```
>>> import math
>>> math.floor(2.5)
2
>>> math.floor(-2.5)
-3
>>> math.trunc(2.5)
2
>>> math.trunc(-2.5)
-2
```

Przy użyciu operatorów dzielenia tak naprawdę odcinamy resztę jedynie dla dodatnich wyników, ponieważ w takim przypadku odcięcie da nam samą podstawę. W przypadku liczb ujemnych uzyskujemy w rzeczywistości dzielenie bez reszty (tak naprawdę oba działania są dzieleniem bez reszty, a w przypadku liczb dodatnich dzielenie bez reszty jest równoznaczne z odcinaniem). Oto przykłady z wersji 3.0:

```
C:\misc> c:\python30\python
>>> 5 / 2, 5 / -2
(2.5, -2.5)

>>> 5 // 2, 5 // -2          # Odcina resztę do podstawy — zaokrągla do pierwszej mniejszej liczby całkowitej
(2, -3)                      # 2.5 staje się 2, −2.5 staje się −3

>>> 5 / 2.0, 5 / -2.0
(2.5, -2.5)

>>> 5 // 2.0, 5 // -2.0     # Tak samo w przypadku liczb zmiennoprzecinkowych, choć tu wynik jest także
                             # liczbą zmiennoprzecinkową
(2.0, -3.0)
```

W przypadku wersji 2.6 jest podobnie, jednak wyniki dla operatora `/` znowu będą różne:

```
C:\misc> c:\python26\python
>>> 5 / 2, 5 / -2           # Inaczej niż w 3.0
(2, -3)

>>> 5 // 2, 5 // -2         # Tu i poniżej wyniki w 2.6 i 3.0 są takie same
(2, -3)

>>> 5 / 2.0, 5 / -2.0
(2.5, -2.5)

>>> 5 // 2.0, 5 // -2.0
(2.0, -3.0)
```

Jeśli naprawdę potrzebne jest nam odcinanie, bez względu na znak, w każdej wersji Pythona możemy zawsze przekazać wynik dzielenia liczb zmiennoprzecinkowych do funkcji `math.trunc` (powiązaną funkcjonalność można znaleźć we wbudowanej funkcji `round`):

```
C:\misc> c:\python30\python
>>> import math
>>> 5 / -2                          # Zachowanie reszty
 2.5
>>> 5 // -2                         # Zaokrąglenie wyniku w dół
-3
>>> math.trunc(5 / -2)              # Odcięcie reszty zamiast zaokrąglenia w dół
 2

C:\misc> c:\python26\python
>>> import math
>>> 5 / float( 2)                    # Reszta w wersji 2.6
 2.5
```