na listach można również ustawić za pomocą wycinków:

>>> my\_list[1:3] = ['b', 'c']

>>> my\_list

['a', 'b', 'c', 4, 5, 6, 7, 8, 9, 10]

# Replacing 2 sized slice with length 3 list inserts new element

my\_list[3:5] = ['d', 'e', 'f']

print(my\_list)

Możemy usunąć sekcję listy, przypisując pustą listę do plasterka:

>>> my\_list = ['a', 'b', 'c', 'd', 5, 6, 7]

>>> my\_list[4:] = []

>>> my\_list

['a', 'b', 'c', 'd']

Usuwanie pozycji z listy na podstawie wartości można wykonać za pomocą metody .remove:

>>> my\_list.remove('b')

>>> my\_list

['a', 'c', 'd']

Próba usunięcia i element, którego nie ma na liście, spowoduje błąd:

>>> my\_list.remove('f')

Traceback (most recent call last):

File "", line 1, in

ValueError: list.remove(x): x not in list

Elementy można również usuwać z końca listy za pomocą metody pop:

>>> my\_list = ['a', 'c', 'd']

>>> my\_list.pop()

'd'

>>> my\_list

['a', 'c']

Możemy również użyć metody pop, aby usunąć elementy z określonego indeksu:

>>> my\_list.pop(0)

'a'

>>> my\_list

['c']

>>> my\_list.pop(1)

Traceback (most recent call last):

File "", line 1, in

IndexError: pop index out of range

>>> [].pop()

Traceback (most recent call last):

File "", line 1, in

IndexError: pop from empty list