Unit 10 Assignment

**ANLY:520-51 (Fall 2016)**

Dean D’souza

H.U. ID: 168424

# **Solutions:**

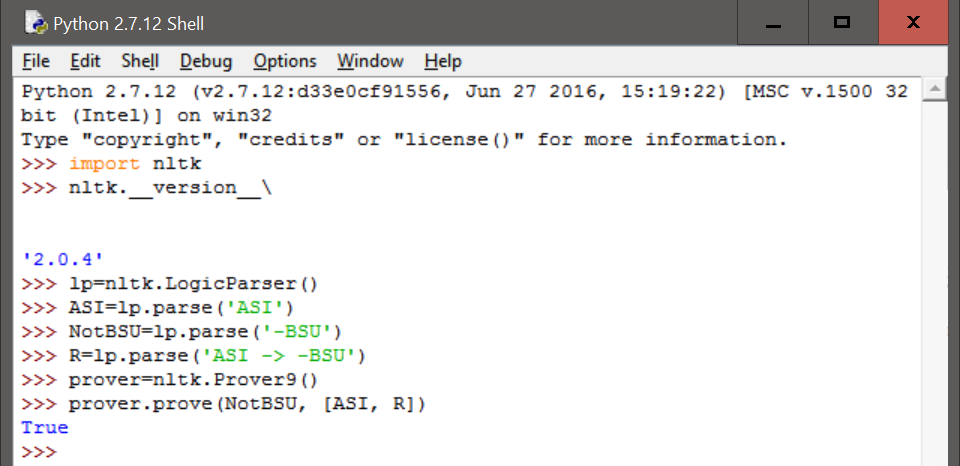
1. **Translate the following sentences into propositional logic and verify that they parse with LogicParser. Provide a key which shows how the propositional variables in your translation correspond to expressions of English.**

We provide the keys and screenshots as follows:

* 1. **If Angus sings, it is not the case that Bertie sulks.**

ASI : Angus Sings

BSU: Bertie Sulks

****

* 1. **Cyril runs and barks.**

CRU: Cyril Runs

CBA: Cyril barks

****

* 1. **It will snow if it doesn't rain.**

WSN: it will snow

NotWRN: it doesn’t rain

****

* 1. **It's not the case that Irene will be happy if Olive or Tofu comes.**

IWH: Irene will be happy

OOTC: Olive or Tofu comes

****

* 1. **Pat didn't cough or sneeze.**

PC: Pat coughs

PS: Pat sneezes

It should be noted here that the ‘|’ operator should give false as both are essentially false. So maybe the and operator should be used.



* 1. **If you don't come if I call, I won't come if you call.**

CIC: Come if I call

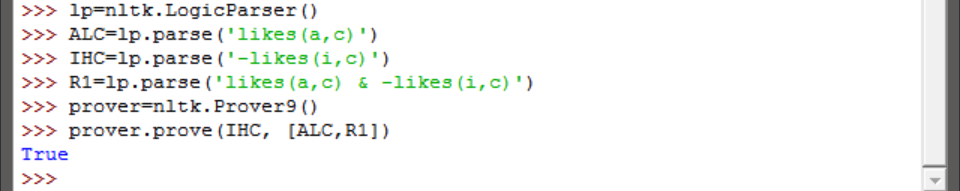
CYC: Come if You call



1. **Translate the following sentences into predicate-argument formula of first order logic.**

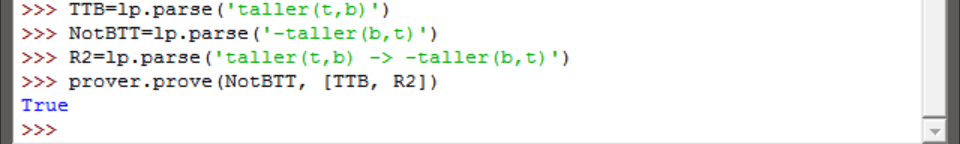
While we are not asked to show a result in python, screenshots for each, using the same method as in the previous question, are attached as follows:

* 1. **Angus likes Cyril and Irene hates Cyril.**

****

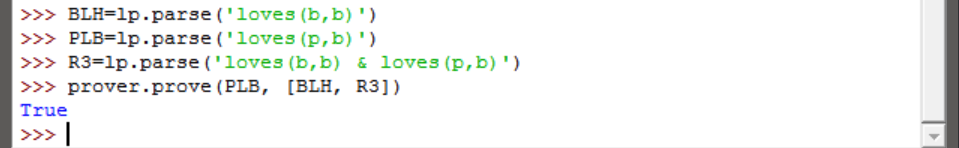
* 1. **Tofu is taller than Bertie.**

Here only the first line should suffice but additional inference was made.

