Sphinx's math WRITEUP

The challenge provided the following link:

- http://codingbox4sm.reply.it:1338/sphinxsEquaji/

By navigating to the link, a system of linear equations was proposed. The challenge required to solve this system of linear equation in less than 3 seconds, for a total of 512 steps.

So, we coded a Python3 script that iteratively parses and solves the system, pushing each time the answer to the server, which you can find below:

```
import requests
import re
import numpy as np

# URL for the challenge
url = "http://codingbox4sm.reply.it:1338/sphinxsEquaji/"
url_post = "http://codingbox4sm.reply.it:1338/sphinxsEquaji/answer"

enigma_session = requests.Session()
curr_enigma = enigma_session.get(url).text

while(1):
    # Get the current enigma and step
    enigma = re.search(r"<div class=\"enigma\">(.+?)<\/div>", curr_enigma, flags=
    step = re.search(r"<h2>(.+?)</h2>", curr_enigma).group(1)
    equations = re.findall(r"<p>(.+?)</p>", enigma)

    print(step)
    print(enigma)

    # Parse the equations
    parsed_equations = []
    b = []
    for eq in equations:
        sx, _ , dx = eq.partition("=")
        try:
            dx = int(float(dx))
```

```
            b.append(dx)
        except:
            pass

        emojis = re.findall(r'[^\d+-\.\(\)]', sx)
        coeff = []
        for x in re.split(r'[^\d+-\.\(\)]', sx):
            if x:
                coeff.append(eval(x))

        eq = {}
        for e, c in zip(emojis, coeff):
            eq[e] = c

        parsed_equations.append(eq)

    # Get all the labels contained in the equations
    labels = set([])
    for peq in parsed_equations:
        labels |= set([*peq])

    # Build the matrix a
    aa = []
    for peq in parsed_equations:
        pk = list(peq.keys())
        tmp = []
        for l in labels:
            if l in pk:
                tmp.append(peq[l])
            else:
                tmp.append(0)

        aa.append(tmp)

    # Keep only useful equations of the system
    squared_matrix = []
    for index_row in reversed(range(len(aa[:-1]))):
        tmp_matrix=squared_matrix[:][:]
        tmp_matrix.append(aa[index_row])
        if np.linalg.matrix_rank(tmp_matrix)==len(squared_matrix)+1:
            squared_matrix=tmp_matrix
        else:
            b.pop(index_row)
```

```
    # Solve the linear system of equations
    a = np.array(squared_matrix)
    b = np.array(b[::-1])
    x = np.linalg.solve(a, b)

    # Solve the last equation
    ris=np.dot(np.array(aa[-1]),x)

    # Push the answer to the server
    data_ = {"answer": str(int(round(ris)))}
    curr_enigma = enigma_session.post(url_post, data=data_).text

    # Print the flag once it is printed in the webpage
    if "FLG" in curr_enigma:
        print(curr_enigma)
        quit()
```

At the end of the step 512, we get the flag: `{FLG:F0r63t_7h3_4r4b1c-num3r4l5_hi3r06lyph5_w1ll_n3v3r-d13!}`