CSE 5120 Homework Assignment 4 – Fall 2021

Instructor: Dr. Kerstin Voigt

Total number of points: 20

This assignment asks you to implement an interactive theorem prover that proves a propositional sentence with resolution refutation. This means that the negation of the sentence to be proved is added to a set of premise sentences, and via a sequence of resolutions of pairs of clauses, a contradiction, in form on an emerging empty clause, is revealed.

The lecture presented a program which fully automated the resolution refutation process. It was up to the program to determine suitable clauses for each set of resolution. Sometimes, the most effective application is one where human and machine collaborate to jointly arrive at a desired outcome. An interactive theorem prover is such a program, and you are to implement one.

Based in the lecture, we already have a good amount of Python code to use. Here is some code that will help with you with the assignment … understand the parts and put the pieces together.

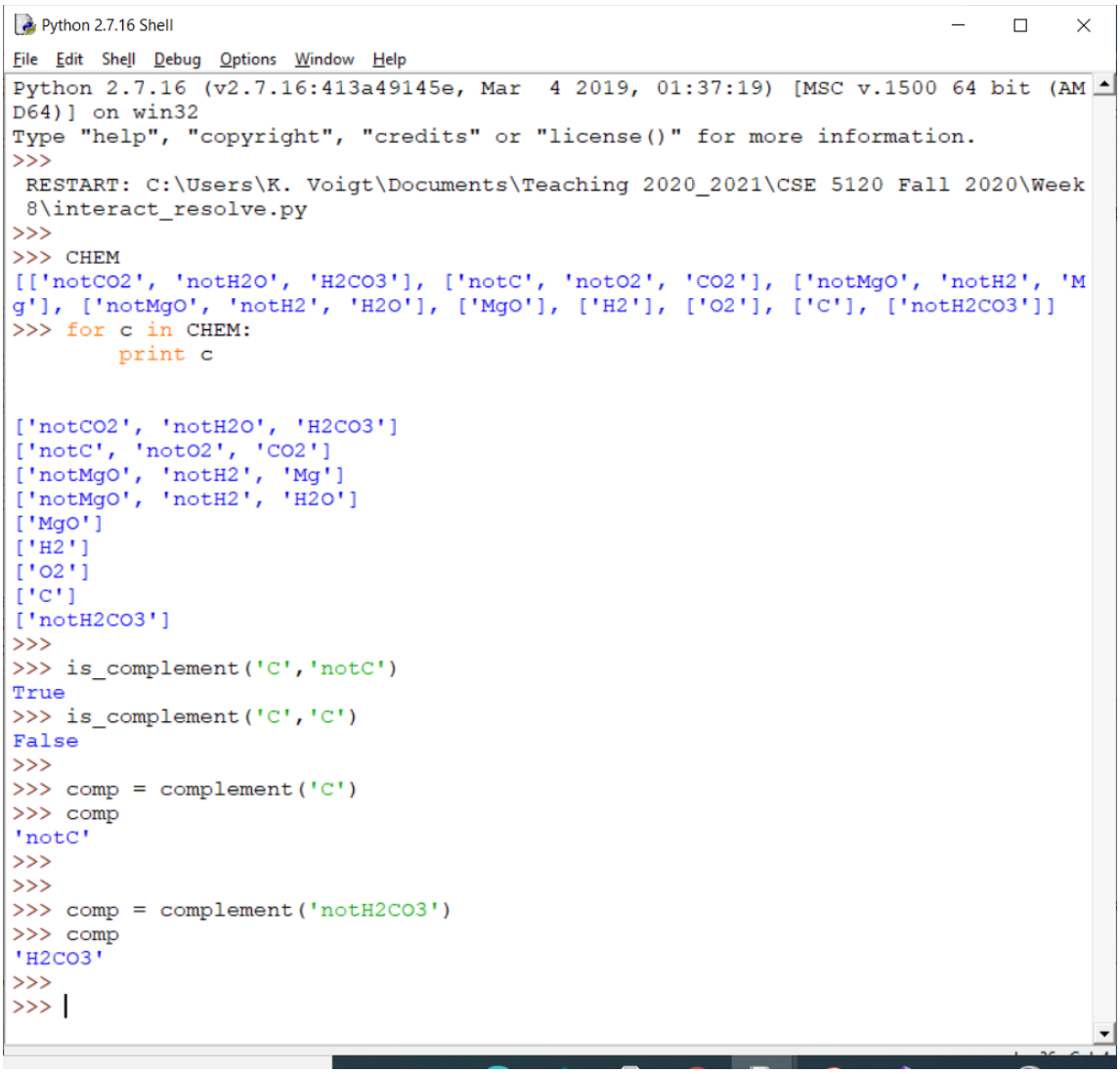
```
CLS = [['notP', 'notQ', 'R'], ['P', 'R'], ['Q', 'R'], ['notR']]

CHEM = [['notCO2', 'notH2O','H2CO3'], ['notC', 'notO2', 'CO2'],\
        ['notMgO', 'notH2', 'Mg'], ['notMgO', 'notH2','H2O'],
        ['MgO'], ['H2'], ['O2'], ['C'], ['notH2CO3']]

GOV = [['T', 'notE', 'D'], ['notT', 'C'], ['E','notD', 'I'],\
       ['notG', 'notD', 'I'], ['T', 'C', 'notD', 'G'], ['notC'],\
       ['notI', 'notG'], ['D'], ['E']]



def try_resolvent(count,c1, c2):
    res = None
    for lit1 in c1:
        for lit2 in c2:
            if is_complement(lit1,lit2):
                print "\n[%d.] Resolving %s and %s ..." % (count,c1,c2)
                print "... found compl lits (%s,%s)" % (lit1,lit2)
                res = c1[:]
                res.remove(lit1)
                for x in c2:
                    if not x in res and x != lit2:
                        if complement(x) in res:
                            return True
                        else:
                            res.append(x)
                return res
    print "No resolvent"
    return res
```

