W przypadku wywołania tego modułu jako skryptu wygeneruje on wynik prezentowany na listingu poniżej. Metoda __contains__ przechwytuje operację testu przynależności, metoda __iter__ obsługuje konteksty iteracyjne, w których wywoływana jest metoda __next__, natomiast metoda __getitem__ nie jest nigdy wywoływana.

```
contains: True
iter=> next:1 | next:2 | next:3 | next:4 | next:5 | next:
iter=> next:next:next:next:next:next:[1, 4, 9, 16, 25]
iter=> next:next:next:next:next:next:['Ob1', 'Ob10', 'Ob11', 'Ob100', 'Ob101']
iter=> next:1 @ next:2 @ next:3 @ next:4 @ next:5 @ next:
```

Sprawdźmy, co się stanie w przypadku, gdy zostanie zakomentowana metoda __contains__ — test przynależności jest realizowany przez metodę __iter__.

```
iter=> next:next:next:True
iter=> next:1 | next:2 | next:3 | next:4 | next:5 | next:
iter=> next:next:next:next:next:next:[1, 4, 9, 16, 25]
iter=> next:next:next:next:next:next:['Ob1', 'Ob10', 'Ob11', 'Ob100', 'Ob101']
iter=> next:1 @ next:2 @ next:3 @ next:4 @ next:5 @ next:
```

Na koniec przetestujmy wynik skryptu po ukryciu metod __contains__ i __iter__ — w takim przypadku wykorzystywana będzie metoda __getitem__, wywoływana z kolejnymi indeksami w kontekście testu przynależności oraz w iteracjach.

```
get[0]:get[1]:get[2]:True
get[0]:1 | get[1]:2 | get[2]:3 | get[3]:4 | get[4]:5 | get[5]:
get[0]:get[1]:get[2]:get[3]:get[4]:get[5]:[1, 4, 9, 16, 25]
get[0]:get[1]:get[2]:get[3]:get[4]:get[5]:['Ob1', 'Ob10', 'Ob11', 'Ob100','Ob101']
get[0]:1 @ get[1]:2 @ get[2]:3 @ get[3]:4 @ get[4]:5 @ get[5]:
```

Jak widać, metoda __getitem__ jest jeszcze bardziej ogólna: oprócz iteracji potrafi obsłużyć indeksowanie oraz tworzenie wycinków. Wyrażenia wycinające wywołują metodę __getitem__ z obiektem wycinka określającym zakresy. Mechanizm działa tak samo dla typów wbudowanych, jak i dla klas zdefiniowanych przez użytkownika, zatem tworzenie wycinków zadziała w naszej klasie bez dodatkowego kodu:

```
>>> X = Iters('spam')                  # Indeksowanie
>>> X[0]                               # __getitem__(0)
get[0]:'s'

>>> 'spam'[1:]                         # Składnia wycinania
'pam'
>>> 'spam'[slice(1, None)]             # Obiekt wycinka
'pam'

>>> X[1:]                              # __getitem__(slice(..))
get[slice(1, None, None)]:'pam'
>>> X[:-1]
get[slice(None, -1, None)]:'spa'
```

W bardziej realistycznych sytuacjach związanych z iteratorami, które nie wykorzystują sekwencji, użycie metody __iter__ może być łatwiejsze, ponieważ nie wymaga obsługi indeksu, natomiast metoda __contains__ pozwala w niektórych przypadkach znacznie zoptymalizować testy przynależności.