Cryptarithmetic Puzzle

Een cryptarithmetische puzzel is een soort wiskundige puzzel of woordpuzzel waarbij het doel is een rekenkundige vergelijking te ontcijferen waarbij de cijfers worden weergegeven door letters of symbolen. De uitdaging is om de juiste cijfer-naar-letter-toewijzing te vinden die voldoet aan de beperkingen van de vergelijking. Deze puzzels worden vaak gebruikt voor recreatieve en educatieve doeleinden en kunnen behoorlijk uitdagend zijn.

Ik importeer eerst de libraries die nodig zijn om de taak te maken.

In [ ]:
```
import streamlit as st
from simpleai.search import CspProblem, backtrack
```

Eerst heb ik wat opzoekwerk gedaan over hoe ik een input kan vragen in streamlit. Daar heb ik dan snel een oplossing voor gevonden in de streamlit documentatie.

De streamlit app vraagt aan de gebruiker om 3 woorden in te geven.

De woorden worden dan geconcateneerd en in een set geplaatst zodat ik een lijst krijg met alle letters die zijn ingegeven. Deze lijst gaat geen dubbele letters bevatten omdat het in een set werd gestoken.

In [ ]:
```
number1 = st.text_input("Enter the first word:") #TO
number2 = st.text_input("Enter the second word:") #GO
result = st.text_input("Enter the result:") #OUT
variables = set(number1+number2+result) #TOGU
```

Over het algemeen leek de code die ik moest gebruiken sterk op de code die ik in de oefeningen van hoofdstuk 3 had gemaakt. Het grootste verschil was om de mogelijkheid te bieden voor de letters zelf te kunnen kiezen en deze niet te hardcoden in dit programma. Door dit te doen kwamen er andere problemen tevoorschijn.

Het grootste probleem was dat ik steeds een index out of range error kreeg bij onderstaande code omdat de gebruiker nog geen input had gegeven bij het opstarten van het programma.

In dit stuk code voorkom ik dat de eerste letter van de ingegeven woorden een 0 is.

Om foutmeldingen te voorkomen waarin gezegd wordt dat de index van een van deze woorden "out of range" is doordat er nog geen input werd ingevuld door de gebuiker zet ik dit stuk code in een if statement. In deze if statement kijk ik of dat alle 3 de inputs werden ingevuld.

In [ ]:
```
if len(number1) > 0 and len(number2) > 0 and len(result) > 0:
        domains = {
                number1[0]: list(range(1, 10)),
                number2[0]: list(range(1, 10)),
                result[0]: list(range(1, 10)),
        }
```

De andere letters zitten nog niet in de domain dictionary om aan te geven welke mogelijke waarden een letter kan hebben.

Om dit te doen haal ik elke letter uit de variables set en check ik of deze nog niet in de domains dictionary zit. Hierna geef ik mee dat deze letters een waarde van 0 t.e.m. 9 kunnen aannemen.

In [ ]:
```
if len(number1) > 0 and len(number2) > 0 and len(result) > 0:
    for letter in variables:
        if letter not in domains:
            domains[letter] = list(range(0, 10))
```

Hier schrijf ik 2 functies die gebruikt worden in een Constraint Satisfaction Problem zodat de cijfers die toegekend worden aan de letters voldoen aan de cryptografische puzzel.

Deze functie zorgt ervoor dat 1 cijfer zoals '3' niet aan meerdere letters kan toegewezen worden

In [ ]:
```
def constraint_unique(variables, values):
    return len(values) == len(set(values)) #verwijder dubbele waarden en tel de lengte
```

Als het script een oplossing vindt waarbij alle letters een cijfer vertegenwoordigen en waarbij de return statement True geeft dan is er een oplossing gevonden.

In [ ]:
```
def constraint_add(variables, values):
    factor1 = ""
    factor2 = ""
```

```
    sum = ""
    for char in number1: #ga elk character van de string af
        #het huidige character wordt hier gebruikt om de index te vinden
        #van dat character in de variables set. bv: variables.index('O') == 1

        #de index wordt dan gebruikt om de waarde
        #dat aan die letter werd gegeven te krijgen. bv: values[1] == 1

        #uiteindelijk wordt deze waarde omgezet van een integer naar een string
        #zodat deze toegevoegd kan aan de string factor1 zonder dat de integers worden opgeteld
        #bv: '3' + '1' == '31' en niet 3 + 1 == 4
        factor1 += str(values[variables.index(char)])
    for char in number2:
        factor2 += str(values[variables.index(char)])
    for char in result:
        sum += str(values[variables.index(char)])
        #de strings moeten naar een integer worden omgezet
        #om te kijken of de geassigneerde waardes ervoor zorgen dat factor1 + factor2 == sum
    return (int(factor1) + int(factor2)) == int(sum)
```

Hier geef ik een lijst van constraints mee die in het Constraint Satisfaction Problem worden meegegeven.