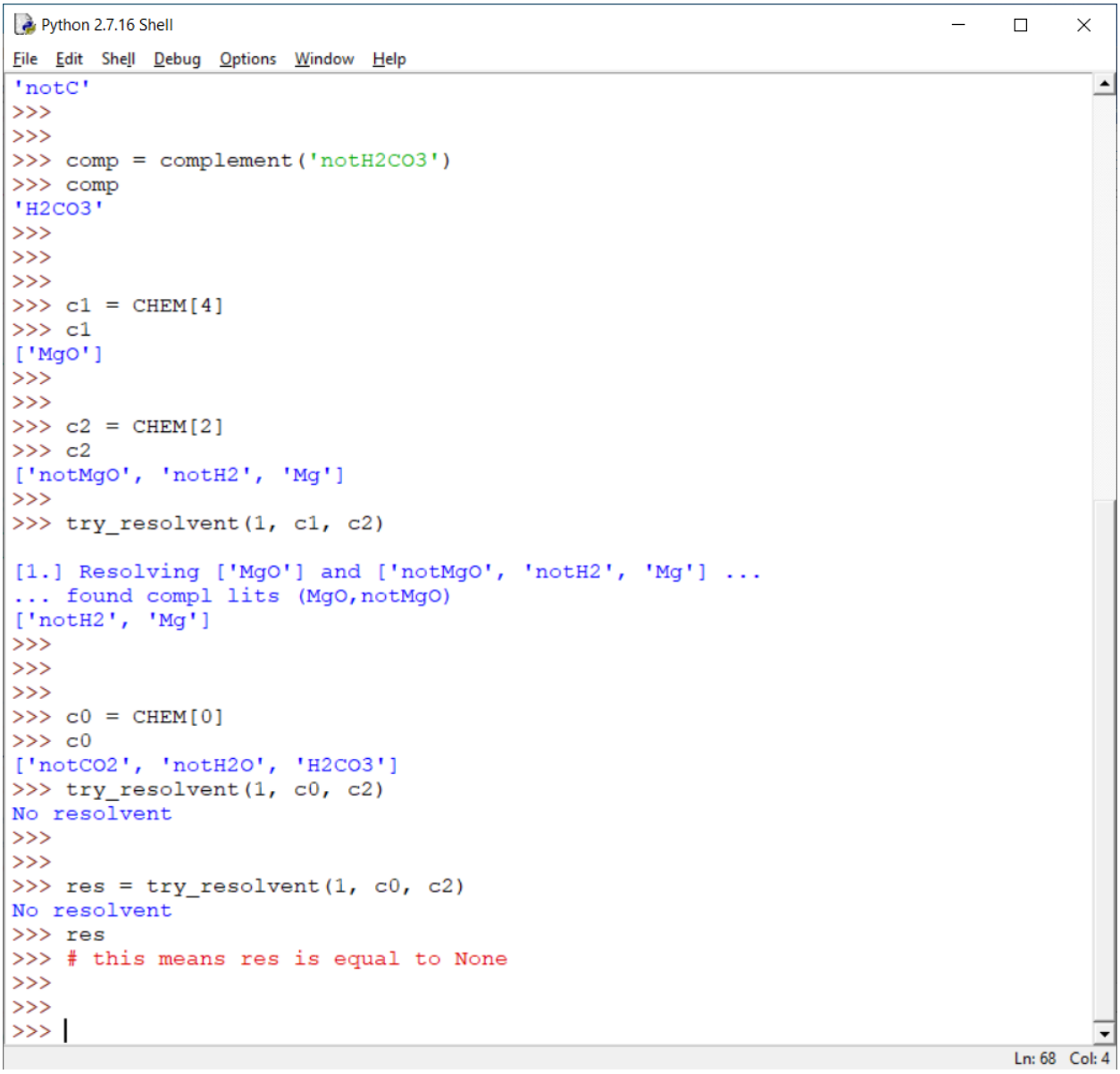
```
def is_complement(x,y):
    if len(x) >= 4 and x[:3] == 'not':
        return x[3:] == y
    elif len(y) >= 4 and y[:3] == 'not':
        return x == y[3:]
    else:
        return False


def complement(x):
    if len(x) >=4 and x[:3] == 'not':
        return x[3:]
    else:
        return 'not'+x


# c1 same as c2, if both contain the same elements, but possibly
# in a different order; iterate over one and test whether it is
# present in the other;

def same_clause(c1,c2):
    if not len(c1) == len(c2):
        return False
    for x in c1:
        if not x in c2:
            return False
    return True
```

