# Challenge Dados - Fevereiro

Fala, pessoal! Sejam muito bem-vindos ao 1º Challenge de Python disponibilizado no Let's Code Pass! Você deve estar acompanhando as Short Classes de Python e, agora, é hora de colocar seus novos conhecimentos em prática! Vamos nessa?! 🚀

Antes de tudo, é válido destacar que por se tratar de um desafio, é essencial que você troque uma ideia com outras pessoas, caso sinta dificuldades, ou mesmo possa utilizar do nosso servidor do Discord para enviar suas dúvidas. Ah, claro, você também pode ficar de olho nas dúvidas de outras pessoas para ajudá-las. Tudo isso vai te ajudar a avançar cada vez mais rumo ao seu objetivo!

## Conhecendo o Challenge

### Criptografia via método de substituição (Twist)

Criptografia é uma área da Ciência da Computação que estuda os métodos de comunicação secreta que transformam uma mensagem (**textoplano**) em uma mensagem cifrada (**textocifrado**), de forma que apenas o real receptor da mensagem seja capaz de restaurar o seu conteúdo original.

O ato de transformar uma mensagem em uma mensagem cifrada é chamado de **codificação** e o ato contrário é chamado de **decodificação**. Um método bastante simples de codificação é o M**étodo Twist**, que requer que ambos: remetente e receptor combinem uma **chave secreta k**, que deve ser um inteiro positivo (1, 2, 3, 4, ...).

O **Método Twist** utiliza quatro listas: **textoplano**, **textocifrado**, **codigoplano** e **cifradocodigo**, sendo **textoplano** e **textocifrado** listas de caracteres e **codigoplano** e **cifradocodigo** listas de inteiros. Todas as listas têm tamanho **n**, onde **n** é o tamanho da mensagem a ser codificada (as listas são iniciadas na posição zero de forma que seus elementos são numerados de 0 a n – 1). Para este problema, as mensagens apenas conterão letras minúsculas, pontos e "barra baixa" (underscore, representando espaço em branco).

A mensagem a ser codificada é armazenada na lista **textoplano**. Dada a **chave k**, a codificação é feita da  seguinte forma:

1. Primeiro as letras em **textoplano** são convertidas em códigos inteiros que são armazenados em **codigoplano**. A conversão é feita da seguinte forma:

   '_' = 0, 'a' = 1, 'b' = 2, ..., 'z' = 26 e '.' = 27

2. Depois, cada código em **codigoplano** é convertido em um código codificado em **cifradocodigo** através da seguinte fórmula:

Para todo i de 0 a n – 1,

```
cifradocodigo[i] = (codigoplano[(k * i) mod n] – i) mod 28
```

A decodificação é realizada de maneira análoga (usando-se um inverso multiplicativo modular), devendo-se fazer

```
codigoplano[(k * i) mod n] = (cifradocodigo[i] + i) mod 28
```

Aqui x mod y é o resto da divisão de x por y; por exemplo,

$$3 mod 7 = 3$$

$$22 mod 8 = 6$$

$$-1 mod 28 = 27$$

Você pode usar o operador `%` do Python como sendo $mod$.

3. Por último, os códigos obtidos em **cifradocodigo** são convertidos novamente em letras (pela mesma regra descrita anteriormente e detalhada na dica 1 abaixo) e são armazenadas em **textocifrado**. Esse método funciona apenas quando **n** e **k** são primos entre si; essa propriedade é necessária para garantir que a mensagem decodificada corresponderá à mensagem original (não se preocupem em garantir que eles sejam primos).

A codificação pelo métodos Twist das mensagens `'ola'` e `'wxyz'`, ambas usando a chave 5, são ilustradas na tabela a seguir:

- **exemplo 1 : 'ola'**

| Vetor | [0] | [1] | [2] |
|---|---|---|---|
| textoplano | 'o' | 'l' | 'a' |
| codigoplano | 15 | 12 | 1 |
| | | | |

| | | | |
|---|---|---|---|
| cifradocodigo | 15 | 0 | 10 |
| textocifrado | ‘o’ | ‘_’ | ‘j’ |

- **exemplo 2 : 'wxyz'**

| Vetor | [0] | [1] | [2] | [3] |
|---|---|---|---|---|
| textoplano | ‘w’ | ‘x’ | ‘y’ | ‘z’ |
| codigoplano | 23 | 24 | 25 | 26 |
| cifradocodigo | 23 | 23 | 23 | 23 |
| textocifrado | ‘w’ | ‘w’ | ‘w’ | ‘w’ |

**Dicas sobre manipulação de caracteres**

Em Python pode-se utilizar duas funções pré-definidas para manipular caracteres e seu código numérico correspondente.

