Lea

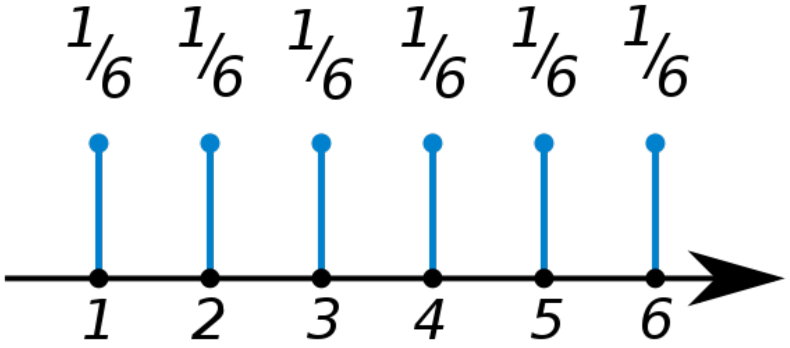
A probability engine in Python

FOSDEM 15

What is Lea?

Finite
discrete
probability
distributions

… in Python !



Yet another stats package?

No.

discrete distributions *only*

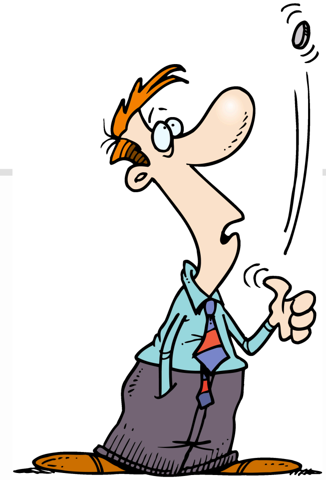
support of any object

integer arithmetic (no float)

calculus on probability distributions

emphasis on ease-of-use

Let's meet Lea!

