Lista de ejercicios 1

Curso: Tópicos de Investigación : Machine Learning CM-072

Lecturas Importantes

1. Notas acerca de ciencia de datos con python escrito por Kunal Jain: `http://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/`
2. `http://composingprograms.com/` es un libro online sobre paradigmas y conceptos de introducción a la Ciencia de la Computación, basados en el Libro de de Abelson y Sussman `Structure and Interpretation of Computer Programs (SICP)`.

Python básico

1. Escribe una función en Python, llamada 'minmax(data)' que toma una secuencia de uno o más números y retorna el menor y el mayor de esos números en forma de tupla de longitud 2. No debes usar la funciones 'min' o 'max' en tu solución.
2. Python permite enteros negativos para ser usados en secuencias, como las cadenas. Si una cadena tiene longitud n y la expresión $s[k]$ es usada para índices $-n \leq k < 0$. ¿ Cuál es el índice quivalente $j \geq 0$ tal que $s[j]$ se refiera al mismo elemento?.
3. Demuestra como usar la sintaxis de las listas por comprensión para producir la lista:

```
> list[0,2,6,12,20,30,42,56, 72, 90]
```

4. El módulo 'random' de Python incluye la función 'choice(data)' que retorna un elemento aleatorio desde una secuencia. Este módulo incluye la función 'randrange' con una parametrización similar a la función 'range' que retorna un valor aleatorio desde un rango dado. Usando solo la función 'randrange', implementa tu propia versión de la función 'choice'.
5. Escribe en Python una función que toma una secuencia de números y determine si todos los números son diferentes uno del otro.
6. Demuestra como usar la sintaxis de las listas de comprensión en Python para producir la lista:

```
> list['a','b','c',.., 'z']
```

   sin tener que escribir los 26 caracteres literalmente.

7. Escribe un programa en Python que lee líneas repetidamente desde la entrada estándar hasta que un `EOFError` es lanzado y luego las salidas de esas líneas en orden inverso.
8. El módulo 'random' de Python incluye la función 'shuffle(data)' que acepta una lista de elementos y los reordena de manera aleatoria de tal forma los posibles ordenamientos ocurren con la misma probabilidad. El módulo 'random' incluye la función 'randint(a,b)' que retorna un entero aleatorio uniformemente desde a hasta b (sin tomar los extremos). Usando solo la función 'randint', implementa tu propia versión de la función 'shuffle'.

9. Un vector $v = (v_1, v_2, \cdots, v_n)$ en un espacio n-dimensional tiene la siguiente norma:

$$\|v\| = \sqrt[p]{v_1^p + v_2^p + \cdots v_n^p}.$$

Realiza una implementación para esta norma con una función `norm(v,p)` que retorna la norma para una lista de números v y un valor p dado.

10. La `paradoja del cumpleaños` dice que la probabilidad de que dos personas en una habitación tengan la misma fecha de nacimiento es más de la mitad, siempre que n, el número de personas en la habitación, es más de 23. Diseña un programa en python, que ponga a prueba esta paradoja con una serie de experimentos en los 'cumpleaños' generados aleatoriamente , que ponen a prueba esta paradoja para $n = 5, 10, 15, 20, \ldots, 100$.

11. Cesar tiene una manera de sumar los valores en una secuencia A de n enteros, donde n es una potencia de dos. El crea una nueva secuencia B de la mitad de tamaño de A y coloca $B[i] = A[2i] + A[2i+1]$ para $i = 0, 1, \ldots, (n/2) - 1$. Si B tiene tamaño 1, entonces la salida es $B[0]$. De otro modo, el reemplaza A con B y repite el proceso. ¿ Cúal es el tiempo de ejecución de su algoritmo?.

12. Sea el código siguiente

```
# Ejemplo de OOP en Python.

import collections

class Vector:
  """Representa un vector en un espacio multidimensional ."""

  def __init__(self, d):
    if isinstance(d, int):
      self._coords = [0] * d
    else:
      try:
        self._coords = [val for val in d]
      except TypeError:
        raise TypeError('tipo de  parametro invalido ')

  def __len__(self):
    """Returna la dimension del vector."""
    return len(self._coords)

  def __getitem__(self, j):
    """Returna la  j-coordenada del vector."""
    return self._coords[j]

  def __setitem__(self, j, val):
    """Coloca la j-coordenada  del vector a un valor dado ."""
    self._coords[j] = val

  def __add__(self, otro):
    """Returna la suma de dos  vectores."""
    if len(self) != len(otro):          # se basa en el metodo  __len__
      raise ValueError('la dimension  debe ser correcta')
    resultado = Vector(len(self))           # empezamos con vectores ceros
    for j in range(len(self)):
      resultado[j] = self[j] + otro[j]
    return resultado
```

```
def __eq__(self, otro):
  """Returna True si el vector tiene las mismas coordenadas del otro vector ."""
  return self._coords == otro._coords

def __ne__(self, otro):
  """Returna True si los vectores difieren uno de otro ."""
  return not self == otro             # se basa en  __eq__

def __str__(self):
  """Produce una representation como cadena del vector ."""
  return '<' + str(self._coords)[1:-1] + '>'

def __neg__(self):
  """Returna una  copia del vector con las coordenadas  son negativas ."""
  resultado = Vector(len(self))           # empezamos con vectores ceros
  for j in range(len(self)):
    resultado[j] = -self[j]
  return resultado

def __lt__(self, otro):
  """Compara vectores basados en el orden  lexicografico ."""
  if len(self) != len(otro):
    raise ValueError('la dimension debe ser correcta')
  return self._coords < otro._coords

def __le__(self, otro):
  """Compara vectores basados en el orden lexicografico."""
  if len(self) != len(otro):
    raise ValueError('la dimension debe ser correcta')
  return self._coords <= otro._coords
```

