```
        tests = 0, 1                        # Nie działają: -1, 1.23
        tests += 12, 123, 1234, 12345, 123456, 1234567
        tests += 2 ** 32, 2 ** 100
        for test in tests:
            print(commas(test))

        print('')
        tests = 0, 1, -1, 1.23, 1., 1.2, 3.14159
        tests += 12.34, 12.344, 12.345, 12.346
        tests += 2 ** 32, (2 ** 32 + .2345)
        tests += 1.2345, 1.2, 0.2345
        tests += -1.2345, -1.2, -0.2345
        tests += -(2 ** 32), -(2**32 + .2345)
        tests += (2 ** 100), -(2 ** 100)
        for test in tests:
            print('%s [%s]' % (money(test, 17), test))

    import sys
    if len(sys.argv) == 1:
        selftest()
    else:
        print(money(float(sys.argv[1]), int(sys.argv[2])))
```

Powyższy plik działa tak samo w Pythonie 2.6 oraz 3.0. Po wykonaniu w sposób bezpośredni sprawdza samego siebie jak poprzednio, tym razem jednak wykorzystując opcje z wiersza poleceń w celu kontrolowania działania testu. By przekonać się, co wyświetla kod testu samosprawdzającego, wystarczy wykonać powyższy plik w sposób bezpośredni bez podawania własnych argumentów wiersza poleceń. W celu przetestowania określonych łańcuchów znaków należy je przekazać w wierszu poleceń wraz z minimalną szerokością pola:

```
C:\misc> python formats.py 999999999 0
$999,999,999.00

C:\misc> python formats.py -999999999 0
$-999,999,999.00

C:\misc> python formats.py 1234567890123456 0
$1,234,567,890,123,456.00

C:\misc> python formats.py -1234567890123456 25
$-1,234,567,890,123,456.00

C:\misc> python formats.py 123.456 0
$123.46

C:\misc> python formats.py -123.454 0
$-123.45

C:\misc> python formats.py
...gotowe testy - do wypróbowania samodzielnie...
```

Tak jak wcześniej, ponieważ kod ten wykorzystywany jest w podwójnym trybie użycia, możemy normalnie zaimportować jego narzędzia w innych kontekstach w postaci komponentów biblioteki:

```
>>> from formats import money, commas
>>> money(123.456)
'$123.46'
>>> money(-9999999.99, 15)
'$ -9,999,999.99'
>>> X = 99999999999999999999
>>> '%s (%s)' % (commas(X), X)
'99,999,999,999,999,999,999 (99999999999999999999)'
```