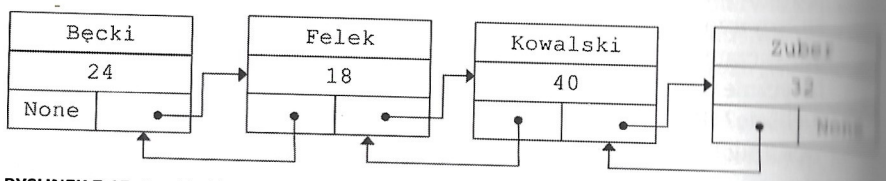
## Lista dwukierunkowa

Listy jednokierunkowe są bardzo wygodne w stosowaniu i wymagają względnie mało pamięci. Mimo to operacje na nich niekiedy zajmują dużo czasu. Zatem żyło to wiele osób i tym sposobem zostały wymyślone inne typy list. Jednym z tych kawszych pomysłów jest lista dwukierunkowa (rysunek 7.15).




**RYSUNEK 7.15.** Przykład listy dwukierunkowej

Każda komórka robocza zawiera wskaźniki do elementów poprzedniego i następnego.

- Pierwsza komórka znajdująca się w liście nie posiada swojego poprzednika; zaznaczamy to, wpisując wartość `None` do pola `poprzedni`.
- Ostatnia komórka znajdująca się w liście nie posiada swojego następnika; zaznaczamy to, wpisując wartość `None` do pola `nastepny`.

W porównaniu z klasyczną listą jej dwukierunkowa wersja jest nieco bardziej kosztowna, jeśli chodzi o zajętość pamięci, ale czasem szybkość działania jest ważniejsza niż małe zużycie pamięci — a właśnie szybkość działania jest zaletą listy dwukierunkowej.

W liście dwukierunkowej z każdego elementu możesz przejść do elementu następnego lub poprzedniego (zwanych, odpowiednio, *następnikiem* i *poprzednikiem*).

Popatrzmy, jak może wyglądać przykładowa realizacja listy dwukierunkowej (*Lista2Kier.py* w katalogu *MojeTypy*):

```
class Element:       # Rekord danych
    def __init__(self, pNazwisko="Doe", pWiek=0):
        self.nazwisko = pNazwisko
        self.wiek = pWiek
        self.nastepny = None
        self.poprzedni = None

class Lista2Kier:    # Właściwa lista dwukierunkowa
    def __init__(self):
        self.glowa = None
        self.ogon = None
```

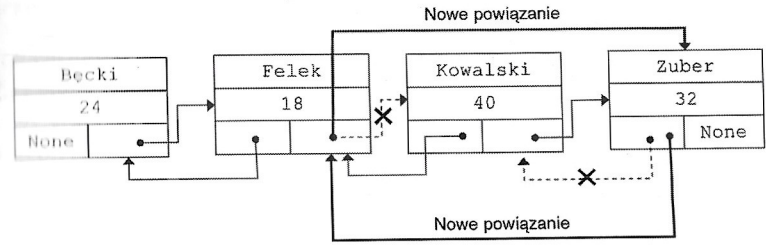
Jak może wyglądać dodawanie nowego elementu do takiej listy? Cały wysiłek sprowadza się do wstawienia nowego rekordu przy jednoczesnym poprawnym uzupełnieniu referencji do obiektów poprzedniego i następnego — jeśli takowe istnieją. Oto możliwa realizacja:

```
def wstaw(self, pNazwisko, pWiek):  # Proste wstawianie na koniec listy
    nowy = Element(pNazwisko, pWiek)
    if self.glowa !=None:
        self.ogon.nastepny = nowy
        nowy.poprzedni = self.ogon
        self.ogon=nowy
    else:
        self.glowa = nowy
        self.ogon =  nowy                    }
```

Wstawianie danych mamy już za sobą, zajmijmy się teraz ich usuwaniem.

Załóżmy, że podczas przeglądania elementów listy dwukierunkowej zapamiętaliśmy wskaźnik pozycji bieżącej `p`. (Przykładowo, szukaliśmy elementu spełniającego pewien warunek i na wskaźniku `p` nasze poszukiwania zakończyły się sukcesem). Jak usunąć element `p` z listy? Już wiemy, że do prawidłowego wykonania tej operacji niezbędna jest znajomość wskaźników `przed` i `po`, wskazujących, odpowiednio, na komórki poprzednią i następną. W przypadku listy dwukierunkowej w komórce wskazywanej przez `p` te dwie informacje już się znajdują i wystarczy tylko po nie sięgnąć!

Rysunek 7.16 stanowi ilustrację działania procedury usuwania rekordu `Kowalski` — potrzebne modyfikacje wskaźników są zaznaczone linią pogrubioną.




**RYSUNEK 7.16.** Usuwanie danych ze środka listy dwukierunkowej

Oczywiście przypadki lewego skrajnego i prawego skrajnego elementu będą wymagać przestawienia wartości pól `glowa` i `ogon`.

W realnym kodzie Pythona pomożemy sobie metodą wyszukiwania rekordu:

```
def szukaj(self, pNazw): # Odszukaj rekord 'pNazw' na liście
    tmp = self.glowa     # Zmienna zapamiętująca status przeszukiwania listy
    znaleziono=False
    while tmp != None:
        if tmp.nazwisko==pNazw:
            znaleziono=True
            break                # Wychodzimy z pętli
        else:
```