INF100 – Introdução à Programação I

Roteiro da Aula Prática 08 - 9 a 12 de novembro de 2020
Comandos repetitivos e arranjos unidimensionais
Valor: 2 pontos

Instrodução

A leitura de dados com o comando input, por padrão, na linguagem Python, entende que o valor digitado é do tipo string. E uma string é um arranjo de caracteres. A execução dos comandos abaixo imprime cada caractere digitado pelo usuário na tela do computador (uma em cada linha):

```
linha = input('Digite uma frase: ')
for i in range ( 0, len(linha) ) :
    print( linha[i] )
```

A função **len()** retorna a dimensão do arranjo. A codificação ASCII (*American Standard Code for Information Interchange*) é o padrão adotado nos computadores que utilizamos (https://pt.wikipedia.org/wiki/ASCII). O nosso "\n" (contrabarra N), por exemplo é o caractere de código 10, a letra "A" tem o código 65, a letra "a" tem o código 97 e o caractere "0" tem código 48.

A primeira versão da tabela ASCII continha apenas 128 caracteres. Depois, foi padronizada uma tabela ASCII estendida com 256 símbolos, incluindo os caracteres acentuados (ç, ã, á, etc) - https://web.fe.up.pt/~ee96100/projecto/Tabela%20ascii.htm. Hoje existem diversas extensões para incluir símbolos de diferentes línguas tais como chinês, árabe, etc.

Em Python existem funçõe para obter o código numérico de um caractere (**ord()**) e também para obter um caractere (**chr()**), dado seu código numérico um número inteiro. Exemplos de utilização de **ord()** e **chr()**:

```
x = ord('A')              # retorna o valor inteiro 65, x = 65
y = chr(48)               # retorna o valor '0' (string), y = '0' (caractere 0)
z = chr( ord('A') + 1)    # z = chr ( 65+1 ) = chr(66) = 'B'
```

O comando abaixo daria erro!!

```
w = 'A' + 1  # não é possível somar string com número inteiro!!!
```

Já o comando abaixo não dá erro:

```
w = 'A' + 'B'            # w = 'AB' string com 2 caracteres
```

Mas, a operação é chamada de concatenação de strings (que NÃO será usada nesta prática).

Instruções

Nome do arquivo a ser entregue: **p08.py**

Importante: Como qualquer outra prática de INF100 você deve:

1. Criar o cabeçalho obrigatório.
2. Após finalizar o cabeçalho salve o arquivo com o nome correto

3. Leia as instruções até o final e, após finalizar sua leitura, inicie sua programação.

Obs.: Recomenda-se salvar o arquivo com certa frequência para não perder a digitação já feita em caso de uma falha na rede elétrica.

☞ A saída do programa deve obedecer à formatação **exata** mostrada nos exemplos.

☞ Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p08.py**) através do sistema do LBI.

Questões a serem Resolvidas

1) Faça um programa para imprimir os caracteres visíveis da tabela ASCII padrão, com 128 símbolos ou caracteres ou sinais.

   A tabela deverá ser apresentada em 6 colunas, cada uma delas com o código numérico decimal (Dec.) e o caractere (Sinal) correspondente, conforme mostrado na Figura abaixo. Certifique-se de que a janela do Shell tem tamanho (largura) suficiente para imprimir as 6 colunas.

   Para o caractere espaço (que não é visível) deverá ser impresso o nome "espaço".

```
>>> q1()

Tabela ASCII - Caracteres Visíveis

Dec Sinal   Dec Sinal   Dec Sinal   Dec Sinal   Dec Sinal   Dec Sinal
 32 espaço   48   0      64   @      80   P      96   `     112   p
 33   !      49   1      65   A      81   Q      97   a     113   q
 34   "      50   2      66   B      82   R      98   b     114   r
 35   #      51   3      67   C      83   S      99   c     115   s
 36   $      52   4      68   D      84   T     100   d     116   t
 37   %      53   5      69   E      85   U     101   e     117   u
 38   &      54   6      70   F      86   V     102   f     118   v
 39   '      55   7      71   G      87   W     103   g     119   w
 40   (      56   8      72   H      88   X     104   h     120   x
 41   )      57   9      73   I      89   Y     105   i     121   y
 42   *      58   :      74   J      90   Z     106   j     122   z
 43   +      59   ;      75   K      91   [     107   k     123   {
 44   ,      60   <      76   L      92   \     108   l     124   |
 45   -      61   =      77   M      93   ]     109   m     125   }
 46   .      62   >      78   N      94   ^     110   n     126   ~
 47   /      63   ?      79   O      95   _     111   o     127   ¨
 48   0      64   @      80   P      96   `     112   p
>>>
```

2) Uma função importante na segurança da informação é a técnica de criptografia (ou cifragem). Um dos primeiros registros do uso de criptografia foi no Império Romano, pelo imperador César.