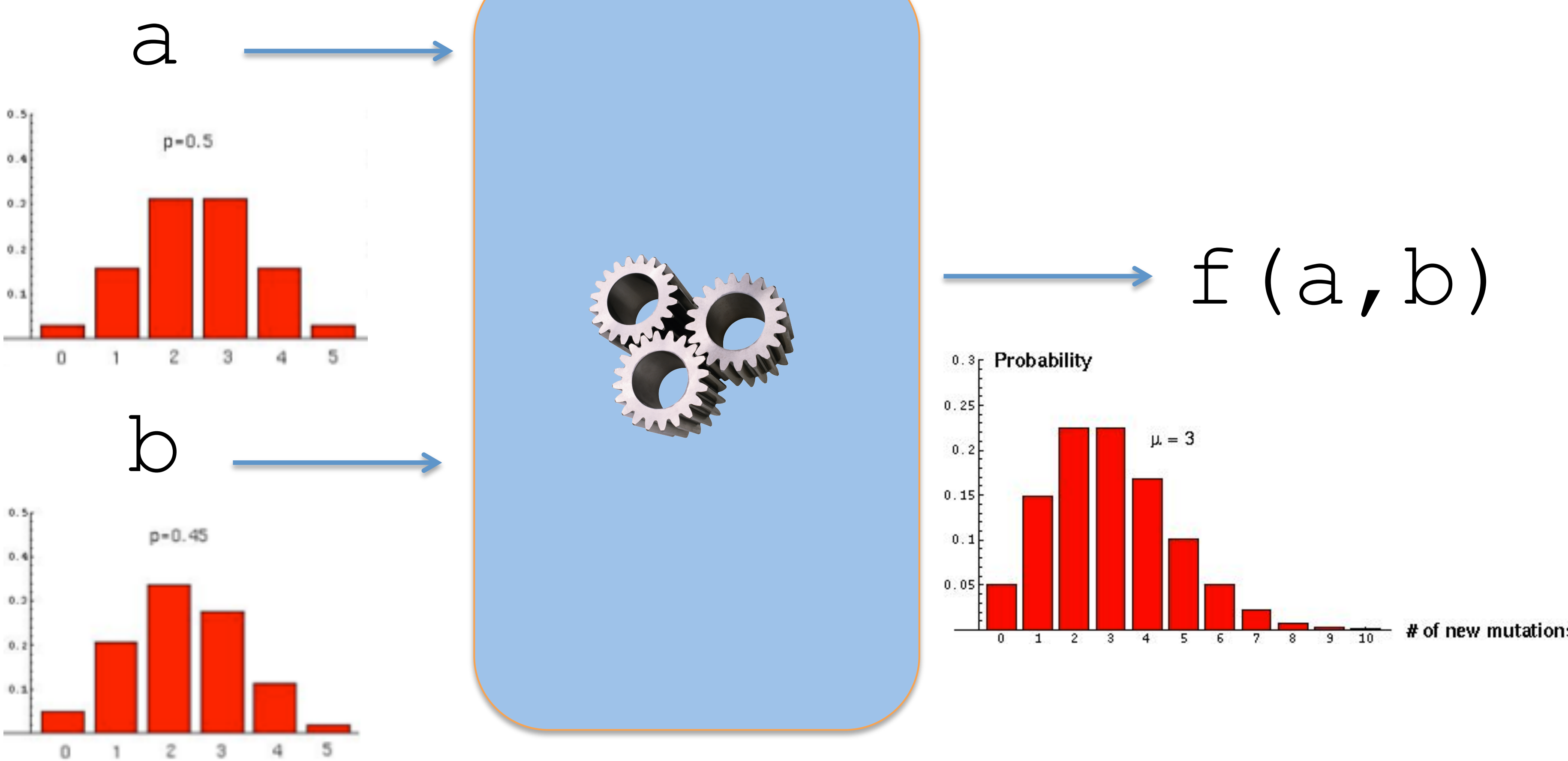


```
>>> flip = Lea.fromVals('Head','Tail')
>>> flip
Head : 1/2
Tail : 1/2

>>> biasFlip = Lea.fromVals('Head','Tail','Tail')
>>> biasFlip
Head : 1/3
Tail : 2/3

>>> biasFlip.random(10)
('Tail', 'Tail', 'Head', 'Tail', 'Head', 'Tail',
'Tail', 'Head', 'Head', 'Tail')

>>> Lea.fromVals(*(biasFlip.random(3000000)))
Head :  999407/3000000
Tail : 2000593/3000000
```

Play with a die

```
>>> die1 = Lea.fromVals(1,2,3,4,5,6)

>>> die1
1 : 1/6
2 : 1/6
3 : 1/6
4 : 1/6
5 : 1/6
6 : 1/6

