Lista de Ejercicios de Programación #04

01 - Similaridad del coseno (PCD22)

Escriba una función en el lenguaje Python 3.x que reciba dos vectores en numpy (nd-array) y retorne la similaridad del coseno[1] entre ambos vectores. Recuerde que la similaridad del coseno es un número real que se encuentra entre -1 y 1.

La fórmula es la siguiente:

$$coseno(a,\ b)\ =\ \frac{a.b}{|a||b|}$$

donde a.b es el producto interno entre los vectores a, b y |a| es la norma L2 del vector a.

La función debe tener la siguiente cabecera:

```
coseno(a, b)
```

Nota: Puede asumir que a y b tienen el mismo tamaño.

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/989

Ejemplo:

Ejemplo de código ejecutado	Salida

`a = np.array([0,1])`
`b = np.array([1,0])`
`print(coseno(a,b))`
	`0.0`

Ejemplo de código ejecutado	Salida
`a = np.array([0,1])` `b = np.array([0,2])`	`1.0`

[1] https://en.wikipedia.org/wiki/Cosine_similarity

`print(coseno(a,b))`	

Ejemplo de código ejecutado	**Salida**

`a = np.array([0.5,1,3])`
`b = np.array([-3,2,6])`
`print(coseno(a,b))`
	`0.8254898842683464`

02 - Neurona Artificial (PCD23)

Escriba una función en el lenguaje Python 3.x que reciba dos vectores en numpy (nd-array) representando a las entradas `x` y a los pesos `w` de una neurona artificial y número real `b`. Con estos parámetros deberá calcular el valor de pre-activación de la neurona artificial usando la siguiente fórmula.

$$neurona\_artificial(x,\ w,\ b)\ =\ x.w\ +\ b,$$

donde $x.w$ es el producto interno entre los vectores x y w.

La función deberá tener la siguiente cabecera:

```
neurona_artificial(x, w, b)
```

Nota: Puede asumir que x y w tienen el mismo tamaño.

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/990

Ejemplo:

Ejemplo de código ejecutado	**Salida**

`x = np.array([0,1])`
`w = np.array([1,0])`
`b = 1.0`
`print(neurona_artificial(x,w,b))`
	`1.0`

Ejemplo de código ejecutado	Salida

`x = np.array([3, 4, 5])`
`w = np.array([0.5,0.25, 0.2])`
`b = -1.0`
`print(neurona_artificial(x,w,b))`
	`2.5`

03 - Distancia de Minkowski (PCD24)

Escriba una función en el lenguaje Python 3.x que reciba dos vectores a, b en numpy (nd-array) y el número entero p, parámetro de la distancia de Minkowski[2] usando la siguiente fórmula.

$$minkowski(a,b,p) = \left(\sum_{i=1}^{n} \left|a_i - b_i\right|^p\right)^{\frac{1}{p}}$$

La función deberá tener la siguiente cabecera:

```
minkowski(a,b,p)
```

Nota: Puede asumir que a y b tienen el mismo tamaño.

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/991

Ejemplo:

Ejemplo de código ejecutado	Salida

`a = np.array([0,3])`
`b = np.array([4,0])`
`p = 2`
`print(minkowski(a,b,p))`
	`5.0`

[2] https://en.wikipedia.org/wiki/Minkowski_distance

Ejemplo de código ejecutado	Salida

`a = np.array([0,3])`
`b = np.array([4,0])`
`p = 1`
`print(minkowski(a,b,p))`
	`7.0`

04 - Distancia de Chebyshev (PCD25)

Escriba una función en el lenguaje Python 3.x que reciba dos vectores `a`, `b` en `numpy` (nd-array) y calcule la distancia de Chebyshev[3] usando la siguiente fórmula.

$$chebyshev(a,b) = max\ (\ |a_i - b_i|\ )$$

Como referencia, la distancia de Chebyshev es la máxima diferencia absoluta componente a componente entre dos vectores. Por ejemplo,

```
a = [3, 5]
b = [2, 10]
```

Entonces las diferencias absolutas, componente a componente, serán

```
| a - b | = [1, 5]
```

Y por lo tanto, la máxima diferencia absoluta será 5, entonces

```
chebyshev(a,b) = 5
```

La función deberá tener la siguiente cabecera:

```
chebyshev(a,b)
```

Nota: Puede asumir que a y b tienen el mismo tamaño.

