https://colab.research.google.com/drive/1BjAheJMNWXI68AgBvFWbceGznQTflSwG?usp=sharing

Team Challenge

Presented by:
Jann Moises Nyll De los Reyes
Dylan James N. Dejoras

The Sheep, Wolf and Cabbage Problem

> A farmer with a wolf, a goat, and a cabbage must cross a river by boat.The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten?

Solution:

SWC -- > S -- > River WC <--- S River W --> C --> SC River w <---S--C River S---> w---> C river ----> S--- WC River ----> CWS

```
class Boat:

    def __init__(self, name):
        self.name = name

    def ride_boat(self):
        pass

class Farmer(Boat):

    def ride_boat(self):
        print("The " + self.name + " rode the boat.")

    def final_cross(self):
        print("All characters have safely crossed the river.")

class Wolf(Boat):

    def ride_boat(self):
        print("The " + self.name + " rode the boat while looking menacingly at the sheep.")

class Cabbage(Boat):

    def ride_boat(self):
        print("The " + self.name + " rode the boat and it is safe from the sheep.")

class Sheep(Boat):

    def ride_boat(self):
        print("The " + self.name + " rode the boat and it is hungry for the cabbage.")


sheep = Sheep("Sheep")
wolf = Wolf("Wolf")
cabbage = Cabbage("Cabbage")
farmer = Farmer("Farmer")

boat = []
island_one = ['W', 'S', 'C', 'F']
island_two = []

sheep = Sheep("Sheep")
wolf = Wolf("Wolf")
cabbage = Cabbage("Cabbage")
farmer = Farmer("Farmer")

boat = []
island_one = ['W', 'S', 'C', 'F']
island_two = []
```

```
while len(island_two) < 4:
    input_char = input("Enter who will you like to carry across:\nW - Wolf\nS - Sheep\nC - Cabbage\nF - Farmer\n").upper()

    if input_char not in island_one + island_two:
        print("Invalid choice")
        continue

    if input_char == 'F' and island_two == 3: # delete
        print("The Farmer must be the last one to cross.")
        continue

    if input_char == 'W' and 'S' in boat:
        print("The Wolf cannot be left alone with the Sheep.")
        continue

    if input_char == 'C' and 'S' in boat:
        print("The Cabbage cannot be left alone with the Sheep.")
        continue

    if input_char == 'S' and 'F' not in boat:
        print("The Sheep must be accompanied by the Farmer.")
        continue

    if input_char in island_one:
        boat.append(input_char)
        choice = input("Do you want it to cross the island?(Y/N) ").upper()

        if choice == 'Y':
            island_one.remove(input_char)
            boat.remove(input_char)
            island_two.append(input_char)

            if input_char == 'W':
                wolf.ride_boat()
            elif input_char == 'S':
                sheep.ride_boat()
            elif input_char == 'C':
                cabbage.ride_boat()
            elif input_char == 'F':
                farmer.ride_boat()
        elif choice == 'N':
            continue

    elif input_char in island_two:
        boat.append(input_char)

        if input_char == 'W':
            wolf.ride_boat()
        elif input_char == 'S':
            sheep.ride_boat()
        elif input_char == 'C':
            cabbage.ride_boat()
        elif input_char == 'F':
            farmer.ride_boat()

        choice = input("Do you want it to cross the island?(Y/N) ").upper()

        if choice == 'Y':
            island_two.remove(input_char)
            boat.remove(input_char)
            island_one.append(input_char)
        elif choice == 'N':
            continue

    result_string = ' '.join(boat)
    result_string2 = ', '.join(island_two)
    result_string3 = ', '.join(island_one)
    print("Boat: " + result_string)
    print("Island 2: " + result_string2)
    print("Island 1: " + result_string3)

    if (island_two == ['S', 'W'] and island_one == ['C', 'F']):
        pass

    elif (island_two == ['C', 'S', 'F'] or island_one == ['C', 'S', 'F'] or island_two == ['S', 'C', 'F'] or island_one == ['S', 'C', 'F']  )
        print("The cabbage got eaten by the sheep...")
        break
```

```
    elif (island_two == ['W', 'S', 'F'] or island_one == ['W', 'S', 'F'] or island_two == ['S', 'W', 'F'] or island_one == ['S', 'W', 'F']):
        print("The sheep got eaten by the wolf...")
        break
    elif (island_two == ['W', 'C', 'S', 'F']):
        farmer.final_cross()
        break

 Invalid choice
 Invalid choice
```

Double-click (or enter) to edit

Alternative Solution

In the given puzzle, we have the following conditions:

- We want the Farmer (Jack) to cross the river with his wolf,sheep and cabbage 🧑 🐑 🐺 🥬
- The plank can only carry him and one of the items at a time. 🧑 ▯ 🛶
- If he leaves the sheep with the wolf, the wolf will eat the sheep. 🐺 🍽 🐑
- if he leaves the sheep with the cabbage, the sheep will eat the cabbage. 🐑 🍽 🥬
- Jack has to be on each trip since he is the only one that can carry the items delivered on the each end of the river. 🧑 ⛵

In order to cross the river here is the following solution:

1. Position 1:🧑 🐺 🐑 🥬 ➞ Position 2:
2. Position 1:🐺 🥬 ➞ Position 2: 🧑 🐑
3. Positon 1: 🧑 🐺 🥬 ⬅ Position 2:🐑
4. - A.) Position 1: 🥬 ➞ Position 2: 🧑 🐺 🐑
   - B.) Position 1: 🐺 ➞ Position 2: 🧑 🥬 🐑
5. - A.) Position 1: 🧑 🐑 🥬 ⬅ Position 2: 🐺
   - B.) Position 1: 🧑 🐑 🐺 ⬅ Position 2: 🥬
6. Position 1: 🐑 ➞ Position 2: 🧑 🥬 🐺
7. Position 1: 🧑🐑 ⬅ Position 2: 🐺 🥬
8. Position 1: ➞ Position 2: 🧑 🐺 🐑 🥬

Write all the possible valid statement into vertices.

- a --> (FWSC , Ø)
- b --> (FWS, C)
- c --> (FSC, W)
- d --> (FCW, S)
- e --> (FS ,CW)
- f --> (WC,FS)
- g --> (S, WFC)
- h --> (W, SFC)
- i --> (C, SFW)
- j --> (Ø, FWSC)

POSSIBLE SEQUENCES :

- A - F - D - I - C - G - E - J
- A - F - D - H - B - G - E - J

```
import networkx as nx
#import matplotlib.pyplot as plt

# Create a new graph
G = nx.Graph()

# Add nodes to the graph
G.add_nodes_from(['A', 'B', 'C', 'D', 'E','F','G','H','I','J'])

# Add edges to the graph
G.add_edges_from([('A', 'F'), ('F', 'D'), ('D', 'I'), ('D', 'H'), ('I', 'C'), ('H', 'B'), ('B', 'G'),('C','G'),('G','E'),('E','J')])

paths = (nx.all_simple_paths(G, 'A','J'))

# Print the two paths
print('The two paths from node A to node J are:')
for path in paths:
    print(' -> '.join(path))

# Draw the graph
pos = nx.spring_layout(G)
nx.draw_networkx(G, pos=pos, with_labels=True, node_color='lightblue', node_size=1000, font_size=20, font_weight='bold', font_color='black',
#plt.show()
```

```
The two paths from node A to node J are:
A -> F -> D -> I -> C -> G -> E -> J
A -> F -> D -> H -> B -> G -> E -> J
```