Atividade Aula 3 – Parte B #ilovepython ;-)

Em textos disponíveis na Internet é muito comum o uso de hashtags e emoticons. Um pesquisador da área de linguística está interessado em medir o quanto o uso destes elementos é importante para a interpretação destes textos. Para isso ele irá entregar os textos originais para um grupo de leitores e os mesmos textos sem emoticons e hashtags para outro grupo. Finalmente, ele irá comparar as impressões dos grupos sobre os textos.

Sua tarefa será escrever a versão inicial de um programa em Python para auxiliar o pesquisador a filtrar os textos. Adotaremos as seguintes regras simplificadas para a classificação dos elementos:

- **Palavra**: sequência de letras (sem acento).
  Exemplos: `UNIP estrutura de dados`
- **Número**: sequência de dígitos precedidos ou não do sinal negativo (-).
  Identificaremos números inteiros, não tratando números racionais, reais, complexos ou com pontos indicando a separação em grupos de três dígitos.
  Exemplos: `2020 -100000`
- **Hashtag**: caractere # seguido de letras.
  Exemplos: `#python #unip #amoviajar`
- **Emoticon**: sequência de um ou mais caracteres que não se enquadra nas descrições anteriores. Emoticons são compostos principalmente por caracteres de pontuação, mas podem conter letras, números ou serem precedidos por #.
  Exemplos: `:-) #:-) :D <3`

Descrição da entrada

Nesta versão inicial, a entrada virá pré-processada em uma linha contendo uma sequência de elementos separados por espaços em branco. Veja o exemplo abaixo:

```
#ilovepython <3 amo programar :-)
```

Descrição da saída

A saída conterá listas dos elementos classificados, respeitando a ordem da entrada. Para a entrada usada como exemplo a saída será:

```
Palavra(s): amo programar
Numero(s):
Hashtag(s): #ilovepython
Emoticon(s): <3 :-)
```

Cuidado com os espaços em branco!!!!

Nenhum espaço em branco deve ser escrito ao final das listas!
Observe abaixo quais são os caracteres que devem ser escritos para o exemplo utilizado anteriormente:

```
Palavra(s): amo programar
Numero(s):
Hashtag(s): #ilovepython
Emoticon(s): <3 :-)
```

Testes

Esta tarefa contém 7 testes abertos e 3 testes fechados. Todos os elementos a serem processados estão de acordo com as regras apresentadas na primeira seção.

a) Entrada:	`#UNIP 2020 ciencia da computacao @>--;--`
Saída:	`Palavra(s): ciencia da computacao` `Numero(s): 2020`

	Hashtag(s): #UNIP Emoticon(s): @>--;--
b) Entrada:	Python if elif else True False while for def
Saída:	Palavra(s): Python if elif else True False while for def Numero(s): Hashtag(s): Emoticon(s):
c) Entrada:	-10 -5 0 5 10 15 20
Saída:	Palavra(s): Numero(s): -10 -5 0 5 10 15 20 Hashtag(s): Emoticon(s):
d) Entrada:	#tbt #melhordodia #ilovepython
Saída:	Palavra(s): Numero(s): Hashtag(s): #tbt #melhordodia #ilovepython Emoticon(s):
e) Entrada:	:-) :) #:-) @:-) 8-) :-D =-) :-(>:-(;-) :-/
Saída:	Palavra(s): Numero(s): Hashtag(s): Emoticon(s): :-) :) #:-) @:-) 8-) :-D =-) :-(>:-(;-) :-/
f) Entrada:	feliz :-) superfeliz :-D triste :-(bravo >:-(piscando ;-) perplexo :-/
Saída:	Palavra(s): feliz superfeliz triste bravo piscando perplexo Numero(s): Hashtag(s): Emoticon(s): :-) :-D :-(>:-(;-) :-/
g) Entrada:	#AmoChocolate cacau 70 amargo #foradieta :D
Saída:	Palavra(s): cacau amargo Numero(s): 70 Hashtag(s): #AmoChocolate #foradieta Emoticon(s): :D

Releia, se necessário, as instruções para fazer os testes.

Dicas de Python 3 para esta tarefa:

- Para ler a linha com os elementos a serem classificados e montar uma lista de strings podemos utilizar a função `split()`:

  ```
  lista = input().split()
  ```

- Para inserir um elemento em uma lista, utilize a função `append()`:

  ```
  lista_palavras.append(palavra)
  ```

- Para operar com o primeiro elemento de uma string `s` escreva `s[0]`.
- Para operar com uma substring criada a partir da remoção do primeiro elemento da string `s` escreva `s[1:]`.
- Para verificar se uma string `s` contém apenas dígitos use a função `s.isdigit()`.
- Para verificar se uma string `s` contém apenas letras use a função `s.isalpha()`.