Example for syntax highlight with Pygments

Translate this document to HTML with a pygments enhanced frontend:

```
rst2html-pygments --stylesheet=pygments-default.css
```

or to LaTeX with:

```
rst2latex-pygments --stylesheet=pygments-default.sty
```

to gain syntax highlight in the output.

Convert between text <-> code source formats with:

```
pylit --code-block-marker='.. code-block:: python'
```

Run the doctests with:

```
pylit --doctest for-else-test.py
```

for-else-test

Test the flow in a *for* loop with *else* statement.

First define a simple *for* loop.

```
def loop1(iterable):
    """simple for loop with `else` statement"""
    for i in iterable:
        print i
    else:
        print "iterable empty"
    print "Ende"
```

Now test it:

The first test runs as I expect: iterator empty -> else clause applies:

```
>>> execfile('for-else-test.py')
>>> loop1(range(0))
iterable empty
Ende
```

However, the else clause even runs if the iterator is not empty in the first place but after it is "spent":

```
>>> loop1(range(3))
0
1
2
```

```
iterable empty
Ende
```

It seems like the else clause can only be prevented, if we break out of the loop. Let’s try

```
def loop2(iterable):
    """for loop with ‘break‘ and ‘else‘ statement"""
    for i in iterable:
        print i
        break
    else:
        print "iterable empty"
    print "Ende"
```

And indeed, the else clause is skipped after breaking out of the loop:

```
>>> loop2(range(3))
0
Ende
```

The empty iterator runs as expected:

```
>>> loop2(range(0))
iterable empty
Ende
```