In [1]: ▶
```python
import nltk
lp = nltk.sem.Expression.fromstring
```

In [7]: ▶
```python
#a
p='be given war and peace'
q='pat'
lp('p->q')
```

Out[7]: <ImpExpression (p -> q)>

In [76]: ▶
```python
lp('\\x. (if (exists p. given(p,x)) and then(exists q. pat(q,x))) ')
```

Out[76]: <LambdaExpression \x.(if(exists p.given(p,x)) & then(exists q.pat(q,x)))>

In [65]: ▶
```python
#b
p='everyone loves pat'
q='mathew loves pat'
lp('-p & q')
```

Out[65]: <AndExpression (-p & q)>

In [66]: ▶
```python
lp('\\x. -exists p. loves(p,x) and exists q. matthew(q,x)')
```

Out[66]: <AndExpression (\x.-exists p.loves(p,x) & exists q.matthew(q,x))>

In [8]: ▶
```python
#c
p='it will snow'
q='it will rain'
lp('-q->p')
```

Out[8]: <ImpExpression (-q -> p)>

In [75]: ▶
```python
lp('\\x.(if (-exists q. rain(q,x))and then (exists p. snow(p,x)))')
```

Out[75]: <LambdaExpression \x.(if(-exists q.rain(q,x)) & then(exists p.snow(p,x)))>

In [15]: ▶
```python
#d
p='loved by everyone'
q='detested by everyone'
lp('p&-q')
```

Out[15]: <AndExpression (p & -q)>

In [62]: ▶
```python
lp('\\x. (all p. love(p,x) & -exists q. detested(q,x))')
```

Out[62]: <LambdaExpression \x.(all p.love(p,x) & -exists q.detested(q,x))>

In [78]: ▶
```python
#e
p='Pat cough'
q='pat sneeze'
lp('-p&-q')
```

Out[78]: <AndExpression (-p & -q)>

In [29]: ▶
```python
lp('\\x(-all p. cough(p,x) and -all q. sneeze(q,x))')
```

Out[29]: <LambdaExpression \x.(-all p.cough(p,x) & -all q.sneeze(q,x))>

In [18]: ▶
```python
#f
p='i come'
q='you call'
s='you come'
t='i call'
lp('-(t->-s)->(q->-p)')
```

Out[18]: <ImpExpression (-(t -> -s) -> (q -> -p))>

In [77]: ▶| `'\\x (- (exists t. icall(t,x) & -exists s. youcome(s,x)) and  (exists q. youcall(q,x) and -exists p. icome(p,x)))')`

Out[77]: `<LambdaExpression \x.(-(exists t.icall(t,x) & -exists s.youcome(s,x)) & exists q.youcall(q,x) & -exists p.icome(p,`
          `x))>`

In [ ]: ▶|