Necklace

Operator overloading with custom types

If Python needs to evaluate the following expression

```
o1 + o2
```

it converts the expression into

```
type(o1).__add__(o1, o2)
```

This way, you can specify how the `+`-operator is evaluated if the object `o1` belongs to a custom type (defined using the `class` keyword). This is called *operator overloading*. However, operator overloading is not restricted to the `+`-operator. In fact, Python converts each built-in operator (like mathematical operators and comparison operators) into calling a method on the left operand `o1` whose name has been fixed by the Python developers (all names begins and ends with a double underscore). Here's an overview of some of these *magic* methods:

operator	method
`+`	`__add__`
`-`	`__sub__`
`*`	`__mul__`
`/`	`__truediv__`
`//`	`__floordiv__`
`**`	`__pow__`

Operator overloading initially converts the evaluation of an operator into calling a *magic* method on the left operand `o1`. But what if the class of the left operand `o1` does not define the magic method for object of type `o2`? In that case an exception is thrown, and Python makes a second attempt to call another *magic* method (whose name has an extra letter `r` in front) on the right operand `o2`.

For example, if the addition we observed above fails when calling the `__add__` method on the left operand `o1`, Python attempts to call the following method on the right operand `o2`

```
type(o2).__radd__(o2, o1)
```

Note that the name of the method has become `__radd__` instead of `__add__`, and that the order of the arguments has been inverted. This is important for asymmetric operations.

Returning a reference to the current object

Some objects return a reference to themselves after a change. As an example we implement the Tic-Tac-Toe game:

```
class TicTacToe:
  def __init__(self):
    self.grid = [
      [ None, None, None ],
      [ None, None, None ],
      [ None, None, None ]
    ]
    self.player = 'O'

  def play(self, i, j):
    self.grid[i][j] = self.player
    self.player = 'O' if self.player == 'X' else 'X'
    return self
```

This allows us to play the game as follows:

```
>>> game = TicTacToe().play(1, 1).play(0, 0).play(0, 1).play(1, 0)
>>> game.grid
[
  [ 'X', 'X', None ],
  [ 'O', 'O', None ],
  [ None, None, None ]
]
```

The important part here is the `return self`.

Data compression

Specific information

To decode a bitstring b you can execute the following procedure until the bitstring b is the empty string:

- find the shortest prefix p of the bitstring b that represents a symbol s that can be decoded
- add the symbol s to the decoded message
- remove the prefix p from the start of the bitstring b

Racetrack Playa

Specific information

Figure 1 shows what happens if a block slides.

Figure 1 shows what happens if a block tilts.