>>> die1.mean
3.5

>>> die1 % 2
0 : 1/2
1 : 1/2
```

What is *calculus* on probability distributions?



Play with two dice

```
>>> die2 = die1.clone()

>>> dice = die1 + die2

>>> dice
 2 : 1/36
 3 : 2/36
 4 : 3/36
 5 : 4/36
 6 : 5/36
 7 : 6/36
 8 : 5/36
 9 : 4/36
10 : 3/36
11 : 2/36
12 : 1/36
```

Replay with two dice

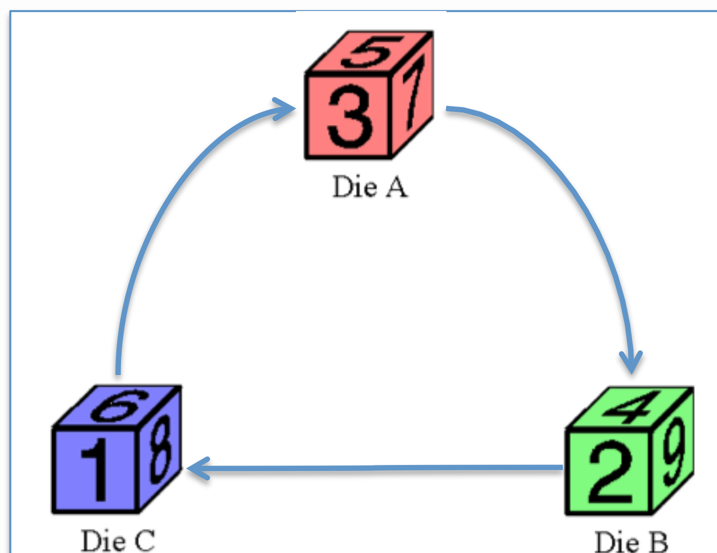
```
>>> dice <= 3
False : 11/12
 True :  1/12

>>> (dice <= 3).p(True)
1/12

>>> dice.map(lambda x: 'odd' if x%2 else 'even')
even : 1/2
 odd : 1/2
```

