Cat Code

TechmandanCZ

Úloha obsahuje 2 soubory, meowmeow.py (soubor 1) a meow.py (soubor 2)

```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-
"""
meow meow meow
"""


from meow import meow, meowmeow


def meoow():
    """
    meoow
    """
    meoword = ''
    while meoword != 'kittens':
        meoword = input('Who rules the world? ')
        if meoword in ['humans', 'dogs']:
            print('MEOW MEOW!')
    print(meowmeow(meow(sum([ord(meow) for meow in meoword]))))


if __name__ == '__main__':
    meoow()

# MEOF
```

```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-
"""
meow
"""

UNITE = 1
UNITED = 2
meeow = [
  [80, 81],
  [-4, 13],
  [55, 56],
  [133, 134],
  [-8, -7, -5],
  [4, 5],
  [5, 6],
  [6, 7],
  [7, 8],
  [15, -1],
  [11, 12],
  [13, 14],
  [17, 18],
  [18, 19],
  [15, 21],
  [22, 23],
  [26, 27],
  [44, 45],
  [48, 49],
  [31, -29],
  [50, 51],
  [60, 61],
  [72, 73],
  [73, 74],
  [19, 2, 20]]


def meow(kittens_of_the_world):
    """
    meowwwwww meow
    """
    print('meowwww ', end='')
    if kittens_of_the_world < UNITED:
        return kittens_of_the_world
    return meow(kittens_of_the_world - UNITE) + meow(kittens_of_the_world - UNITED)


def meowmeow(meow):
    """
    meow meow meow
    """
    meeoww = ''
    for meoww in meeow:
        print('meowwww ', end='')
        meeoww = f"{meeoww}{chr(int(''.join(str(meow)[m] for m in meoww)))}"
    return meeoww

# MEOF
```

První věcí co jsem udělal je nastavení “meoword” na “kittens”, abych nemusel pokaždé psát daný text.
Následně jsem šel na analýzu meow.py

Soubor nevypadá že by obsahoval přímo šifrování, ale naopak spíše dešifrování, jen se spouští až moc pomalu.

První funkce počítá *něco* pomocí rekurze, která je v pythonu (i mimo něj, ale v pythonu je to znát o to více) pomalá. Začal jsem přemýšlet jak funkci přepsat do klasické for loopy, i jsem udělal pár pokusů, následně mě napadlo prostě přidat cache jelikož funkce díky rekurzi neustále počítala ty stejná čísla. Přidal jsem proto jednoduchou cache:

```
cache = [None] * 771

def meow(num):
    if cache[num] is not None:
        print("cached", num)
        return cache[num]
    if num < 2:
        return num
    i = meow(num - 1) + meow(num - 2)
    print("i", num, i)
    cache[num] = i
    # print(cache)
    return i
```

A kód mi vyzradil svá tajemství, vlajka byla na světě.

Později jsem při řešení jiného problému narazil na dotaz na stack overflow s velice podobnou podobou - zjistil jsem tak dvě informace, jednak že místo vlastní implementace cache stačilo přidat dekorátor @lru_cache(maxsize=None) a jednak že daný kód počítá fibonacciho posloupnost. Což mi při zpětném pohledu asi mohlo dojít, ale spíš jsem zvyklý koukat na XOR funkce a podobné.