7 Dezvoltarea no.3b - Apelarea funcţiei

Modifica rândurile următoare de la final, dupa corpul funcţiei. Testează!

```
mesaj = input("Introdu textul tau:")
rezult  =  crypto_in(mesaj)
print("Text original: ", mesaj
print("Text criptat:  ", rezult)
```

8 Dezvoltarea no.4 - KEY flexibilă

Acum să adăugăm o cheie de criptare flexibilă pe care o va alege utilizatorul. Pentru ca să pastrăm problema cat mai apropiata de CRIPTAREA “CAESAR” vom cere (şi vom verifica) ca aceasta să fie un număr între 1 si 26.
Adaugă acest cod la începutul programului:

```
vKey=input("Cheia de criptare este:")
vKey=int(vkey)
if vKey not in range(1,27):
    print("Cheia  trebuie sa fie un nr intre 1 si 26")
    key = 0
```

Cum se va modifica acum funcţia crypto_in?

```
def crypto_in(mess,key):
  KEY = key
  tx_in= mess
  tx_out=""

  for litera in tx_in:
    litera_noua = litera
    if litera.isalpha():
      num = ord(litera)
      num = num + KEY
      if litera.isupper() and num > ord("Z"):
        num = num - 26
      if litera.islower() and num > ord("z"):
        num = num - 26
      litera_noua = chr(num)

    tx_out=tx_out + litera_noua
  return tx_out
```

Observă ca vom avea doi parametrii acum. Al doilea parametru este key, adică valoarea introdusă de la tastatură de către util utilizator. Chiar la înmceput vom atribui aceasta valoare variabilei KEY pe care am utilizat-o si mai înainte, si mai departe nu se modifică nimic în corpul funcţiei.

Ai grija ca la apelarea funcţiei să foloseşti tot doi parametrii:

```
mesaj = input("Introdu textul tau:")
rezult  =  crypto_in(mesaj, vKey)
print("Text original: ", mesaj)
print("Text criptat:  ", rezult)
```

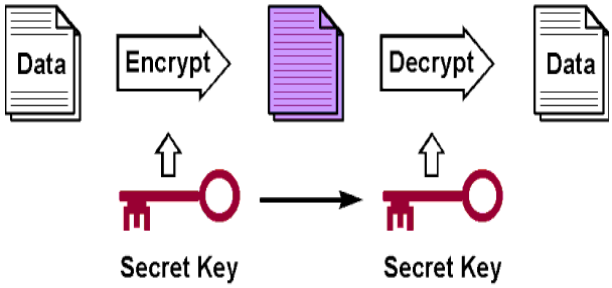
9 Dezvoltarea no.5 – Decriptarea

Vom crea acum si funcţia inversă, pentru decriptarea unui mesaj. Seamană foarte mult cu funcţia originală, doar că operaţiunea de transformare a literei se va face pe dos, adică vom scădea cheia, în loc să o adunăm.

```
def crypto_out(mess,key):
  KEY = key
  tx_in= mess
  tx_out=""

  for litera in tx_in:
    litera_noua = litera
    if litera.isalpha():
      num = ord(litera)
      num = num - KEY
      if litera.isupper() and num < ord("A"):
        num = num + 26
      if litera.islower() and num < ord("a"):
        num = num + 26
      litera_nou = chr(num)

    tx_out=tx_out + litera_noua
  return tx_out
```



Mai trebuie sa avem grija la capetele alfabetului, la fel ca în Dezvoltarea no.2 , dar de data aceasta doar când ajungem la litera A sau a, să incrementăm cu 26 codul ASCI corespunzărtor.

10 Dezvoltarea no.6 – Utilizatorul va alege modul de lucru (Criptare/Decriptare)

Vom cere utilizatorului sa decidă dacă vrea să faca o criptare sau decriptare.

```
mod = input(" Doresti sa faci o Criptare(C) sau o Decriptare (D)?")
mod = mod.upper()

mesaj  = input("Introdu textul tau:")
print(" /nText original:   " , mesaj)

if mod == "C":
  mesaj2 = crypto_in(mesaj,vKey)
  print("Text   criptat:  ", mesaj2)
elif mod == "D":
  mesaj2 = crypto_out(mesaj,vKey)
  print("Text decriptat:  ", mesaj2)
else:
  print(" Va multumesc!  ")
```

11 Finalizarea programului.

Acum trebuie doar să mai punem laolaltă toate secvenţele de cod pe care le-am analizat şi pregătit aici.