Jak również wspominałem w rozdziale 4., funkcja wbudowana `map` ma podobne działanie, ale zamiast wyrażeń na elementach sekwencji wywoływane są funkcje, a wyniki zwracane są w postaci nowej sekwencji:

```
>>> list(map(abs, [ 1, -2, 0, 1, 2]))        # Wywołanie funkcji map na sekwencji
[1, 2, 0, 1, 2]
```

Na tym etapie książki nie jesteśmy jeszcze gotowi na przekazanie pełnych informacji dotyczących iteratorów, zatem odłożymy to zagadnienie do późniejszych rozdziałów. W dalszej części niniejszego rozdziału wrócimy jednak na chwilę do tego zagadnienia przy okazji podobnych wyrażeń słowników składanych.

Indeksowanie, wycinki i macierze

Ponieważ listy są sekwencjami, indeksowanie i wycinki działają w ten sam sposób dla list, jak i dla łańcuchów znaków. Wynikiem zindeksowania listy jest jednak dowolny typ obiektu znajdujący się na pozycji o podanej wartości przesunięcia, natomiast wycinek listy zawsze zwraca nową listę.

```
>>> L = ['mielonka', 'Mielonka', 'MIELONKA!']
>>> L[2]                                      # Wartości przesunięcia rozpoczynają się od 0
'MIELONKA!'
>>> L[-2]                                     # Wartość ujemna: odliczamy od końca
'Mielonka'
>>> L[1:]                                     # Wycinek pobiera części listy
['Mielonka', 'MIELONKA!']
```

Jedna uwaga: ponieważ wewnątrz list można zagnieżdżać inne listy (oraz inne typy), czasami w celu wejścia w głąb struktury danych niezbędne będzie połączenie ze sobą kilku operacji indeksowania. Jednym z łatwiejszych sposobów reprezentowania w Pythonie macierzy (tablic wielowymiarowych) są listy z zagnieżdżonymi podlistami. Poniżej widać prostą dwuwymiarową tablicę 3×3 opartą na listach.

```
>>> matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Jeden indeks pozwala nam pobrać cały wiersz (a tak naprawdę zagnieżdżoną podlistę), natomiast dwa indeksy — pojedynczy element tego wiersza.

```
>>> matrix[1]
[4, 5, 6]
>>> matrix[1][1]
5
>>> matrix[2][0]
7
>>> matrix = [[1, 2, 3],
...           [4, 5, 6],
...           [7, 8, 9]]
>>> matrix[1][1]
5
```

W powyższym kodzie widać, że listy w naturalny sposób mogą się rozciągać na kilka wierszy, jeśli tego chcemy, ponieważ znajdują się one w parze nawiasów kwadratowych (więcej informacji na temat składni w kolejnej części książki). W dalszej części niniejszego rozdziału zobaczymy macierz utworzoną za pomocą słownika. W przypadku większych zadań z dziedziny programowania numerycznego przyda się omówione w rozdziale 5. rozszerzenie NumPy, które również obsługuje macierze.