A Cifra de César consistia em deslocar os símbolos (letras) com uma determinada distância D, como mostrado a seguir:

Alfabeto regular: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cifra de César (D=3): DEFGHIJKLMNOPQRSTUVWXYZABC

A lógica do processo de cifragem é substituir cada símbolo pelo símbolo que está a uma distância D. A letra A é substituída pela letra D, a letra B pela E e, assim por diante. Para decifrar a mensagem, basta que o destinatário saiba que distância D foi utilizada. Por isso, a distância D é chamada de chave do algoritmo de cifragem.

Este tipo de técnica é chamada de cifragem por substituição e tem problemas por ser muito fácil de decifrar. Por exemplo, para uma dada chave, a letra A será sempre substituída pelo mesmo símbolo. E, se a letra A for a letra que aparece (estatisticamente) em textos em Português, fica fácil de descobrir o tamanho da chave e, como consequência, decifrar todo o texto.

Para tornar o algoritmo de cifragem mais forte (mais difícil de ser quebrado) a chave pode ser alterada na substituição de cada símbolo. Vamos mostrar como fazer isso em um pequeno exemplo, supondo (para simplificar) que os símbolos sejam apenas as letras maiúsculas e a mensagem seja: SOCORRO.

A cifra de César para D=3 seria: VRFRUUR

Com a chave variável, o primeiro caractere seria substituído utilizando a chave D=3 e, para cada nova substituição o valor da chave será incrementado de 2 unidades. A cifra com chave inicial D=3 ficaria: VTJXCED

Note que as letras iguais (O e R) foram substituídas por símbolos diferentes: O virou T, X e D e R virou C e E.

Para decifrar o texto é necessário conhecer o valor inicial da chave e a lógica de variação do valor da chave. Repare que a operação de substituição usará a operação inversa, considerando a distância no sentido oposto. Ou seja, se na cifragem a substituição foi feita utilizando a soma (caminhando para frente), na decifragem a substituição deverá utilizar a subtração.

Porém, a operação feita para variar a chave deverá ser a mesma. No nosso exemplo, a substituição do primeiro símbolo foi feita com distância 3 e do segundo símbolo com distância 5, depois 7, 9, 11 13 e 15.

Implemente um programa que execute as seguintes tarefas:

- ler uma frase qualquer digitada pelo usuário, armazenando-a em uma variável de nome frase:

```
frase = input(('Digite uma frase: ')  # este nome é obrigatório
```

- ler um valor de chave que deve estar entre 1 e 50.
- criar um arranjo para receber os caracteres gerados pela substituição da Cifra de César, com chave fixa;
- criar um segundo arranjo para receber os caracteres gerados pela substituição, com chave variável, sendo o valor inicial o mesmo valor definido pelo usuário. Após cada substituição a chave deverá ser alterada usando o comando:

```
chVar = chVar + 2  # o nome é só uma sugestão. Pode usar outro qualquer
```

- após executar as duas cifragens, o comando abaixo (para apagar a frase digitada pelo usuário) deverá ser colocado em seu código:

```
frase = ‘’  # string nula
```

- na sequência, o programa deve criar mais um arranjo para receber a decifragem do texto.
- o programa deverá utilizar o arranjo com o resultado da segunda cifragem (utilizando chave variável) para gerar o texto decifrado.
- Ao final da execução, o programa deverá perguntar ao usuário se ele deseja executar novamente. Se a resposta for sim, o programa deverá executar mais uma vez.

A figura abaixo mostra uma execução do programa, onde o usuário pediu para repetir a execução apenas 1 vez:

```
>>> q1()

Digite uma frase: cifragem com chave fixa é fraca!!!!!!!!!!! ABCDEF 0123456789
Entre com o valor da chave (1-50): 1

Chave fixa: djgsbhfn!dpn!dibwf!gjyb!ê!gsbdb"""""""""""!BCDEFG!123456789:
Chave var.: dlkyjrr|1v⟪9~⟯¯❥❨E  ▭£xOĚS▭©▭    `bdfhjlnprtu★▭  ¡¤§⟯\ ★▭  ¡¤§ª-°
Frase dec.: cifragem com chave fixa é fraca!!!!!!!!!!! ABCDEF 0123456789
Executar novamente (S/N)? : s

Digite uma frase: 10 As + 10 asteriscos: AAAAAAAAAA **********
Entre com o valor da chave (1-50): 10

Chave fixa: ;:*K}*5*;:*k}~o|s}my}D*KKKKKKKKKK*4444444444
Chave var.: ;<.Q⟩4A8KL>^\ ★⟯▭▭▭▭ ▭¥nVy{}¨^⟫❯❱⟯lxz|~¯⟪❮❨(
Frase dec.: 10 As + 10 asteriscos: AAAAAAAAAA **********
Executar novamente (S/N)? : N
>>>
```

<u>OBS: o algoritmo de cifragem e decifragem apresentados na questão 2 são simplificações grosseiras e podem não funcionar, caso a frase tenha caracteres diferentes dos mais comumente utilizados.</u>

☞ <u>A saída do seu programa (em ambas as questões) deve obedecer à formatação **exata** mostrada nos exemplos (certifique-se que a largura da janela do Shell será suficiente para a exibição correta).</u>

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p08.py**) através do sistema do LBI.

<u>A entrega deverá ser feita até às 23h59 do dia 12/11/2020 (5ª. feira)</u>