Processamento de Linguagens

Ficha Introdutória para as Aulas Práticas sobre ER

Ano Lectivo de 20/21

Problema 0: Treino de ERs

O objetivo deste primeiro exercício é duplo. Por um lado, pretende-se que explore o conceito de Expressão Regular (RE) e seus operadores (básicos ou estendidos) para descrever padrões de texto; por outro lado, pretende-se que recorra ao módulo '`re`' do Python para pesquisar a ocorrência de strings dentro de uma linha que obedeçam a cada um dos seguintes padrões (para facilitar o trabalho explore os programas anexos '`basic0.py`' a '`basic5a.py`'):

1. a palavra '`hello`' no início da linha (basic0)
2. a palavra '`hello`' em qualquer posição da linha (basic1)
3. todas as ocorrências da palavra '`hello`' dentro da linha, admitindo que a palavra seja escrita com maiúsculas ou minúsculas (basic2)
4. todas as ocorrências da palavra '`hello`' dentro da linha substituindo cada uma por '*YEP*' (basic3, basic4)
5. todas as ocorrências do carater vírgula, separando cada parte da linha por esse carater (basic5, basic5a)

Para consolidar os conhecimentos anteriores e treinar outras ER, realize as várias sugestões de pesquisa de um padrão dentro de uma linha recorrendo à função genérica '`busca(ER)`' proposta no ficheiro anexo '`search-geral.py`'.

Experimente ainda calcular a Nota Média de cada aluno, sendo que a informação de cada aluno (número, nome e notas) é fornecida numa linha sendo os vários campos separados por ';' (explore o programa anexo '`basic6.py`').

Problema 1: Alien username

Numa galáxia distante, num planeta muito diferente do nosso, cada utilizador dum computador tem um identificador que segue o seguinte formato:

- Começa com um "underscore", '_', ou um ponto, '.';
- Após o primeiro caráter devem seguir-se um ou mais dígitos;
- Após um certo número de dígitos devem seguir-se três ou mais letras, maiúsculas e/ou minúsculas;
- O nome do utilizador pode ser, opcionalmente terminado por um "underscore", '_'. Caso isso não aconteça, o último caráter deve ser uma letra.

Dada uma lista de nomes de utilizadores, um por linha, verifica quais são válidos. Para esses, imprime "VÁLIDO"numa linha. Para os outros casos imprime "INVÁLIDO".
Exemplo de input:

```
_0898989811abced_
_abce
_09090909abcD0
```

Exemplo de output:

```
VÁLIDO
INVÁLIDO
INVÁLIDO
```

Adicionalmente explore o programa anexo '`usernames.py`'.

Problema 2: Endereços IP

Dada uma lista de linhas em que cada uma deve conter um endereço IP, IPv4 ou IPv6, supostamente válido, desenvolva um filtro de texto que identifica o tipo de endereço IP ou acusa um erro.

Os endereços IPv4 foram os primeiros endereços a serem usados na Internet e eram constituídos por 4 bytes. O formato normal é "A.B.C.D"em que A, B, C e D são inteiros compreendidos entre 0 e 255 inclusive.

O IPv4 limitava o número de endereços a um número baixo para as necessidades de hoje em dia e assim surgiu o IPv6. Com 128 bits, veio permitir um maior espaço de endereçamento. Os 128 bits dum endereço IPv6 são representados em 8 grupos de 16 bits cada um. Cada grupo é representado por 4 dígitos hexadecimais e cada grupo é separado do seguinte por ':'. Por exemplo: "2001:0db8:0000:0000:0000:ff00:0042:8329". Os zeros à esquerda podem ser omitidos. Um endereço contendo "...:0:..."ou "...:5:..."é idêntico a "...:0000:..."ou "...:0005:....".

Exemplo de input:

```
Esta é uma linha de lixo.
121.18.19.20
2001:0db8:0000:0000:0000:ff00:0042:8329
epl.di.uminho.pt 193.136.19.129
```

Exemplo de output:

```
Erro
IPv4
IPv6
Erro
```

Problema 3: Latitude e Longitude

Dada uma lista de pares de coordenadas, latitude e longitude, supostamente válidos, especifica um filtro de texto que faz a sua verificação atendendo aos seguintes requisitos:

- Cada par de coordenadas deverá seguir a forma: (X, Y) onde X e Y são números decimais (parte inteira e uma parte decimal opcional);
- O valor de X deverá estar entre -90 e $+90$;
- O valor de Y deverá estar entre -180 e $+180$;
- Quer X quer Y podem ser opcionalmente precedidos por um sinal $+$ ou $-$;
- Tem de haver um espaço entre Y e a vírgula precedente;
- Não há espaço entre X e o parentesis precedente nem entre Y e o parentesis que se lhe segue;
- Não poderá haver zeros desnecessários à esquerda nas partes inteiras.

Exemplo de input:

```
(75, 180)
(+90.0, -147.45)
(77.11112223331, 149.99999999)
(+90, +180)
(90, 180)
(-90.00000, -180.0000)
(75, 280)
(+190.0, -147.45)
(77.11112223331, 249.99999999)
(+90, +180.2)
(90., 180.)
(-090.00000, -180.0000)
```

Exemplo de output:

```
Válido
Válido
Válido
Válido
Válido
Válido
Inválido
Inválido
Inválido
Inválido
Inválido
Inválido
```

Problema 4: Matrículas de outro mundo

Num país algures, as matriculas seguem os seguintes requisitos:

- Uma matrícula tem 8 dígitos;
- Os 8 dígitos estão divididos em 4 partes iguais de 2 dígitos por um separador que pode ser '...', '-' ou ':';
- Cada matrícula só pode ter uma espécie de separador;
- Os separadores têm de ter dígitos antes e depois, não há espaços.

Desenvolva um filtro de texto que apanhe e contabilize as matriculas num texto.