Conditional probabilities

```
>>> dice.given(dice <= 4)
2 : 1/6
3 : 2/6
4 : 3/6

>>> (dice <= 3).given(die1 <= 2)
False : 3/4
 True : 1/4
```

Play with nontransitive dice


```
>>> dieA = Lea.fromVals(3,3,5,5,7,7)
>>> dieB = Lea.fromVals(2,2,4,4,9,9)
>>> dieC = Lea.fromVals(1,1,6,6,8,8)
>>> dieA > dieB
False : 4/9
 True : 5/9
>>> dieB > dieC
False : 4/9
 True : 5/9
>>> dieC > dieA
False : 4/9
 True : 5/9
```



Meeting Leapp

Small “PPL” (Probabilistic Programming Language)

Easy / Concise

Probabilities expressed as

- fractions,
- percentages,
- decimals

From Leapp to Python

```
?{'H': 34.5%, 'T': 65.5%}
```

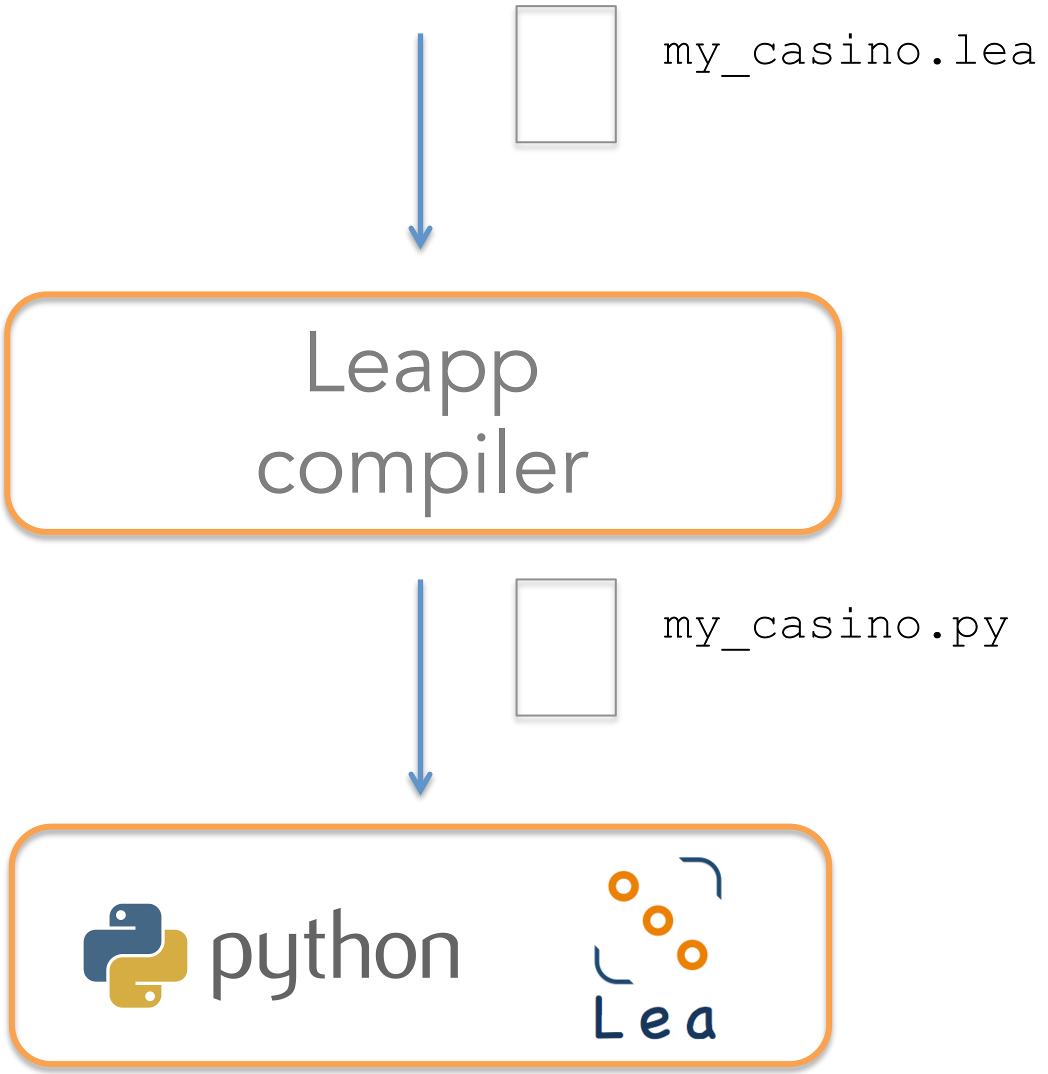
Leapp interpreter

```
Lea.fromValFreqs(('H',69),('T',131))
```