In [ ]:
```
constraints = [
    (variables, constraint_unique), #variables = TOGU
    (variables, constraint_add), #variables = TOGU
]
```

Hier wordt weer nagekeken of dat alle inputs zijn ingevuld om een index out of range error te vermijden.

Er wordt een CspProblem object gemaakt met de constructor 'CspProblem' die 3 waarden gebruikt namelijk:

1. De variables - deze bevatten alle unieke letters die werden ingegeven door de gebruiker
2. De domains - deze bevat de mogelijke waarden elke letter kan hebben
3. De constraints - deze omschrijft aan welke voorwaarde de assignatie van cijfers aan letters moeten voldoen

Als er nog geen inputs zijn ingegeven dan wordt er 'No solution' afgeprint.

In [ ]:
```
if len(number1) > 0 and len(number2) > 0 and len(result) > 0:
    problem = CspProblem(variables, domains, constraints)
    output = backtrack(problem)
    print('\nSolutions:', output)
else:
    output = None
    print('No solution')
```

```
No solution
```

Om de juiste output te tonen in streamlit heb ik ook veel problemen mee ondervonden. Eerst werd alles onder elkaar getoond omdat ik telkens een `st.write` deed voor elke letter en cijfer. Ik heb lang geprobeerd om het mooi in een grid 3X3 te krijgen en deze naast elkaar te tonen maar uiteindelijk is het enkel gelukt om de 'grid' onder elkaar te tonen

In deze Code cell staat alles om het resultaat te tonen in streamlit.

Eerst is er een if statement die checkt of er een uitkomst is gevonden zodat er geen 'NoneType' errors voorkomen.

In [ ]:
```
letters = []
numbers = []
letters1 = ""
letters2 = ""
letters3 = ""
if output is not None:
    #hier worden de letters en cijfers die aan elkaar verbonden zijn
    #uit de dictionary gelezen en dan toegevoegd aan de letters en numbers list.
    for letter, number in output.items():
        letters.append(letter)
        #ik maak een string van de nummers
        #zodat de spacing in de output overeenkomt met de letters
        numbers.append(str(number))
    #de lijsten worden onder elkaar getoond
    st.text(letters)
    st.text(numbers)
    #---------------------------------
    #hier worden de ingegeven woorden simpleweg getoond om aan te tonen wat de oorspronkelijke som is
    #en zodat je kan makkelijk zien hoe de cijfers overeenkomen met de letters
    st.write(number1)
    st.write("&plus;", number2)
```

```
    st.write(result)
    #---------------------------------
    #hier ga ik door elk woord heen door elke letter af te gaan
    for letter in number1:
        #met deze letter ga ik de value opvragen die eraan is gekoppeld in de output dictionary.
        #bv: output = {'T' : 2} dan wordt output['T'] == 2

        #dit cijfer moet ik nog omzetten naar een string zodat ik deze kan toevoegen
        #aan de string letters1
        letters1 += str(output[letter])
    for letter in number2:
        letters2 += str(output[letter])
    for letter in result:
        letters3 += str(output[letter])
        #hier worden de string getoond op streamlit
    st.write(letters1)
    st.write("&plus;", letters2)
    st.write(letters3)
```

Generative AI Tools

Ik heb vooral BingAI gebruikt i.p.v. ChatGPT omdat ik vooral vragen moest stellen over hoe ik de juiste output kan tonen in een streamlit app en omdat ChatGPT gelimiteerd was tot September 2021 werden er soms niet de nieuwste functies van streamlit getoond.

Prompts used

In python can I loop over a list to create a key value pair in a dictionary - BingAI - Bij deze prompt wordt er mooi uitgelegd hoe dit haalbaar is met een voorbeeld.

In python using the simpleai library explain how the constraints work - BingAI - Hier kan je zien dat de tekst bijna rechtstreeks van de simpleai docs komt en wordt er terug een voorbeeld gegeven.

in streamlit when assigning a user input to a variable can i assign a default value if the user doesn't enter a value - BingAI - Dit had ik gevraagd om de index out of range error proberen op te lossen. BingAI zegt ook dat dit mogelijk is maar wanneer ik dit toepaste kon ik maar 1 keer het hele programma laten runnen, daarna als ik een woord veranderede in de input velden stopte het programma.

how do i make a grid in streamlit that grows dynamically with the length of a number where every individual number has its own column - BingAI - Hier stelde het voor om st.beta_columns te gebruiken terwijl dat dit niet meer bestaat, dit is immers st.columns geworden dus de bron die het geraadpleegd heeft is niet de meest recente.

explain this code: problem = CspProblem(variables, domains, constraints) output = backtrack(problem) - ChatGPT

explain what a cryptarithmetic puzzle is - ChatGPT - Hier krijg je niet de bronnen te zien omdat het ChatGPT is dus als je niet goed zou weten wat een cryptarithmetic puzzle is dan kan het zijn dat je een foute uitleg aanleerd.