- Implementa el método `__sub__` para el código anterior tal que la expresión $u - v$ retorna una nueva instancia vector representando la diferencia entre dos vectores.
- Implementa el método `__neg__` para el código anterior tal que la expresión $-v$ retorna una nueva instancia vector cuyas coordenadas son los valores negativos de las coordenadas del vector v.
- Implementa el método `__mul__` para el código anterior tal que la expresón $v * 3$ retorna un nuevo vector con coordenadas que son 3 veces a las coordenadas del vector v.
- Implementa el método `__mulp__` para el código anterior tal que la expresón $u * v$ retorna un escalar que representa el producto escalar de los vectores, esto es $\sum_{i=1}^{n} u_i \cdot v_i$.

Numpy básico

1. Sean los siguientes arrays

   ```
   a = np.array([0, 1, 2, 3])
   b = np.array([[0, 1, 2], [3, 4, 5]])
   c = np.array([[[1], [2]], [[3], [4]]])
   ```

   usa las funciones `len()` y `numpy.shape()` sobre esos arrays. ¿ Cómo están relacionados uno del otro?. ¿Qué hay acerca del atributo `ndim`?.

2. Experimenta con `arange, linspace,ones, zeros, eye` y `diag`.

3. Crea diferentes arrays con números aleatorios:

```
a = np.random.rand(5)
b = np.random.randn(5)
```

prueba con `seed` antes de crear un array con valores aleatorios.

4. Sea

```
np.arange(6) + np.arange(0, 51, 10)[:, np.newaxis]
```

usando `slicing` consigue los siguiente:

```
array([14, 15])
array([[52, 53, 54, 55]])
array([[30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15]])
array([[20, 22, 24]])
array([[20, 22, 24],
       [50, 52, 54]])
array([[ 1,  3,  5],
       [31, 33, 35]])
```

5. Crea los siguientes arrays

```
[[1, 1, 1, 1],
 [1, 1, 1, 1],
 [1, 1, 1, 2],
 [1, 6, 1, 1]]

[[0., 0., 0., 0., 0.],
 [2., 0., 0., 0., 0.],
 [0., 3., 0., 0., 0.],
 [0., 0., 4., 0., 0.],
 [0., 0., 0., 5., 0.],
 [0., 0., 0., 0., 6.]]
```

6. Usa la función `np.tile` para construir el siguiente array:

```
[[4, 3, 4, 3, 4, 3],
 [2, 1, 2, 1, 2, 1],
 [4, 3, 4, 3, 4, 3],
 [2, 1, 2, 1, 2, 1]]
```

7. Multiplica una matriz 5×3 por una matriz 3×2.

8. Escribe un programa en python, que considere una función que genera 10 enteros y los utiliza para construir una matriz.

9. ¿ Qué produce el siguiente código:

```
X, Y = np.meshgrid(np.linspace(-1,1,10), np.linspace(-1,1,10))
D = np.sqrt(X*X+Y*Y)
sigma, mu = 1.0, 0.0
G = np.exp(-( (D-mu)**2 / ( 2.0 * sigma**2 ) ) )
print(G)
```

10. Considera una matriz aleatoria 10×2 representando coordenadas cartesianas. Escribe un script de python que convierta coordenas cartesianas a coordenadas polares.

11. Dado un número arbitrario de vectores, diseña un programa en Python para construir el producto cartesiano.

12. ¿ Qué hace el siguiente código en Python:

```
Z = np.random.randint(0,5,(10,10))
n = 3
i = 1 + (Z.shape[0]-3)
j = 1 + (Z.shape[1]-3)
C = stride_tricks.as_strided(Z, shape=(i, j, n, n), strides=Z.strides + Z.strides)
print(C)
```

13. Considera el vector $[1, 2, 3, 4, 5]$, ¿cómo se puede construir un nuevo vector con 3 ceros consecutivos intercalados entre cada valor?.

14. ¿ Qué hace el siguiente código en Python:

```
def iterate(Z):
    N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
         Z[1:-1,0:-2]                + Z[1:-1,2:] +
         Z[2:  ,0:-2] + Z[2:  ,1:-1] + Z[2:  ,2:])

    n = (N==3) & (Z[1:-1,1:-1]==0)
    s = ((N==2) | (N==3)) & (Z[1:-1,1:-1]==1)
    Z[...] = 0
    Z[1:-1,1:-1][n| s] = 1
    return Z

Z = np.random.randint(0,2,(50,50))
for i in range(100): Z = iterate(Z)
```

15. usando el paquete mayavi, que produce el siguiente código en Python:

```
import numpy as np
x,y=np.ogrid[-1:5:0.5:100j, -1.0:1.0:1000j]
z = x +1j*y
julia =np.zeros.(z.shape)
c = -0.7 - 0.4j

for it in range(1,101):
  z = z**2 + c
  es = z*z.conj() >4
  julia += (1/float(it))*es

from mayavi import mlab
mlab.figure(size=(800,600))
mlab.surf(julia,colormap='gist_ncar', warp_scale='auto',vmax=1.5)
mlab.view(15,30,500,[-0.5,-0.5,2.0])
mlab.show()
```