homework-task1

September 27, 2023

1 Task 1

[1]:

```
from simpleai.search import CspProblem, backtrack

# This definition checks if the last word is the same length or one letter␣
 ↪longer than the other two words.
def check_last(first,second,last):
    if len(first) > len(second):
        if len(last)<len(first) or len(last)>len(first)+1:
            last_word = input('The last word that you chose is invalid, please␣
 ↪choose another one. ')
            check_last(word1,word2,last_word)
    if len(first) < len(second):
        if len(last)<len(second) or len(last)>len(second)+1:
            last_word = input('The last word that you chose is invalid, please␣
 ↪choose another one. ')
            check_last(word1,word2,last_word)

# Here we check the length of the first word. In this case it is set to 5 so␣
 ↪that it does not take too long to solve the puzzle.
def check_length1(word):
    if len(word)>5:
        word1 = input('The word that you chose is too long, please choose␣
 ↪another one. ')
        check_length1(word1)

# Here we check the length of the second word. In this case it is set to 5 so␣
 ↪that it does not take too long to solve the puzzle.
def check_length2(word):
    if len(word)>5:
        word2 = input('The word that you chose is too long, please choose␣
 ↪another one. ')
        check_length2(word2)

# Here we check if there are more than 10 different letters because you only␣
 ↪have 10 numbers.
def check_letters(letterslist):
    if len(letterslist)>10:
        input_words()
        letters=[]

# this function is used to change the words if there are too manny different␣
 ↪letters.
def input_words():
    word1 = input('There were to manny different letters. What is your first␣
 ↪word? ')
    word2 = input('What is your second word? ')
    last_word = input('What is your last word? ')

# Here we input all the words and they are checked if they are correct.
word1=''
word2=''
last_word=''

word1 = input('What is your first word? ')
check_length1(word1)
word2 = input('What is your second word? ')
check_length2(word2)
last_word = input('What is your last word? ')

check_last(word1,word2,last_word)

letters=[]
domains = {}

# This definition fills up the letters list with all different letters.
def add_to_list(word):
    for letter in word:
        if letter not in letters:
            letters.append(letter)

add_to_list(word1)
add_to_list(word2)
add_to_list(last_word)

# Here we turn the letters list into a tupple and give a range of numbers to␣
 ↪all of the letters.
variables = tuple(letters)
for letter in variables:
    # The if function is used to check if the letter is a first letter of a␣
 ↪word.
    # If it is then the range has to be from 1 to 9, otherwise it is from 0 to␣
 ↪9. (The last number of the range function is not included.)
    if letter == word1[0] or letter == word2[0] or letter == last_word[0]:
```

```python
        domains[letter] = list(range(1, 10))
    else:
        domains[letter] = list(range(0, 10))

# This checks if the tupple of letters are all different letters.
def constraint_unique(variables, values):
    return len(values) == len(set(values))

# Here we build the words back up but with the values that all of the letters represent and check if word1 + word2 = last_word.
def constraint_add(variables, values):
    # I make a string where the value of the word will be stored as a string.
    wordString1 = ''
    # Then I loop over all of the letters in that word.
    for letter in word1:
        # For every letter I search for the index of that letter in the letters tuple (with 'variables.index(letter)').
        # Then I get the value of that letter and add it to the wordString1 and if all the letters are done then you have the value of the word in this string.
        wordString1+=str(values[variables.index(letter)])

    wordString2 = ''
    for letter in word2:
        wordString2+=str(values[variables.index(letter)])

    wordStringlast = ''
    for letter in last_word:
        wordStringlast+=str(values[variables.index(letter)])

    # Here i turn the wordstrings into an integer, add them and the check if it is the same as the value of the last word.
    return (int(wordString1) + int(wordString2)) == int(wordStringlast)

# I set the constraints for the variables (letters tuple).
constraints = [
    (variables, constraint_unique),
    (variables, constraint_add),
]

# I use the simpleai cspProblem and give it the letters the ranges and the constraints.
problem = CspProblem(variables, domains, constraints)
# Then you just backtrack the problem so you get to a solution and then print this solution.
output = backtrack(problem)
```

```
print(word1,'+',word2,'=',last_word)
print('\nSolutions:', output)
```

```
to + go = out

Solutions: {'t': 2, 'o': 1, 'g': 8, 'u': 0}
```