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/992

Ejemplo:

[3] https://es.wikipedia.org/wiki/Distancia_de_Chebyshov

Ejemplo de código ejecutado	Salida

`a = np.array([3, 5])`
`b = np.array([2, 10])`
`print(chebyshev(a,b))`
	`5.0`

Ejemplo de código ejecutado	Salida

`a = np.array([0,3,99,-10])`
`b = np.array([4,0,98,-1])`
`print(chebyshev(a,b))`
	`9.0`

Ejemplo de código ejecutado	Salida

`a = np.array([0,3,99,-10,0])`
`b = np.array([4,0,98,-1,1000])`
`print(chebyshev(a,b))`
	`1000.0`

05 - Promedio Final (PCD26)

Considere que necesita calcular la nota final de `M` estudiantes de un curso, para lo que cuenta con `N` evaluaciones. Sin embargo, debe notar que estas evaluaciones tienen pesos distintos por lo que tendrá que calcular un promedio ponderado.

Escriba una función en el lenguaje Python 3.x que reciba una matriz `NOTAS` y un vector `pesos` y retorne el promedio final en un vector de tamaño `M`. Asuma que el vector de pesos tiene números entre 0 y 1 y suma 1 en su totalidad.

Nota: La matriz tendrá una dimensión de `MxN` y corresponderá a las notas en las `N` evaluaciones de los `M` estudiantes. El vector tendrá los pesos de la `N` evaluaciones.

La función deberá tener la siguiente cabecera:

```
promedio_final(NOTAS,pesos)
```

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/987

Ejemplo:

Entrada	Salida

`NOTAS = np.array([[15, 17],`
`[11, 13]])`
`pesos = np.array([0.5, 0.5])`
`print(promedio_final(NOTAS,pesos))`
	`[16.0, 12.0]`

Entrada	Salida

`NOTAS = np.array([[15, 17, 18],`
`[11, 13, 14]])`
`pesos = np.array([0.25, 0.25, 0.5])`
`print(promedio_final(NOTAS,pesos))`
	`[17.0, 13.0]`

06 - Vector Unitario (PCD27)

Escriba una función en el lenguaje de programación Python 3.x que reciba un vector `v` a forma de arreglo n-dimensional de numpy y retorne el vector unitario[4] `u` correspondiente, de acuerdo a la siguiente fórmula

$$u = \frac{v}{||v||_2}$$

Donde $||v||_2$ es la norma[5] euclidiana del vector `v` que se puede calcular de la siguiente manera

$$||v||_2 = \sqrt{\sum_i v_i^2}$$

[4] Un vector unitario mantiene la misma dirección que el vector original, pero su norma es 1
[5] Considere el uso de la función `np.linalg.norm()`

La función deberá tener la siguiente cabecera:

```
vector_unitario(v)
```

Ejemplo:

Entrada	Salida

`v = np.array([3,4])`
`u = vector_unitario(v)`
`print(u)`
	`[0.6, 0.8]`

`v = np.array([10,5,30,6,7])`
`u = vector_unitario(v)`
`print(u)`
	`[0.30015011, 0.15007506, 0.90045034, 0.18009007, 0.21010508]`

`v = np.array([1,0,0])`
`u = vector_unitario(v)`
`print(u)`
	`[1, 0, 0]`

Hint: la función deberá servir para calcular el vector unitario, siendo el vector v de diversos tamaños. Si usa alguna función de numpy no olvide importar numpy como `import numpy as np`

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/988

07 - Ángulo entre 2 vectores (PCD28)

Escriba una función en el lenguaje Python 3.x que calcule el ángulo entre dos vectores no nulos a y b, usando la fórmula a continuación

$$\angle(a,b) = arccos(\frac{a.b}{||a||_2||b||_2})$$

Donde $||a||_2$ es la norma[6] euclidiana del vector `a` que se puede calcular de la siguiente manera

[6] Considere el uso de la función `np.linalg.norm()`

$$||a||_2 = \sqrt{\sum_i a_i^2}$$

Recuerde que `a.b` denota el producto interno entre los vectores `a` y `b`, que se puede calcular con la función `np.dot` en numpy. El ángulo resultante será un número entre `[0, π]`

La función deberá tener la siguiente cabecera:

```
angulo(a, b)
```

Hint: La función `np.arccos()` permite calcular el arcocoseno. Si usa alguna función de numpy no olvide importar numpy como `import numpy as np`

El nombre del programa debe ser: `funciones.py`
Link: https://grader.labs.org.pe/web/project/993

Entrada	Salida

`a = np.array([1,0])`
`b = np.array([0,1])`
`print(angulo(a,b))`
	`1.5707963267948966`

`a = np.array([1,0,0])`
`b = np.array([0,1,0])`
`print(angulo(a,b))`
	`1.5707963267948966`

`a = np.array([1,1,5])`
`b = np.array([4,1,0])`
`print(angulo(a,b))`
	`1.3352440840134494`