Symbolic math in sympy

Sympy allows us to define variables that are just symbols, in the way they are used in mathematical formulas and equations. Sympy is then capable of manipulating expressions symbolically, rather than numerically (like numpy does). For example, if we want to evaluate the function $f(x) = \frac{x}{\pi}$ at $x = 2$, we would use numpy as follows:

```
import numpy as np
x=2
x/np.pi
Out:  0.6366197723675814
```

In sympy, we would instead define x to be a symbol, and $f(x)$ a symbolic expression. Then, we can evaluate the expression at a certain value of x using the `subs` function in sympy (which takes a dictionary as input). Note that this function does not affect the values of x and f. (They remain symbolic.)

```
import sympy as sp
x=sp.symbols('x')
f=x/sp.pi
f.subs({x:2})
Out:  2/pi
```

When assigning symbols, be aware of the difference between the variable x and the symbol x. You can think of the symbol as the value of the variable. For example, they do not have to match:

```
x=sp.symbols('a')
x/sp.pi
Out:  a/pi
```

Boolean variables

Boolean variables are variables that take values `True` or `False`. We can again define functions that take Boolean variables as input, and give a Boolean output. You can reuse symbols in different types of functions: x can represent a number in one function and a Boolean in another. You should know the following operators.

```
x,y=sp.symbols('x y')
~x
Out:  Not(x)
x&y
Out:  And(x, y)
x|y
Out:  Or(x, y)
x^y
Out:  Xor(x, y)
```

The symbols and written out functions work the same way, but the symbols are easier for us to use when programming, as they are easier to read and limit the number of parentheses. For example, compare:

```
(x&y)|(~x&~y&~(x|y))
Out:  Or(And(Not(Or(x, y)), Not(x), Not(y)), And(x, y))
```

We can use the `subs` function as before:

```
s=(x&y)|(~x&~y&~(x|y))
s.subs({x:True,y:False})
Out:  False
```

Some Boolean formulas are always `False`, no matter what the values of the input variables are. For example, the formula

```
x&~x
```

can never be satisfied. The `satisfiable` function in sympy's logic library tests whether a formula is satisfiable, and, if it is, returns a satisfying assignment of its variables. For example,

```
from sympy.logic.inference import satisfiable
satisfiable(s)
Out:   {x: False, y: False}
satisfiable(x&~x)
Out:   False
```

For the homework, you will be writing a function that gives all possible satisfying assignments, rather than just 1 (or 0). For this, a useful function is the `atoms()` function, which returns all the variables in a formula. (Since you need to know how many variables the formula has, and what they are called.)

```
list(s.atoms())
Out:   [x, y]
```

Exercises

- Find a Boolean formula that uses more than 2 variables, and is unsatisfiable.
- Find a Boolean formula that is satisfied **only** by `{x:True,y:True,z:True}`.
- Find a Boolean formula that uses variables x, y, z and is satisfied by any of the 2^3 possible assignments.
- Find a Boolean formula that uses variables x, y, z and is False if any of the variables are True.