Por exemplo, o caractere/dígito ‘0’ tem código 48, o caractere/dígito ‘1’ tem código 49 e assim por diante (esses valores vêm do “velho” sistema de codificação **ASCII – American Standard Code for Interchange**, que usava 8 bits).

Então pode-se usar as funções `ord(char)` e `chr(int)`, a primeira, dado o caractere **C**, devolve seu código e a segunda, dada um inteiro **N** devolve o caractere cujo código é **N**. Experimente o seguinte código em Python:

```
def converteInt2Char(k): # k deve ser inteiro
    return chr(k) # devolve o caractere associado a k
def converteChar2Int(c): # c deve ser caractere
    return ord(c) # devolve o código inteiro associado ao caractere c
print(f"N | converteInt2Char(N)\n48 | {converteInt2Char(48)}\n49 | {converteInt2Char(49)}\nC | converteChar2Int(C)\n'0'| {converteChar2Int('0')}\n'1'| {converteChar2Int('1')}")
```

> 💡 Você pode solucionar o Challenge sem uso das funções `ord()` e `chr()`, se preferir.

# Requisitos Funcionais

Sua tarefa é escrever um programa que codifica e decodifica uma mensagem usando **Método Twist**. Inicialmente o usuário escolhe a opção 0 (codifica) ou 1 (decodifica), sendo:

- Para a opção 0: o usuário deve, em seguida, digitar o valor **k** da chave e depois uma frase (que será codificada);
- Para a opção 1: o usuário deve, em seguida, digitar o valor **k** da chave e depois uma frase (que será decodificada).

Por exemplo, se o usuário digitar:

> `1`

> `5`

> `'o_j'`

Seu programa deve invocar a função de decodificação (com os parâmetros adequados e devolver `'ola'` (correspondendo ao texto decodificado). Então o programa deve imprimir o texto original (decodificando) `'ola'`. Se, por outro lado, o usuário digitar:

> `0`

> `5`

> `'wxyz'`

o programa deve imprimir `'wwww'`.

## Entrada e Saída de Dados

A entrada será via teclado. Primeiramente deve-se digitar a opção (0 ou 1), depois digita-se a chave **k** e por último a mensagem (a ser codificada ou decodificada).

Pode-se supor que o tamanho da mensagem **n** esteja entre 1 e 70 caracteres. A chave **k** será um inteiro positivo menor ou igual a 300 que seja primo com o tamanho **n**, ou seja, mdc(k, n) = 1. Abaixo apresentamos 3 exemplos de codificações/decodificações, para as chaves 5, 11 e 29.

| Chave | Frase original | Frase codificada |
|---|---|---|
| 11 | cachorro. | cbmowxbkg |
| 29 | espero_que_funcione. | edcnkjyzmjrpianvbyzo |

> 💡 Os códigos dos caracteres são sequências, sendo `97` o código do `'a'`, `98` o código do `'b'` e assim por diante. Neste Challenge pode-se supor que serão digitados apenas letras minúsculas (logo entre `'a'` e `'z'`), ponto `'.'` ou "barra baixa" `'_'`.

Abaixo ilustraremos como seriam as mensagens de seu programa para o primeiro exemplo da tabela acima. Atenção, tente copiar precisamente as mensagens como aparecem, pois caracteres errados podem produzir erros.

- ***1. Exemplo de codificação.****

```
Digite 0 para codificar e 1 para decodificar: 0
Digite a chave: 29
Digite a mensagem: espero_que_funcione.
Frase final: edcnkjyzmjrpianvbyzo
```

- ***2. Exemplo de decodificação.****

```
Digite 0 para codificar e 1 para decodificar: 1
Digite a chave: 29
Digite a mensagem: edcnkjyzmjrpianvbyzo
Frase final: espero_que_funcione.
```

# Referência

Baseado em um trabalho pedido em cursos da USP pelo Departamento de Ciência da Computação (DCC).