Global variables CLS, CHEM and GOV are examples of sets of clauses that can be used for the testing of the theorem prover. All three sets of clauses are already the products of having converted sets of premises into conjunctive-normal form CNF (i.e., clauses) and added to these clause the result of the CNF-form of the negated statement to prove. Example, the clause [notH2CO3], which is the negation of H2CO3, is already included in the set of clauses CHEM.

Function try_resolvent(count, c1, c2) returns one of three possible outputs: (1) the resolvent clause of c1 and c2, if one exists, including the empty clause (), count could the running number for the next resolution in the process; (2) value True if the resolving yields an irrelevant tautology, or (3) otherwise. Three utility functions are also given; is_complement() and complement() are called in try_resolvent(), same_clauses() may turn out useful for this assignment (check it out, read the comment).

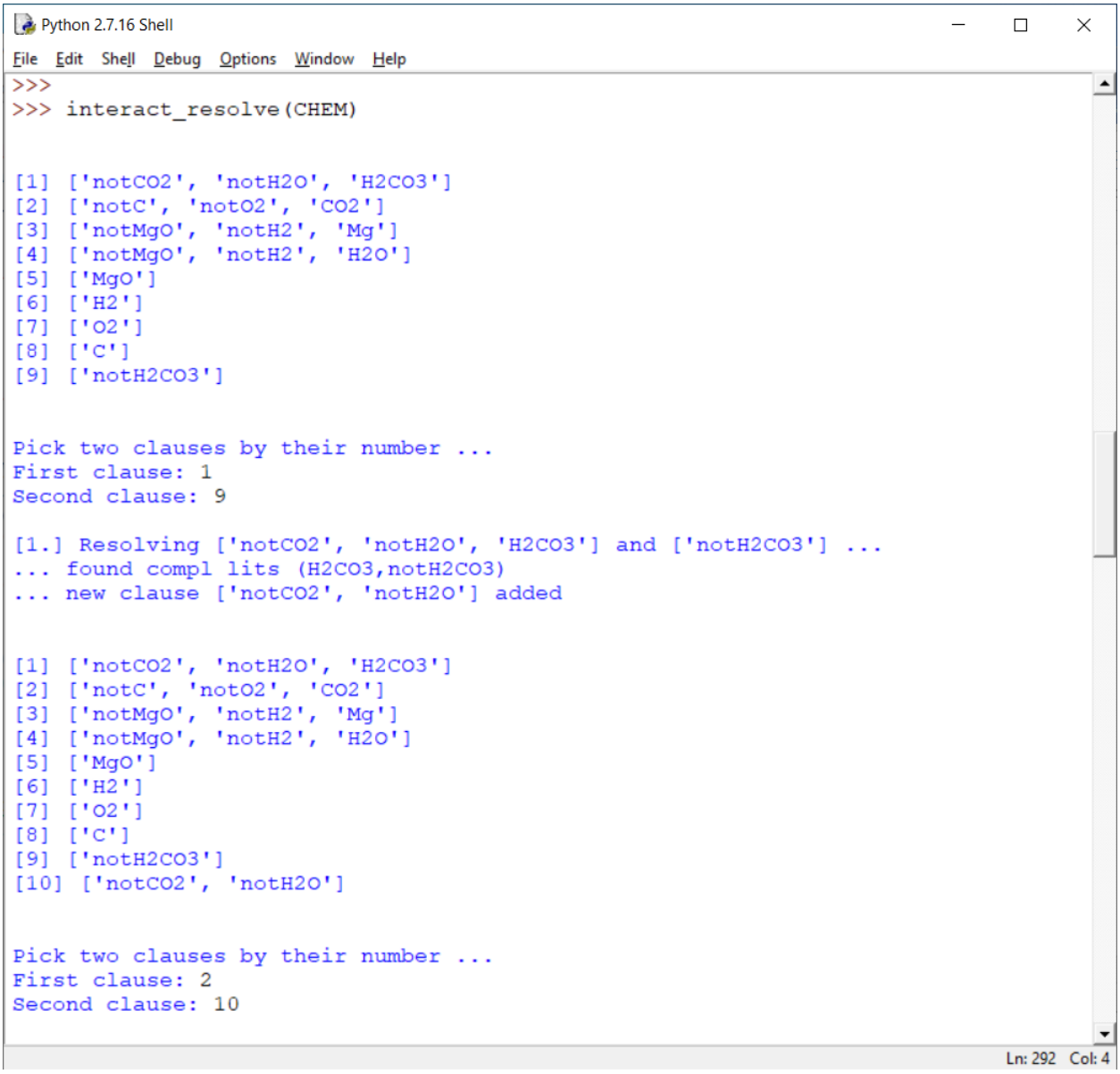
Familiarize yourself with the functions by writing the above into a .py file, loading it, and experimenting ….

