Curso de Extensão: **Introdução à Linguagem de Programação Python**

Avaliação (opcional) / Gabarito - Go, 06/07/19 - Prof. Cláudio

Questões:

1. Identifique, comente e indique o site oficial (nos casos possíveis) de cada um dos seguintes elementos da Ling. Python:
   a. IPython: uma casca Python avançada; http://ipython.scipy.org/moin/
   b. Numpy: fornece objetos do tipo arranjo numérico poderosos, e rotinas para manipulá-los; http://www.numpy.org/
   c. Scipy: rotinas de processamento de dados de alto níveis. Otimização, regressão, interpolação etc.; http://www.scipy.org/
   d. Matplotlib: visualização 2D de dados, fornece gráficos prontos para publicações; http://matplotlib.sourceforge.net
2. Forme o seguinte arranjo 2D (sem digitá-lo explicitamente):

```
[[ 1, 6, 11],
 [ 2, 7, 12],
 [ 3, 8, 13],
 [ 4, 9, 14],
 [ 5, 10, 15]]
```

```
import numpy as np
a = np.resize(np.arange(1,16),(3,5)).T
print(a)
```

3. Gere uma matriz 10 x 3 de números aleatórios (no intervalo [0,1], distribuídos normalmente). Para cada linha, escolha o número mais próximo de 0,5.
   Use `abs` e `argsort` para encontrar a coluna `j` com o elemento mais próximo em cada linha. Use uma indexação sofisticada para mostrar os números e as respectivas colunas.

```
a = np.random.normal(0,1,(10,3))    # maiores que 1
inf = 0.; sup = 1.; med = 0.; dp = 1.
b = np.random.normal((inf-med)/dp,(sup-med)/dp,(10,3)) # 0 a 1
print(b)
ind_ord = np.argsort(np.abs(b-0.5))
print('\nElemento mais próximo   Coluna')
for lin in range(b.shape[0]):
  print("\t%8.5f\t%2d" % (b[lin,ind_ord[lin,0]],
int(ind_ord[lin,0])))
```

4. Crie uma matriz da imagem da Lena, com uma centralização mais estreita, removendo 30 pixels de todas as bordas da imagem. Apresente o resultado, exibindo essa nova imagem com `imshow()`.

```
import pylab as plt
import numpy as np
imagem='E:\\python\\curso_ifg\\jun19\\2_Packages\\img\\lena.png'
a = np.array(plt.imread(imagem))
brd = 30                          # borda
b = a[brd:-brd,brd:-brd]          # imagem recortada
plt.subplot(121); plt.imshow(a,plt.cm.gray);
plt.title('Original'); plt.xticks([]); plt.yticks([])
```

```
plt.subplot(122); plt.imshow(b,plt.cm.gray)
plt.title('Zoom'); plt.xticks([]); plt.yticks([])
```

5. Mostre a imagem da Lena com um círculo amarelo cheio, no centro da imagem, com raio de 15% da largura da imagem.

```
import pylab as plt
import numpy as np
imagem = 'E:\\python\\jun19\\2_Packages\\img\\lena.png'
a = np.array(plt.imread(imagem))
lx, ly = a.shape
r = int(np.ceil(0.15 * max(lx,ly)))
circ = plt.Circle((lx/2,ly/2),r,color='y')
plt.subplot(121); plt.imshow(a,plt.cm.gray)
plt.title('Círculo'); plt.xticks([]); plt.yticks([])
ax = plt.gca(); ax.add_artist(circ)
```

6. Dado uma função de variável simples, $f(x)$, estabelecida por uma função de usuário, faça um *script* para localizar o ponto de mínimo no intervalo $-10 < x < 10$.
*Dica*: use a função `minimize()` do subpacote `minimize` do pacote scipy.

```
def f(x):
    return x**2 + 10*np.sin(x)
```

```
from scipy.optimize import minimize
from numpy import arange, sin, pi
from pylab import plot
def f(x):
    return (x%pi)**2 + sin(x)

tempo = arange(0.,6*pi,0.001)
plot(tempo,f(tempo))
res = minimize(f, pi/8, method='BFGS')
print(res)
```

7. Compare dois conjuntos de 1000 amostras gaussianas cada, $N[0,1]$. Mostre o gráfico de barras do histograma dos conjuntos num mesmo plano cartesiano.

```
import numpy as np
import pylab as plt
a = np.random.normal(0,1,100)
b = np.random.normal(0,1,100)
ha, bins_a = np.histogram(a, bins=15)
hb, bins_b = np.histogram(b, bins=15)
larg = 0.7 * (bins_a[1] - bins_a[0])
cent = (bins_a[:-1] + bins_a[1:]) / 2
plt.bar(cent, ha, align='center',width=larg, label='Variável A')
plt.bar(cent, hb, align='center',width=larg, label='Variável B')
plt.legend()
```

Críticas e Sugestões (suas impressões sobre o curso):