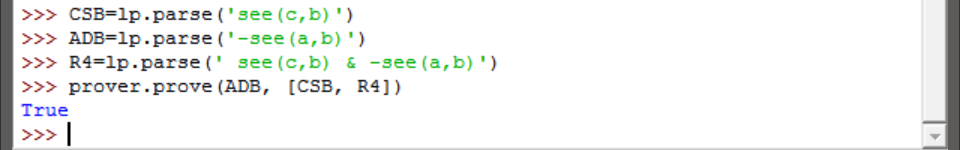
****

* 1. **Bruce loves himself and Pat does too.**

Here b stands for Bruce and p stands for Pat

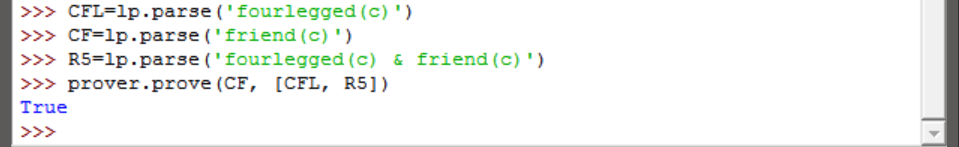
****

* 1. **Cyril saw Bertie, but Angus didn't.**

****

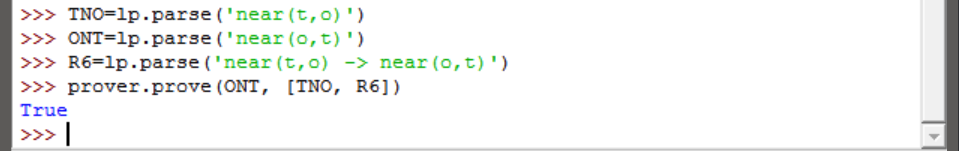
* 1. **Cyril is a four legged friend.**

Here, c stands for Cyril.

****

* 1. **Tofu and Olive are near each other.**

In this case as well additional inferences were made.



1. **Translate the following sentences into quantified formulas of first order logic.**

As we do not need to show the result in python (and since there are 9 sentences) we give the quantified formulas for each as follows:

* 1. **Angus likes someone and someone likes Julia.**

Person(x) -> exists x. (likes(Angus,x) & likes(x,Julia))

* 1. **Angus loves a dog who loves him.**

Dog(d) -> exists d. (loves(Angus,d) & loves(d,Angus))

* 1. **Nobody smiles at Pat.**

Person(x) & all x. (-smile(x,Pat))

* 1. **Somebody coughs and sneezes.**

Person(x) -> exists x. (cough(x) & sneeze(x))

* 1. **Nobody coughed or sneezed.**

Person(x) & all x. (-cough(x) & -sneeze(x))

* 1. **Bruce loves somebody other than Bruce.**

Person(x) -> exists x. (loves(Bruce,x) & -loves(Bruce,Bruce))

* 1. **Nobody other than Matthew loves somebody Pat.**

Person(y) & all x. ((x!=Matthew & y=Pat) -> -likes(x,y))

* 1. **Cyril likes everyone except for Irene.**

Person(y) & all y. ((y!=Irene) -> likes(Cyril,y))

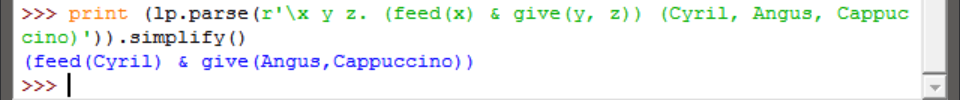
* 1. **Exactly one person is asleep.**

Person(y) & all y. ((y!=x) -> asleep(x))