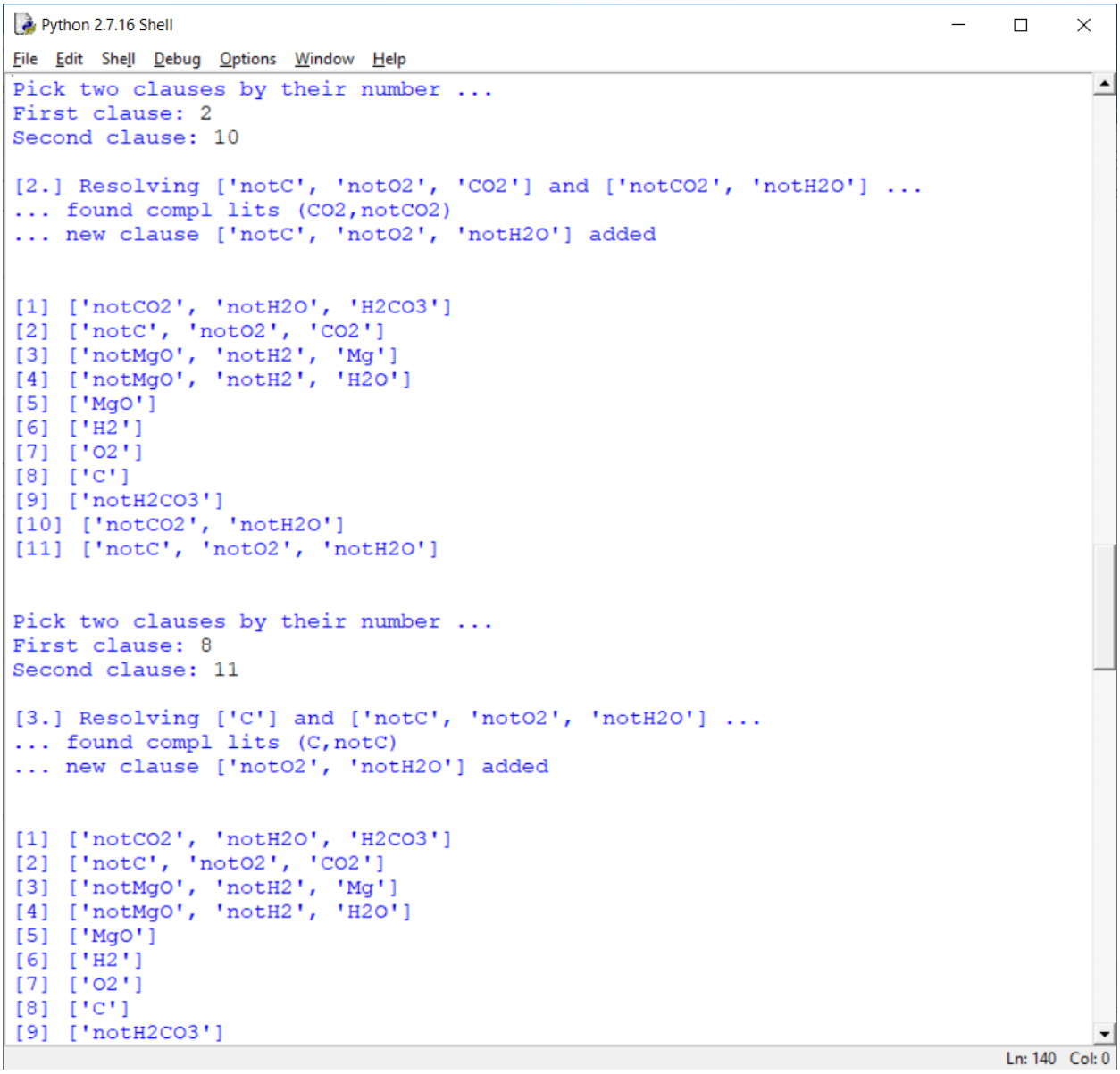

Python 2.7.16 Shell
File Edit Shell Debug Options Window Help
Python 2.7.16 (v2.7.16:413a49145e, Mar 4 2019, 01:37:19) [MSC v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\K. Voigt\Documents\Teaching 2020_2021\CSE 5120 Fall 2020\Week 8\interact_resolve.py
>>>
>>> CHEM
[['notCO2', 'notH2O', 'H2CO3'], ['notC', 'notO2', 'CO2'], ['notMgO', 'notH2', 'Mg'], ['notMgO', 'notH2', 'H2O'], ['MgO'], ['H2'], ['O2'], ['C'], ['notH2CO3']]
>>> for c in CHEM:
print c
['notCO2', 'notH2O', 'H2CO3']
['notC', 'notO2', 'CO2']
['notMgO', 'notH2', 'Mg']
['notMgO', 'notH2', 'H2O']
['MgO']
['H2']
['O2']
['C']
['notH2CO3']
>>>
>>> is_complement('C','notC')
True
>>> is_complement('C','C')
False
>>>
>>> comp = complement('C')
>>> comp
'notC'
>>>
>>>
>>> comp = complement('notH2CO3')
>>> comp
'H2CO3'
>>>
>>>