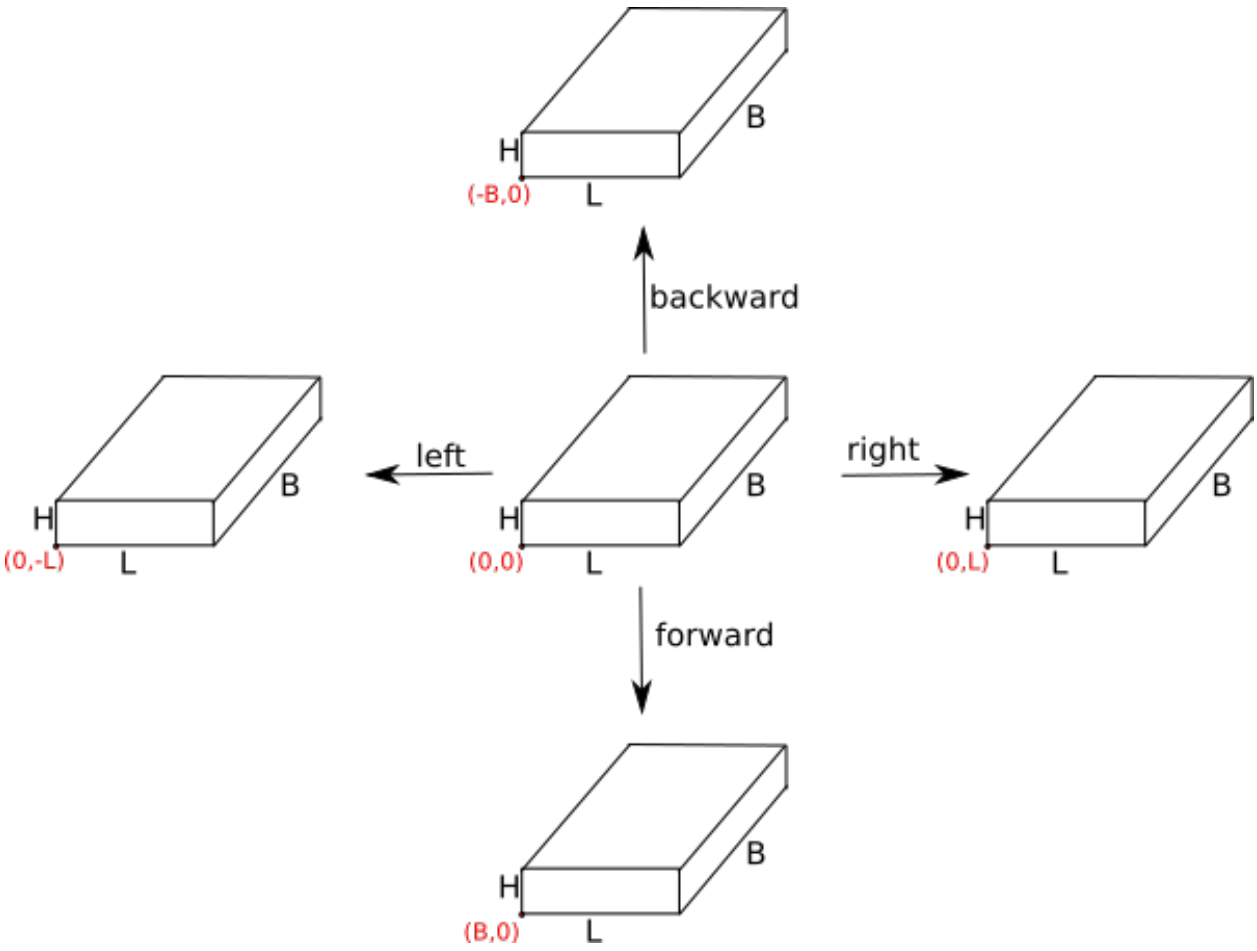

Figure 1: Slide

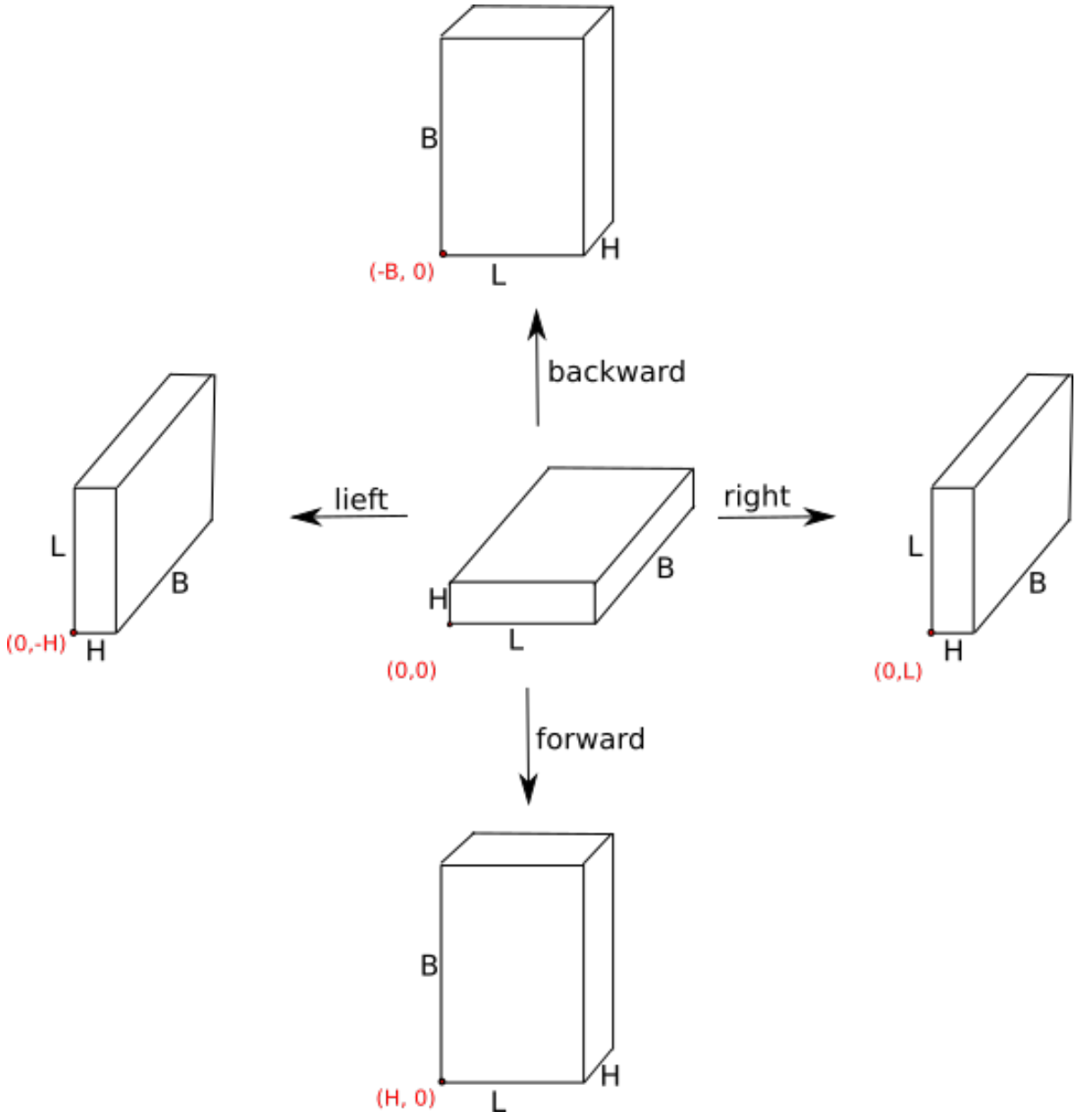

Figure 2: Tilt