Folha de apoio à aula prática 6 (exercícios para praticar antes da aula).

6.1 Escreva duas definições da função `conta_letras(txt)` que retorna o número de letras (maiúsculas ou minúsculas) sem acentos da cadeia de caracteres `txt`; numa das versões compare os carateres de `txt` com `'a', 'A', 'z', 'Z'` e na outra utilize `string.letters`. Exemplo:

```
>>> conta_letras('Ola, mundo!')
8
```

6.2 Implemente a função `apenas_letras(txt)` que testa se uma cadeia de caracteres contém apenas letras maiúsculas ou minúsculas (sem acentos). O resultado deve ser `True` ou `False`. Exemplos:

```
>>> apenas_letras("Abracadabra")
True
>>> apenas_letras("Ola, mundo!")
False
```

6.3 Escreva uma definição da função `filtra_letras(txt)` que, dada uma cadeia de caracteres `txt`, retorna uma cadeia com apenas as suas letras maiúsculas ou minúsculas. Exemplo:

```
>>> filtra_letras('Ola!, -- disse ele...')
'Oladisseele'
```

6.4 Escreva uma função `inversa(txt)` que retorne a cadeia de caracteres dada por ordem inversa. Implemente uma versão com um ciclo for e outra com o operator "fatia" *(slice)*. Exemplo:

```
>>> inversa('Ola Mundo!')
'!odnuM alO'
```

6.5 Uma cadeia de carateres é um *palíndromo* se as sequências obtidas lida da esquerda para a direita e vice-versa são iguais, independentemente das letras serem maiúsculas ou minúsculas. Exemplo: `"reviveR"` é um palíndromo.

Escreva uma definição da função `palindromo(txt)` que verifica se uma cadeia de caracteres é um palindromo; o resultado deve ser `True` ou `False`.

6.6 Mais geralmente, uma cadeia é um palíndromo se se lê da mesma forma nos dois sentidos, ignorando os espaços entre letras, sinais de pontuação e/ou a troca de maiúsculas e minúsculas. Assim, os exemplos seguintes são palíndromos:

```
"Amora me tem aroma."
"Madam, I'm Adam."
"A man, a plan, a canal: Panama"
```

Escreva uma função `palindromo_geral(txt)` para testar se uma cadeia de caracteres `txt` é um palíndromo neste sentido mais geral.

Sugestão: pode resolver este problema combinando o método `lower()` de cadeias de carateres e as soluções dos exercícios 6.3 e 6.5.

6.7 Escreva uma função `rem_espacos(txt)` que remova dois ou mais espaços seguidos numa cadeia de caracteres `txt`, substituindo-os por um único espaço; outros caracteres devem ficar inalterados. Exemplo:

```
>>> rem_espacos(' Ola,     Mundo   !')
' Ola, Mundo !'
```

6.8 A cifra de *Cesar* consiste em substituir cada carater alfabético de uma mensagem pelo carater que está k posições à sua direita, na ordem alfabética. Escreva a função `cesar(k,txt)` que retorna o valor cifrado de `txt` usando a "chave" `k`.

6.9 A cifra de *Vigenère* é uma variação um pouco mais segura da cifra de César que usa uma palavra-chave em vez de um deslocamento único [1]. Começamos por repetir a palavra-chave (e.g., "LUAR") ao longo do texto da mensagem; cada letra da chave de 'A' a 'Z' fazemos corresponder um índice de deslocamento de 0 a 25 (e.g., "LUAR" corresponde aos deslocamentos 11, 20, 0 e 17). Assim, o texto "ATAQUEDEMADRUGADA" seria cifrado em "LNAHFYDVXUDIFAAUL":

A	T	A	Q	U	E	D	E	M	A	D	R	U	G	A	D	A
L	U	A	R	L	U	A	R	L	U	A	R	L	U	A	R	L
L	N	A	H	F	Y	D	V	X	U	D	I	F	A	A	U	L

Escreva uma função `vignere(chave,txt)` que implemente esta cifra.

6.10 Escreva a função `remove_dup(txt)` que remove todas as ocorrências de dois ou mais carateres idênticos seguidos na *string* `txt`, substituindo-os por uma única ocorrência; os restantes caracteres devem ficar inalterados.

Por exemplo, o resultado de `remove_dup("abbcdddeab")` é `abcdeab`.

Nota: não é necessário usar métodos da classe `str`.

6.11 Escreva a função `remove_py_com(txt)` que remove comentários de linhas de código Python, i.e., os sinais de cardinal `#` e tudo o que estiver à sua direita. Note que se o cardinal estiver dentro de uma *string* não é um comentário (considere apenas as *strings* delimitadas por aspas (")).

Por exemplo, o resultado de `remove_py_com("def f(x): # f function ")` é `def f(x):` .

Sugestão: use ciclos `while`. *Nota: não é necessário usar métodos da classe* `str`.

6.12 Escreva a função `remove_c_com(txt)` que remove comentários de linhas com código na linguagem C, i.e., texto entre os sinais `/*` e `*/` inclusive. Ignore a possibilidade desses sinais estarem dentro de *strings*, e caso um comentário não seja "fechado" considere que se estende até o fim da linha.

Por exemplo, o resultado de `remove_c_com("#define PI 3.1415926 /* valor de pi")` é `#define PI 3.1415926` .

Nota: não é necessário usar métodos da classe `str`.

6.13 Escreva a função `remove_cpp_com(txt)` que remove comentários de linhas de código C++, i.e., os sinais `//` e tudo o que estiver à sua direita. Note que se esses sinais estiverem dentro de uma *string* não se trata de um comentário (considere apenas as *strings* delimitadas por aspas (")).

Por exemplo, o resultado de `remove_cpp_com("int main(void) {} // main function")` é `int main(void) {}` .

Sugestão: use ciclos `while`. *Nota: não é necessário usar métodos da classe* `str`.

6.14 Escreva a função `format_pi(n)` que, dado um inteiro $n \leq 10$ devolve a *string* `"pi com ... casas decimais é ..."`, onde as primeiras reticências devem ser substituídas pelo valor de n e as segundas pelo valor de π (tal como definido no módulo `math`) apresentado com n casas decimais.

Nota: a função deve devolver uma string *mas não a deve imprimir.*

Na aula terá exercícios de avaliação contínua baseados nesta folha.

[1] Ver `https://pt.wikipedia.org/wiki/Cifra_de_Vigenère`