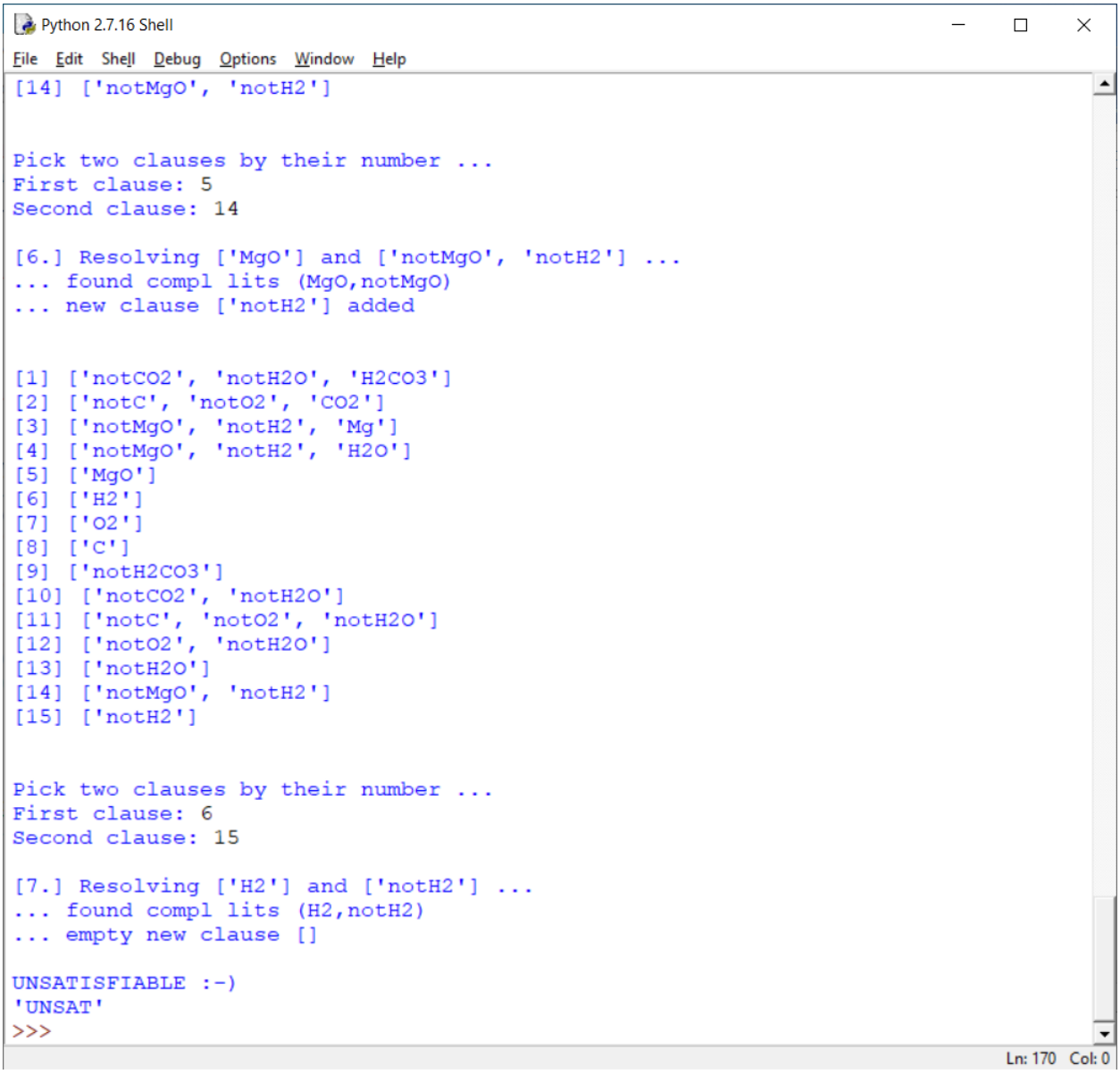
Making use of these functions, write your own interactive resolution-based theorem proving program so that the following type and interaction can take place between the human user and the program:

… see next page ..


Python 2.7.16 Shell
File Edit Shell Debug Options Window Help
>>>
>>> interact_resolve(CHEM)
[1] ['notCO2', 'notH2O', 'H2CO3']
[2] ['notC', 'notO2', 'CO2']
[3] ['notMgO', 'notH2', 'Mg']
[4] ['notMgO', 'notH2', 'H2O']
[5] ['MgO']
[6] ['H2']
[7] ['O2']
[8] ['C']
[9] ['notH2CO3']
Pick two clauses by their number ...
First clause: 1
Second clause: 9
[1.] Resolving ['notCO2', 'notH2O', 'H2CO3'] and ['notH2CO3'] ...
... found compl lits (H2CO3,notH2CO3)
... new clause ['notCO2', 'notH2O'] added
[1] ['notCO2', 'notH2O', 'H2CO3']
[2] ['notC', 'notO2', 'CO2']
[3] ['notMgO', 'notH2', 'Mg']
[4] ['notMgO', 'notH2', 'H2O']
[5] ['MgO']
[6] ['H2']
[7] ['O2']
[8] ['C']
[9] ['notH2CO3']
[10] ['notCO2', 'notH2O']
Pick two clauses by their number ...
First clause: 2
Second clause: 10
Ln: 292 Col: 4



… and several more interactions later (human could have selected better …)



Test your interactive program by running all three test examples, CHEM (the one shown, but make your own, and possibly better selections), CLS (a smaller example, from the textbook), and GOV (a bit harder than CHEM but doable).

Submit by Thursday, 10/28, by 11:59 pm: Via Blackboard portal. (1) A copy of your .py file (e.g., interact_resolve.py), (2) three Word files (or equivalent) with each showing 3 screenshots of successful human-computer interactive theorem proving for the CHEM, CLS, and GOV examples respectively. (You may do what I have done for this document: open up a Word file and paste

into it three selected screenshots of the Python shell window, beginning, middle, and successful end.

File naming: For ease of identification of your submitted work, make sure that you adhere to the following **file naming convention:** for each file XYZ.py, or screenshot that you submit, name the file

Lastname_Firstname_####_XYZ.py(.<any other>)

where #### are the last four digits of your student id. **Your work may not be graded if you do not adhere to this naming convention.**