Instituto Nacional de Pesquisas Espaciais
Curso de Pós-Graduação em Sensoriamento Remoto

SER-347 – Introdução à Programação para Sensoriamento Remoto
Lista de Exercícios 04

Dr. Gilberto Ribeiro de Queiroz *(gilberto.queiroz@inpe.br)*
Dr. Thales Sehn Körting *(thales.korting@inpe.br)*
Dr. Fabiano Morelli *(fabiano.morelli@inpe.br)*
23 de abril de 2018

Introdução à Programação com a Linguagem Python - Lista 04

Exercícios

Atenção:

1. Os exercícios práticos devem ser desenvolvidos em Python. Escreva a documentação que achar pertinente dentro do próprio código fonte, que deverá utilizar a codificação de caracteres UTF-8.

2. A solução de cada exercício deverá ser entregue em um único arquivo de código fonte na linguagem Python. Use arquivos com a extensão `.py` com a seguinte nomenclatura: exercicio-{numero}.py. Ex: `exercicio-01.py`.

3. Envie por e-mail **um único** arquivo no **formato zip**, chamado `lista04.zip`, contendo todos os arquivos de código fonte dos exercícios.

4. O título do e-mail deve seguir o seguinte padrão[1]:
   `[ser347-2018][lista-04] nome-completo-aluno`.

5. O endereço de entrega da lista é: ser347@dpi.inpe.br.

6. **Prazo para entrega:** 02/05/2018 - 22:00

[1] Não use acentos ou caracteres especiais no nome do arquivo.

Exercício 01. Considere a série temporal mostrada abaixo, extraída de dados do sensor MODIS, produto MOD13Q1, para a localização (-54,-12) para o período de 01/01/2015 a 19/12/2015:

```
red_values = ( 168, 398, 451, 337, 186, 232, 262,
               349, 189, 204, 220, 220, 207, 239,
               259, 258, 242, 331, 251, 323, 106,
               1055, 170 )

nir_values = ( 2346, 4431, 4638, 4286, 2752, 3521,
   2928, 3087, 2702, 2685, 2702, 2865,
   2835, 2955, 3019, 3391, 2986, 4042,
   3050, 3617, 2478, 3361, 2613 )

timeline = ( "2015-01-01", "2015-01-17", "2015-02-02",
 "2015-02-18", "2015-03-06", "2015-03-22",
 "2015-04-07", "2015-04-23", "2015-05-09",
 "2015-05-25", "2015-06-10", "2015-06-26",
 "2015-07-12", "2015-07-28", "2015-08-13",
 "2015-08-29", "2015-09-14", "2015-09-30",
 "2015-10-16", "2015-11-01", "2015-11-17",
 "2015-12-03", "2015-12-19" )
```

Faça um programa que:

- Crie a série temporal do NDVI (Obs.: Multiplicar os valores de red e nir por 0.0001 (ver MOD13Q1)).
- Calcule a média do NDVI dessa série temporal.
- Obtenha o maior e menor valores de NDVI e escreva a data em que eles ocorreram.

Você deverá usar laços do tipo `for` para resolver os itens deste exercício. Não utilize funções prontas do Python nem o NumPy.

Exercício 02. Para melhorar a visualização de uma banda de uma imagem de sensoriamento remoto, podemos usar o *offset* e/ou o *ganho*. *offset* é um número real que é somado a todos os pixels de uma banda. *ganho* é um número real que é multiplicado por todos os pixels da banda. A aplicação é feita conforme a equação a seguir:

$$i_2 = i_1 \times G + O$$

onde i_1 é a imagem original, i_2 é a imagem resultante da aplicação de um *offset* (O) e de um *ganho* (G).

Faça um programa que defina uma banda como um array NumPy de 500 x 500 pixels (com valores aleatórios inteiros entre 0 e 255), e aplique um *offset* e um *ganho* informados pelo usuário.

Veja que o valor resultante deste cálculo pode ultrapassar o range da imagem. Neste caso, force o resultado para ficar dentro do range (e.g. um resultado -50 será alterado para 0, um resultado 270 será alterado para 255).

Ao final o programa deve desenhar a imagem original, e a imagem com a aplicação do ganho.

Exercício 03. Faça um programa que defina duas bandas (b1 e b2), armazenadas como matrizes NumPy (50 x 50 pixels), e crie uma terceira matriz NumPy (b3) contendo o resultado da soma das duas bandas.

Lembre-se que a imagem final pode ultrapassar o range (por definição, entre 0 e 255), assim é preciso modificar o resultado para ficar dentro deste range.

Exercício 04. Considere um array NumPy assim definido:

```
# definir um array com 3 matrizes de 5 linhas por 4 colunas
meu_array = np.arange(60).reshape((3,5,4))
# alterar os valores da primeira matriz
# somando-o com a segunda matriz
meu_array[0] = meu_array[0] + meu_array[1]
```

Crie um programa que leia um limiar numérico, e crie uma matriz chamada 'filtrada', contendo os valores originais da primeira matriz de `meu_array` (`meu_array[0]`), porém valores acima do limiar serão alterados para zero.

Exercício 05. Veja a seguinte descrição sobre matriz de confusão:

> In the field of machine learning and specifically the problem of statistical classification, a **confusion matrix**, also known as an **error matrix**, is a specific table layout that allows visualization of the performance of an algorithm (...). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (*i.e.* commonly mislabelling one as another). It is a special kind of contingency table, with two dimensions ('actual' and 'predicted') (...).

Como exemplo, apresentamos a matriz de confusão da Figura 1. A leitura desta matriz pode ser feita da seguinte forma. Os elementos marcados em *Reference* são os elementos com a classe conhecida. Os elementos marcados em *Classification* são os elementos que foram classificados por, por exemplo, algum algoritmo de reconhecimento de padrões.

Assim, 40 elementos da classe b (*Annual Crop*) foram classificados erroneamente na classe a (*Perennial crop*). Pela mesma lógica, 6303 elementos da classe c (*Planted forest*) foram corretamente classificados.

Faça um programa em que receba a matriz de confusão da figura, em formato NumPy (exceto as linhas/colunas *Total*), e calcule:

		Reference				
		a	**b**	**c**	**d**	**Total**
Classification	**a**	**6563**	40	184	127	6914
	b	3	**717**	7	10	737
	c	265	17	**6303**	40	6625
	d	26	21	4	**1916**	1967
	Total	6857	795	6498	2093	16243

Figura 1: Matriz de confusão, onde (a) *Perennial crop*, (b) *Annual crop*, (c) *Planted forest* and (d) *Semi-perennial crop*

.

1. total de elementos de referência de cada classe;
2. taxa de acerto de cada classe;
3. taxa de acerto total.