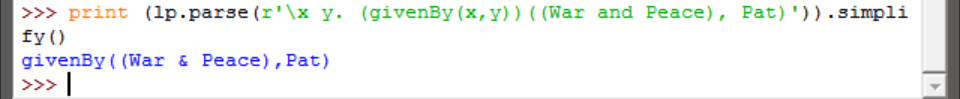
1. **Translate the following verb phrases using λ abstracts. quantified formulas of first order logic.**

The translations are as follows:

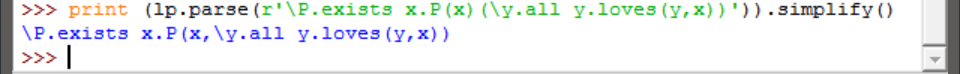
* 1. **feed Cyril and give a cappuccino to Angus**



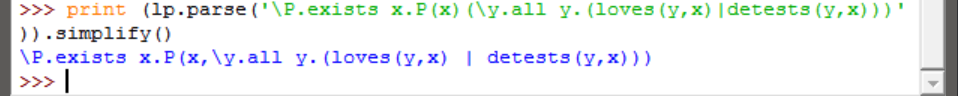
* 1. **be given 'War and Peace' by Pat**



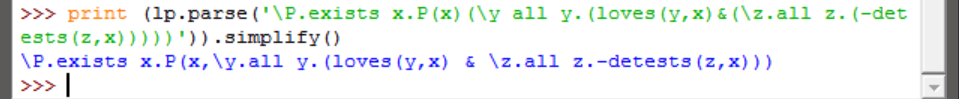
* 1. **be loved by everyone**



* 1. **be loved or detested by everyone**



* 1. **be loved by everyone and detested by no-one**

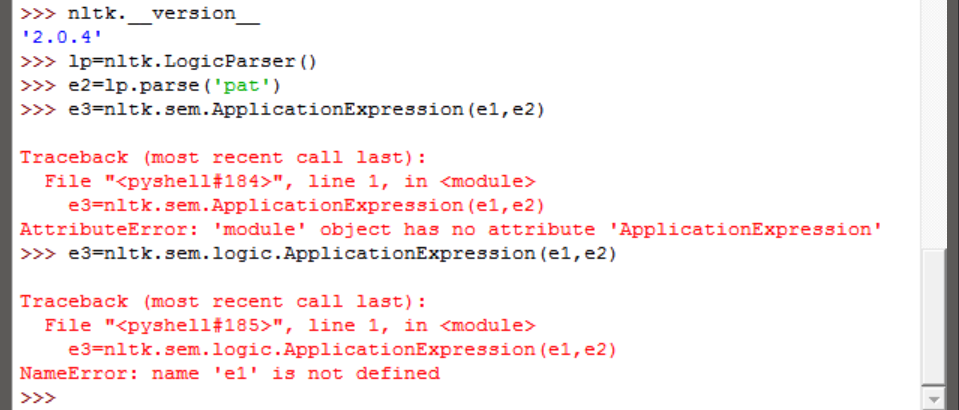


1. **Consider the following statements:**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> lp = nltk.LogicParser()**  **>>> e2 = lp.parse('pat')**  **>>> e3 = nltk.sem.ApplicationExpression(e1, e2)**  **>>> print e3.simplify()**  **exists y.love(pat, y)** | |

**Clearly something is missing here, namely a declaration of the value of e1. In order for ApplicationExpression(e1, e2) to be β-convertible to exists y.love(pat, y), e1 must be a λ-abstract which can take pat as an argument. Your task is to construct such an abstract, bind it to e1, and satisfy yourself that the statements above are all satisfied (up to alphabetic variance). In addition, provide an informal English translation of e3.simplify().**

We verify this as follows:



As we can see above, the ApplicationExpression() function could not be found directly under sem for reason. We proceed as follows:

**Now carry on doing this same task for the further cases of e3.simplify() shown below.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> print e3.simplify()**  **exists y.(love(pat,y) | love(y,pat))** | |
| |  |  | | --- | --- | |  | **>>> print e3.simplify()**  **exists y.(love(pat,y) | love(y,pat))** | |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> print e3.simplify()**  **walk(fido)** | |

There are two of the same case in the above question and hence there are only two solutions required, which are as follows:

