VIGENERE

```
cypher = 'nxxmgd{Uynbe_Fhr_Jyuqk_Tbs}'
key = "vigenere"
dec = ""

j = 0  # indice per la chiave

for c in cypher:
    if c.isalpha():
        k = ord(key[j % len(key)].lower()) - ord('a')
        if c.islower():
            dec += chr((ord(c) - ord('a') - k) % 26 + ord('a'))
        else:
            dec += chr((ord(c) - ord('A') - k) % 26 + ord('A'))
        j += 1
    else:
        dec += c
        j+=1    #da togliere all'occorrenza(se non funziona na tega)


print(dec)
```

https://www.dcode.fr/vigenere-cipher

XOR

```
X1 =
bytes.fromhex("b840946f97ffe078ce6581d145ff3bd86cdfd0add863fc718300")


def xor(a, b):
    return bytes(i ^ j for i, j in zip(a, b))
```

MD5:
https://crackstation.net/

SUBSTITUTION CYPHER:
https://www.guballa.de/substitution-solver

```
frequenze= {}
text="Teeez"
for c in text:
    if c not in frequenze:
        frequenze[c]=1
    else:
        frequenze[c]+=1

sorted_x = sorted(frequenze.items(), key=lambda kv: kv[1], reverse =
True)

print(sorted_x)
```

```
import string
import base64
with open("ciphertext.txt","r") as f:
    testo=f.read()

testo=testo.replace("Z","0").replace("O","1")

t64=""
for i in testo.split(" "):
    t64+=chr(int(i,2))

flag=base64.b64decode(t64).decode("utf-8")
print(type(flag))
dizionario_testo=
{'B':'s','N':'p','O':'r','R':'i','T':'t','U':'z','A':'o','W':'m','Q':'h
','K':'a','E':'w','Y':'e','G':'n','S':'k','P':'v','Z':'y','X':'f','V':'
d','F':'u','C':'c','J':'l','I':'g','H':'b','D':'j','L':'x'}

for k,v in dizionario_testo.items():
    flag=flag.replace(k,v)

print(flag)
```

CAESAR CYPHER (cambiare all'occorrenza -50, 50):

```
import base64

stringa = base64.b64decode("fYZ7ipGIjFtsXpNLbHdPbXdaam1PS1c5lQ==")
```

```
for i in range(-50, 50):
    for c in stringa:
        print(chr(c+i), end="")
    print("\n")
```

https://cryptii.com/pipes/caesar-cipher

URL DECODER:
https://www.urldecoder.org/

BASE64:
https://www.base64decode.org/

MORSECODE:
https://morsecode.world/international/translator.html

WEB:

Puoi runnare le funzioni nella console
Guarda le richieste nel tab Network
Guardare i cookies e le session tab Application
Guardare i commenti (al 90% trovi il backend)
Occhio ai casting
Occhio ai campi di input per injection %26 = & negli url

comandi:
ls -l: directories
cat: mostra il file
curl [url]: per fare richieste http

PER SQL INJECTION:
anything 'or'1'='1 OPPURE ***' or 1=1#***

REQUEST TO THE SITE WITH THE USER-AGENT CHANGED

```
import requests
import string


for i in string.ascii_letters:
    url = f"http://127.0.0.1:9000/?pass={i}"
    r = requests.get(url, headers={'User-Agent': str(i)})
    print(r.text)
```

TO MAKE REQUEST WITH AN IP:PORT AND COOKIES:

```
import hashlib
import codecs
import numpy as np
import requests

#IP
ip = "127.0.0.1"
port= "8084"

#we first check that our MD5 works by comparing Md5(100) with
#the one in the webpage
control = "f899139df5e1059396431415e770c6dd"
tester = 100
tester_b = str.encode(str(tester))
```

```
tester_md5 = hashlib.md5(tester_b).hexdigest()
print(f"tester={tester_md5 == control}")

#try a request to the web application
cookies = {"UID":tester_md5}
r = requests.post(f"http://{ip}:{port}", cookies = cookies)
def_flag = r.cookies["FLAG"] #default flag

print(f"default flag\t{def_flag}")

#brute force cycle
for i in range(100):
    """ compute the MD5 """
    #convert the number to a byte
    byte_i = str.encode(str(i))

    #obtain the MD5
    md5_i = hashlib.md5(byte_i).hexdigest()
    print(md5_i)
    #     """ Send request to the web application """
    #set the cookie
    cookies = {'UID': str(md5_i)}
    r = requests.post(f"http://{ip}:{port}", cookies = cookies)

    #get the i-th flag
    flag_i = r.cookies["FLAG"]

    #if the content of the cookie is different to the 100th one, we
have the flag
    if flag_i != def_flag:
        print(f"FLAG found at iteration={i}\t{flag_i}")
        break
```