python

Lea

Compile `.lea` files



C A S E II^{d}.

To find the Probability of throwing an Ace in three throws.

SOLUTION.

The Probability of throwing an Ace the firſt time is $\frac{1}{6}$, which is the firſt part of the Probability required.

If the Ace be miſſed the firſt time, ſtill it may be thrown in the two remaining throws; but the Probability of miſſing it the firſt time is $\frac{5}{6}$, and the Probability of throwing it in the two remaining times is (by Caſe 1^{st}) $= \frac{11}{36}$. And therefore the Probability of miſſing it the firſt time, and throwing it in the two remaining times is $\frac{5}{6} \times \frac{11}{36} = \frac{55}{216}$, which is the ſecond part of the Probability required; wherefore the Probability required will be $\frac{1}{6} + \frac{55}{216} = \frac{91}{216}$.

The Doctrine of Chance, by A. de Moivre, 1718

P(Ace in 3 throws)

Leapp

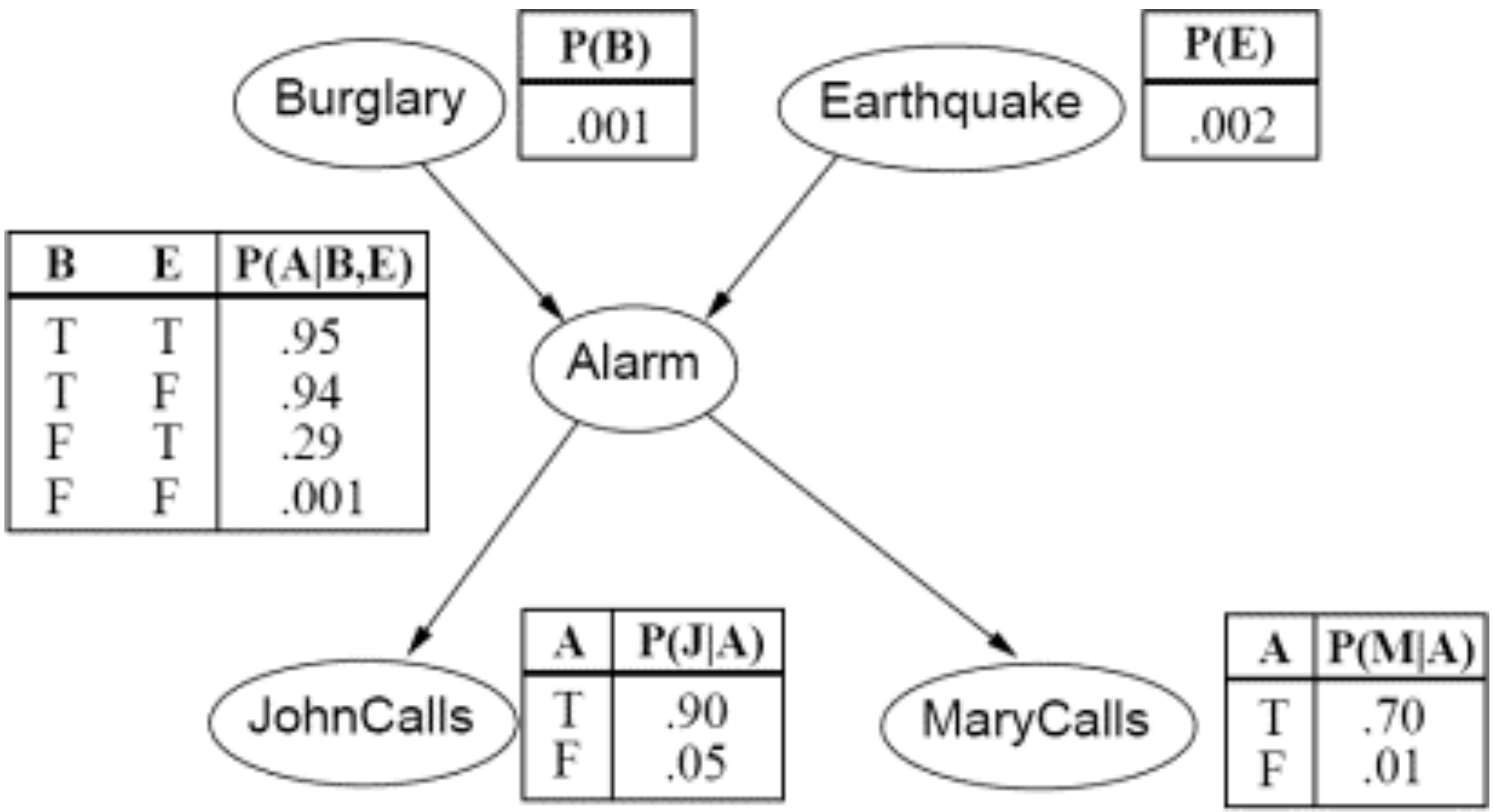
```
lea> die1 = ?(1,2,3,4,5,6)
lea> die2 = ?die1
lea> die3 = ?die1
lea> (die1==1) | (die2==1) | (die3==1)
False : 125/216
 True :  91/216


lea> from operator import or_
lea> ace = (die1==1)
lea> ?[3,or_]ace
False : 125/216
 True :  91/216
```

