# Programação II

## Exercícios 9
## Expressões Regulares em Python

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Tecnologias da Informação

2020/2021

1. Para cada um dos casos abaixo indique quais as strings ou substrings que são reconhecidas pela expressão regular indicada.

   (a) Expressão regular: `r'^[a-z]*[0-9]+.$'`

      i. `'33'`
      ii. `'2'`
      iii. `'carro2'`
      iv. `'carro234'`
      v. `'_25'`

   (b) Expressão regular: `r'^[1-4]\d{3}[^a-z]$'`

      i. `'12345'`
      ii. `'5392A'`
      iii. `'1274a'`
      iv. `'2461_'`
      v. `'4a221'`

   (c) Expressão regular: `r'([a-z0-9]{2}|[A-Z])\d-?'`

      i. `'456'`
      ii. `'ab-'`
      iii. `'F44'`
      iv. `'23A5-'`
      v. `'aZ3x-'`

2. Para cada uma das expressões regulares, indique quantas (e quais) ocorrências serão detetadas nas strings indicadas.

(a) Expressão regular: `r'[a-z0-9]\d'`
   i. `'a456'`
   ii. `'456'`
   iii. `'456b'`
   iv. `'ab456'`
   v. `'a5b2d456'`

(b) Expressão regular: `r'.[^0-9]*,?[^a-zA-Z]*'`
   i. `'234abc'`
   ii. `'abc234'`
   iii. `'234,abc'`
   iv. `',234abc'`
   v. `'xabc,999x'`
   vi. `'xabc999y'`
   vii. `'0'`
   viii. `'00,99'`
   ix. `'\n\n\n'`

3. Para cada um dos casos abaixo escreva uma expressão regular Python.

   (a) Os códigos postais de Portugal. Exemplos: `1749-016 Lisboa`, `2795-241 Linda a Velha`.

   (b) As matrículas de veículos registados em Portugal. Exemplos: `AA-11-11`, `11-AA-11`, `11-11-AA`, ou `AA-11-AA`.

   (c) Um número em formato vírgula flutuante. Exemplos: `2.3` e `-1.345`. Não exemplos: `4` e `0034.0` e `-0.0`.

   (d) Um número escrito na notação científica. Exemplos: `2e4`, `2.3e4` e `-1.345e-34`. Não exemplos: `4`, `-03.0e7`, `0.2e2`, para além de todos os da alínea anterior.

   (e) Os endereços de email dos alunos da FCUL. Exemplo: `fc99999@alunos.fc.ul.pt`.

   (f) Identificadores numa linguagem de programação: uma sequência de letras, algarismos e traços inferiores (_) que não começam por um algarismo.

   (g) Versão simplificada de URL (*Uniform Resource Locator*, wikipedia), da forma

   `scheme:[//[user:password@]host[:port]]`

   onde: os parêntesis retos indicam partes opcionais; `scheme` é uma sequência de carateres que começa com uma letra minúscula, seguida por qualquer combinação de letras minúsculas, números, mais (+), ponto (`.`) ou hífen (-); `user` e `passwd` são sequências não vazias de carateres; `host` é um nome de um domínio ou um endereço IP (em notação decimal-ponto); e `port` é o número de um porto (um a quatro dígitos).

(h) Um número romano. Exemplos: `MM`, `CCI`, `CM`, `XLVII`. Não exemplos: `IM`, `DIC`. Considere apenas números romanos entre 1 e 3999.

4. Baseado nas soluções do exercício anterior, escreva funções Python que, dada uma *string*, devolvam a informação indicada. As funções devem devolver `None` caso a string fornecida não corresponda a um valor bem formado de acordo com a expressão regular.

   (a) Um triplo contendo as três partes de um código postal.

   ```
   >>> cod_postal('2795-241 Linda a Velha')
   ('2795', '241', 'Linda a Velha')
   ```

   (b) Os três constituintes de uma matrícula, juntamente com a informação sobre o tipo de matrícula: letras-primeiro, letras-no-meio, letras-no-fim, números-no-meio.

