Cifra de César

Descrição

"Cifra de César (https://pt.wikipedia.org/wiki/Cifra_de_C%C3%A9sar)" é uma espécie de cifra de texto simples que existe desde (pelo menos) a época de Júlio César. Ele pega uma string e a traduz a uma nova string, deslocando o alfabeto que está sendo usado. Isso produz uma mensagem "criptografada" que não pode ser lida facilmente por alguém que não conhece o valor de deslocamento ("shift").

Nesta pergunta, escreveremos um programa para implementar uma cifra desta forma.

Considere colocar o alfabeto ao longo de uma linha e atribuir um número a cada caractere ("a" é 0, "b" é 1, "c" é 2 e assim por diante). Em uma cifra de César:

- o valor numérico de cada caractere é determinado,
- um valor constante (o "deslocamento" ou "shift") é adicionado ao número associado a cada um dos caracteres na sequência original
  - se o valor deslocado for maior de 25, ele deve voltar a 0 (quer dizer, 26 deve ser interpretado como 0, 27 como 1, ...)
  - se o valor deslocado ficar abaixo de 0, ele deve voltar para 25 (por exemplo, -1 deve ser interpretado como 25)
- os números resultantes são convertidos novamente em caracteres.

Defina uma função `caesar_cipher`, que recebe dois argumentos:

- uma string contendo a string a ser criptografada e
- um número inteiro que representa o valor "shift". `caesar_cipher` deve retornar uma string que representa a forma criptografada da string de entrada.

Existem ligeiras variações desta técnica, mas na nossa versão iremos:

- mudar as letras conforme descrito acima (após converter letras maiúsculas na string de entrada para minúsculas),
- adicionar o valor de deslocamento aos caracteres numéricos na string de entrada (voltando a 0 se formos além de 9 - por exemplo, 11 seria interpretado como 1), e
- deixar todos os outros caracteres inalterados na string resultante Abaixo estão vários exemplos:

```
caesar_cipher('abcd', 1) # retorna 'bcde'
caesar_cipher('ABCD', 1) # retorna 'bcde' (converta para minúsculas prime
iro)
caesar_cipher('abcd', -1) # retorna 'zabc'
caesar_cipher('abcd', -3) # retorna 'xyza'

caesar_cipher('this is pretty neat!', 2) # retorna 'vjku ku rtgvva pgcv!'

caesar_cipher("don't you feel like a roman emperor?",13)  # "qba'g lbh sr
ry yvxr n ebzna rzcrebe?"
caesar_cipher("don't you feel like a roman emperor?",2)  # "fqp'v aqw hgg
n nkmg c tqocp gorgtqt?"
caesar_cipher("don't you feel like a roman emperor?",-24)  # "fqp'v aqw h
ggn nkmg c tqocp gorgtqt?"

caesar_cipher("hot dogs are 5.00 here?!?",2)  # 'jqv fqiu ctg 7.22 jgt
g?!?'
caesar_cipher("97 bottles of pop on the wall...",3)  # '20 erwwohv ri srs
rq wkh zdoo...'
caesar_cipher("1 is the loneliest Number...",-7)  # '4 bl max ehgxebxlm g
nfuxk...'
```

Recursos úteis

- O módulo `string` contém uma string que consiste em todas as letras minúsculas, disponíveis como `string.ascii_lowercase` após o módulo `string` ter sido importado. Da mesma forma, ele contém uma string que consiste em todos os dígitos de 0 a 9 em ordem, como `string.digits` .
- Você pode determinar o índice no qual um determinado caractere (`needle`) aparece em uma string (`haystack`) da seguinte maneira: `haystack.find(needle)` . Se `needle` não existir em `haystack` , `find` retornará `-1` .
- Você pode criar uma string consistindo de todos os caracteres em uma string `x` , mas convertida em minúsculas chamando `x.lower()` .

Testando

Faça um plano antes de escrever qualquer código e esboce seu plano no papel. Certifique-se de escrever e testar seu código *manualmente* para obter alguns exemplos simples (mais simples do que os mostrados acima!) antes de usar Python. Desenhe diagramas de ambiente para alguns casos de teste diferentes. Então, somente quando você estiver convencido de que funcionará, execute seu código com Python em sua própria máquina como um teste.

Planejamento

Este é provavelmente o programa mais complicado que escrevemos até agora. É complicado o suficiente para que provavelmente não queiramos resolver tudo de uma vez.

Uma boa estratégia é começar com um resultado vazio, considerar cada caractere na entrada por vez e adicionar os valores deslocados apropriadamente à saída.

Uma coisa que adiciona um nível de complicação é que três tipos diferentes de caracteres (letras, números e pontuação) são tratados de maneira diferente uns dos outros. Assim, pode fazer sentido começar com um exemplo que use apenas letras.

Depois de ter isso em mente, podemos adicionar código para lidar com o caso em que algo não é uma letra. Uma boa maneira de lidar com isso seria verificar se algo é uma letra ou não e decidir o que adicionar à string de saída com base no resultado dessa comparação.

Então você pode fazer uma mudança semelhante para lidar com a distinção restante (entre números e pontuação).

Submissão

Quando estiver pronto, faça upload do seu arquivo Python no **Problema 2.5** no Gradescope. Lembre de