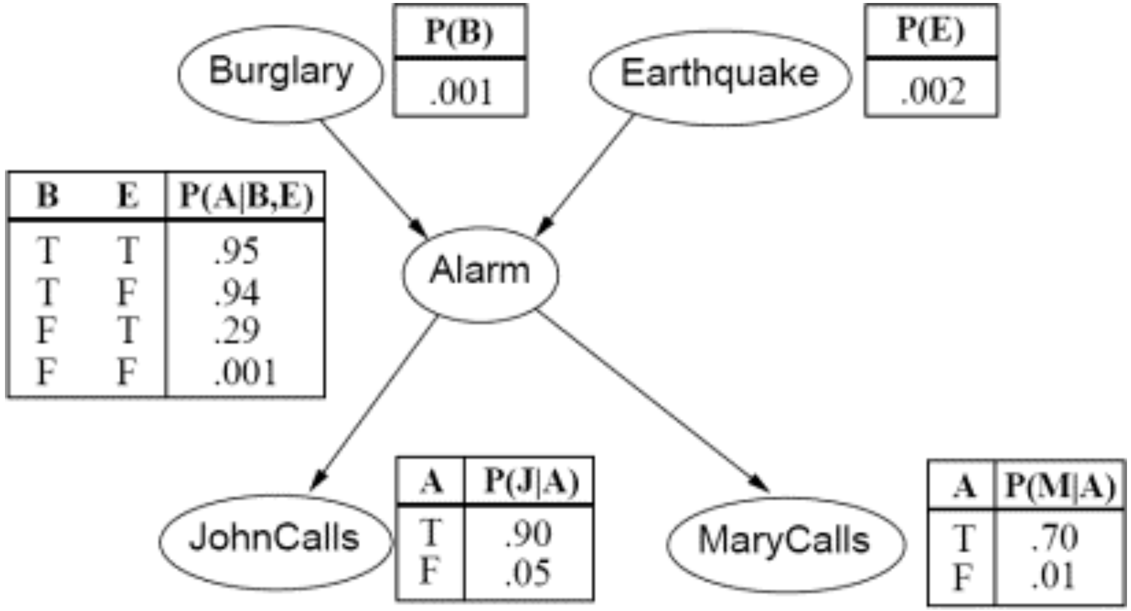
Card deck

```
lea> cards = ?('A23456789TJQK') + ?('♥♣♦♠')
lea> cards
2♠ : 1/52
2♣ : 1/52
2♥ : 1/52
2♦ : 1/52
3♠ : 1/52
…
lea> kind = cards[1]
lea> kind
♠ : 1/4
♣ : 1/4
♥ : 1/4
♦ : 1/4
lea> ' '.join(cards$_(13))
'T♠ J♠ A♠ Q♠ 4♣ 5♣ 7♥ 7♠ J♦ A♥ T♥ A♣ 6♠'
```