2 Streamlit

Streamlit website: https://kieran-cornelissen-homework-task1.streamlit.app/

[ ]:

```
import streamlit as st
from simpleai.search import CspProblem, backtrack

# This definition checks if the last word is the same length or one letter␣
 ↪longer than the other two words.
def check_last(first,second,last):
    if len(first) > len(second):
        if len(last)<len(first) or len(last)>len(first)+1:
            st.text('The last word that you chose is invalid, please choose␣
 ↪another one. ')
            #check_last(word1,word2,last_word)

    if len(first) < len(second):
        if len(last)<len(second) or len(last)>len(second)+1:
            st.text('The last word that you chose is invalid, please choose␣
 ↪another one. ')
            #check_last(word1,word2,last_word)


# Here we check the length of the first word. In this case it is set to 5 so␣
 ↪that it does not take too long to solve the puzzle.
def check_length1(word):
    if len(word)>5:
        st.text('The word that you chose is too long, please choose another one.
 ↪ ')
        #check_length1(word1)
    return True

# Here we check the length of the second word. In this case it is set to 5 so␣
 ↪that it does not take too long to solve the puzzle.
def check_length2(word):
    if len(word)>5:
        st.text('The word that you chose is too long, please choose another one.
 ↪ ')
        #check_length2(word2)
    return True

# Here we check if there are more than 10 different letters because you only have 10 numbers.
def check_letters(letterslist):
    if len(letterslist)>10:
        letters=[]
        input_words()

# this function is used to change the words if there are too manny different letters.
def input_words():
    st.text('There were to manny different letters. What is your first word? ')

# Here we input all the words and they are checked if they are correct.
word1=''
word2=''
last_word=''

if word1 == '':
    word1 = st.text_input('What is your first word? ')
    check_length1(word1)
if word2 == '' and word1 != '':
    word2 = st.text_input('What is your second word? ')
    check_length2(word2)
if last_word == '' and word2 != '' and word1 != '':
    last_word = st.text_input('What is your last word? ')
if last_word != '':
    check_last(word1,word2,last_word)



if last_word != '' and word2 != '' and word1 != '' and len(word1)<6 and len(word2)<6:
    letters=[]
    domains = {}

# This definition fills up the letters list with all different letters.
    def add_to_list(word):
        for letter in word:
            if letter not in letters:
                letters.append(letter)

    add_to_list(word1)
    add_to_list(word2)
    add_to_list(last_word)
```

```
# Here we turn the letters list into a tupple and give a range of numbers to all of the letters.
    variables = tuple(letters)
    for letter in variables:
    # The if function is used to check if the letter is a first letter of a word.
    # If it is then the range has to be from 1 to 9, otherwise it is from 0 to 9. (The last number of the range function is not included.)
        if letter == word1[0] or letter == word2[0] or letter == last_word[0]:
            domains[letter] = list(range(1, 10))
        else:
            domains[letter] = list(range(0, 10))

# This checks if the tupple of letters are all different letters.
    def constraint_unique(variables, values):
        return len(values) == len(set(values))

# Here we build the words back up but with the values that all of the letters represent and check if word1 + word2 = last_word.
    def constraint_add(variables, values):
    # I make a string where the value of the word will be stored as a string.
        wordString1 = ''
    # Then I loop over all of the letters in that word.
        for letter in word1:
        # For every letter I search for the index of that letter in the letters tuple (with 'variables.index(letter)').
        # Then I get the value of that letter and add it to the wordString1 and if all the letters are done then you have the value of the word in this string.
            wordString1+=str(values[variables.index(letter)])

        wordString2 = ''
        for letter in word2:
            wordString2+=str(values[variables.index(letter)])

        wordStringlast = ''
        for letter in last_word:
            wordStringlast+=str(values[variables.index(letter)])

    # Here i turn the wordstrings into an integer, add them and the check if it is the same as the value of the last word.
        return (int(wordString1) + int(wordString2)) == int(wordStringlast)

# I set the constraints for the variables (letters tuple).
    constraints = [
        (variables, constraint_unique),
        (variables, constraint_add),
    ]

# I use the simpleai cspProblem and give it the letters the ranges and the constraints.
    problem = CspProblem(variables, domains, constraints)
# Then you just backtrack the problem so you get to a solution and then print this solution.
    output = backtrack(problem)

    st.text(str(word1+' + '+word2+' = '+last_word))
    st.text('\nSolutions:')
    st.text(output)
```