   ```
   >>> constituintes_matricula('AA-11-11')
   ('letras-primeiro', 'AA', '11', '11')
   ```

   (c) Um número em vírgula flutuante.

   ```
   >>> numero_vf('2.3')
   2.3
   ```

   (d) Um dicionário com os vários elementos constantes num URL.

   ```
   >>> componentes_URL("http://aaa:bbb@ciencias.
       ulisboa.pt")
   {'scheme': 'http', 'user': 'aaa', 'password': '
       bbb', 'host': 'ciencias.ulisboa.pt', 'port':
       None}
   ```

   (e) Um número inteiro, em numeração árabe, correspondente a um dado número romano.

   ```
   >>> romano_para_inteiro('XLVII')
   47
   ```

5. Escreva uma função que agrupe os algarismos de um número de telefone em três.

   ```
   >>> num_telefone("212345123")
   '212 345 123'
   ```

6. Extração de padrões. Para cada alínea escreva uma função que:

   (a) Devolva uma lista com todas as ocorrências de uma expressão regular numa *string*.

```
>>> todas_as_ocorrencias(r'[a-z0-9]\d','a5b2d456'
    )
['a5', 'b2', 'd4', '56']
```

(b) Devolva uma lista com todas as ocorrências de uma expressão regular numa lista de *strings*.

```
>>> todas_as_ocorrencias_lista(r'[a-z0-9]\d',['
    a5b2d456','a456b'])
['a5', 'b2', 'd4', '56', 'a4', '56']
```

(c) Calcule o número de matrículas que ocorrem numa lista de *strings*.

```
>>> acidente1 = 'AF-12-70␣colidiu␣com␣43-PP-98␣
    que␣embateu␣em␣98-00-AP'
>>> acidente2 = '43-PQ-98␣colidiu␣com␣43-PJ-98␣
    que␣embateu␣em␣AG-00-11'
>>> numero_de_matriculas([acidente1, acidente2])
6
```

(d) Obtenha a lista de matrículas que ocorrem numa lista de strings, mas apenas aquelas em que um par de letras começa por uma letra dada.

```
>>> lista_matriculas([acidente1,acidente2],'A')
['AF-12-70', '98-00-AP', 'AG-00-11']
```

7. Substituição de *strings* em ficheiros.

   (a) Escreva uma função que aplique uma dada substituição no conteúdo de um ficheiro. O resultado deve ser escrito num outro ficheiro.

   (b) Utilize o resultado da alínea anterior para substituir vírgulas por ponto e vírgulas num ficheiro CSV.

   (c) Idem para reescrever os nomes das estações do ano em minúscula (de modo a ficar de acordo com a nova ortografia).

   (d) Escreva uma função que elimine de um ficheiro a ocorrência de múltiplos espaços em branco entre duas palavras.

8. Escreva uma função que, dada uma lista de *strings*, devolva um dicionário com cinco campos: número de matrículas do tipo letras-primeiro, número de matrículas do tipo letras-no-meio, número de matrículas do tipo letras-no-fim, número de matrículas do tipo números-no-meio, e número de *strings* que não representam matrículas.

```
>>> ll = ['AF-12-70', '98-00-AP', 'AG-00-11', '43-PP
    -98', '43-PQ-98', '43-PJ-98', 'AB-44-HQ', 'dd-333
    ']
```

```
>>> dic_matriculas(ll)
{'letras-primeiro': 2, 'letras-no-meio': 3, 'letras-
    no-fim': 1, 'numeros-no-meio': 1, 'invalida': 1}
```

9. Escreva uma função, que dado uma lista de números de telefone (em formato *string*), os classifique de acordo com o Plano Nacional de Numeração. Para tal utilize um dicionário em que as chaves são os vários indicativos (`'00'`, `'1'`, `'2'`, ..., `'9'`, `'invalido'`) e os valores são listas de números de telefone (em formato *string*).

```
>>> classifica_numeros(['123456789','234443334','
    003434554','2232323','awwa'])
{'1': ['123456789'], '2': ['234443334'], '00': ['
    003434554'], 'invalido': ['2232323', 'awwa']}
```