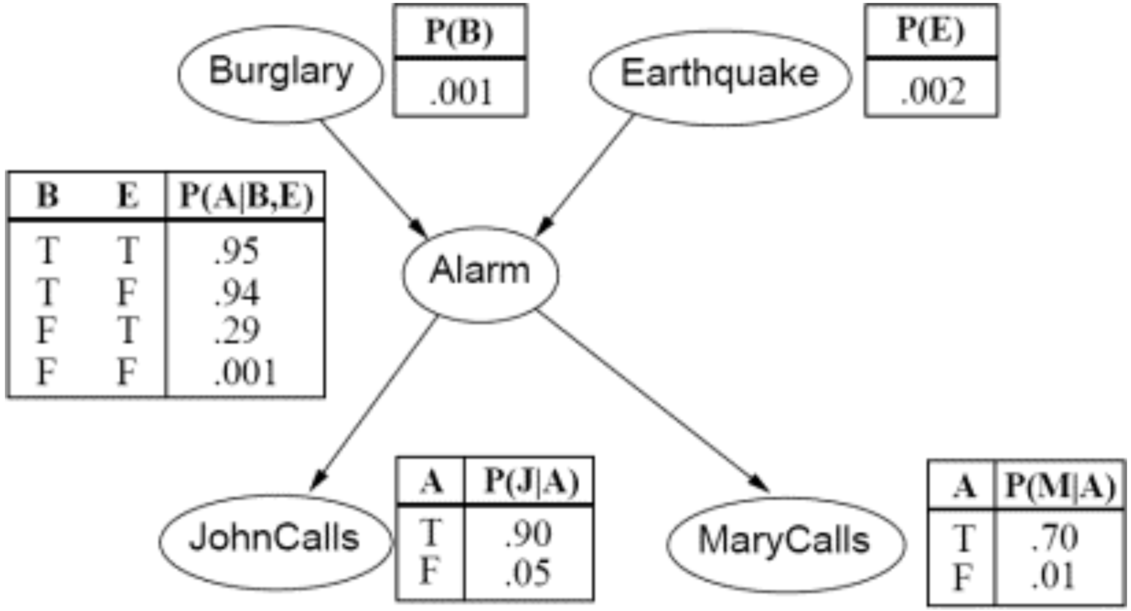
Random draw
without replacement

Bayesian networks






```
burglary    = ?:(.001)
earthquake = ?:(.002)
alarm = ?! (  burglary &  earthquake -> ?:(.950),
              burglary & ~earthquake -> ?:(.940),
             ~burglary &  earthquake -> ?:(.290),
             ~burglary & ~earthquake -> ?:(.001))
johnCalls = ?! (  alarm -> ?:(.900),
                 ~alarm -> ?:(.050))
maryCalls = ?! (  alarm -> ?:(.700),
                 ~alarm -> ?:(.010))
```



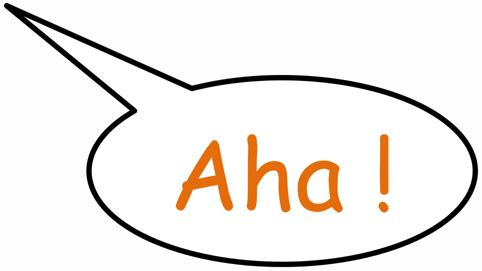
Leapp

```
lea> :.@ maryCalls ! alarm
0.7
lea> :.@ maryCalls ! burglary
0.6586138
lea> :.@ maryCalls
0.01173634498
lea> :.@ burglary ! maryCalls & johnCalls
0.28417183536439294
```

Experience in binding

```
lea> dice = die1 + die2
lea> dice
 2 : 1/36
 3 : 2/36
 4 : 3/36
 5 : 4/36
 6 : 5/36
 7 : 6/36
 8 : 5/36
 9 : 4/36
10 : 3/36
11 : 2/36
12 : 1/36
```

```
lea> dice - die1
1 : 1/6
2 : 1/6
3 : 1/6
4 : 1/6
5 : 1/6
6 : 1/6
```



The *statues* algorithm

```
class Lea(object):

    ...

    def genVPs(self):
        if self._val is not None:
            yield (self._val,1)
        else:
            for (v,p) in self._genVPs():
                self._val = v
                yield (v,p)
            self._val = None
```

Generate
(*value* , prob(*value*))


A bullshit generator using Lea
(snippets…)

Words

```
verb = TerminalNode( "accesses", "activates",
"administrates", "aggregates", "builds", "calculates",
"checks", "competes with",
…

aSimpleName = TerminalNode( "COTS", "GRID processing",
"Java program", "LDAP registry", "Portal", "RSS feed",
"SAML token", "SOAP message", "SSO", "TCP/IP", "UML",
…
```

Grammar rules

Probability weights

```
verbalGroup.setTermsChoices(
    (( verb, nameGroup                          ), 10 ),
    (( adverb, verb, nameGroup                  ),  1 ),
    (( "is", passiveVerb, nameGroup             ), 10 ),
    (( "is", adverb, passiveVerb, nameGroup ),  1 ),
    (( "is", adjective                          ),  1 ),
    (( "is", adverb, adjective                  ),  1 ))
```

A small bullshit sample…

```
The unaffected file validates the entity. As a matter of
fact, the used opportunity populates the environment. The
operational technique is processed by the thread-safe end-
point, which optimizes the UML model. The business model is
driven by the Portal value updated by the aspect.
Nevertheless, the presentation layer that manages the
authorized portal is serialized in the timing persistence
based upon a UML catalogue maximized by an owner action.
The interface retrieves the granularity. The message-based
processor natively mitigates the online acknowledgment.
Nevertheless, the style sheet market that encapsulates the
Java program is extracted from the interface, which shall
be the COTS, within the role.
…
```

Lea

http://code.google.com/p/lea → *doc / issues /…*

http://pypi.python.org/pypi/lea → *download*

author: Pierre Denis
pie.denis@skynet.be (English / French)

```
lea> 'Do you ' + ?('have questions','need a break') + '?'
Do you have questions? : 1/2
  Do you need a break? : 1/2
```