Diplomado en Ciencia de Datos UNAM

Modulo 2: Matématica para ciencia de datos

Dr. Roberto Bárcenas C.

Alumno: Ibarra Ramírez Sergio

El participante identificará los principios fundamentales necesarios para la resolución de problemas del tema matemáticas para Ciencia de Datos.

Problema 1:

Sean los vectores u = (4,-2,-1) , v= (-3, 1, 2) y el escalar k=-2 Calcular:

a) u +v
b) v + ku
c) El producto interno (punto) uv
d) La norma de los vectores u y v

a) u +v

In [16]:
```python
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
import math

u = np.array([4, -2, -1])
v = np.array([-3, 1, 2])

w = u + v

print(f"El vector w = u+v es w:{w}")
```

```
El vector w = u+v es w:[ 1 -1  1]
```

b) v + ku

In [17]:
```python
u = np.array([4, -2, -1])
v = np.array([-3, 1, 2])

k = -2;
p = v + (k*u)

print(f"El vector p = v + ku es p:{p}")
```

```
El vector p = v + ku es p:[-11   5   4]
```

c) El producto interno (punto) uv

In [18]:
```python
m = np.dot(u,v)

print(f"El producto punto  m = u v es m:{m}")
```

```
El producto punto  m = u v es m:-16
```

d) La norma de los vectores u y v

In [19]:
```python
norm_u = np.linalg.norm(u)
norm_v = np.linalg.norm(v)

print("La normal del vector u es:", norm_u)
print("La normal del vector v es:", norm_v)
```

```
La normal del vector u es: 4.58257569495584
La normal del vector v es: 3.7416573867739413
```

Problema 2:

Sean las matrices $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}$ y $B = \begin{bmatrix} -2 & 9 \\ 3 & 5 \end{bmatrix}$ Calcular:

a) A + B
b) 2A - B
c) El producto AB
d) La matriz transpuesta A

a) A + B

In [20]:
```python
import numpy as np

# input two matrices
A = ([1, 2],[-1 ,0 ])
B = ([-2, 9],[3 ,5 ])


C = [[0,0],
        [0,0]]


## Suma de matrices
# iterate through rows
for i in range(len(A)):
# iterate through columns
    for j in range(len(A[0])):
        C[i][j] = A[i][j] + B[i][j]



# print resulted matrix
print(f"La matriz C = A + B es C:{C}")
```

```
La matriz C = A + B es C:[[-1, 11], [2, 5]]
```

Otra manera más sencilla de hacerlo en Python sería:

In [21]:
```python
import numpy as np

A = np.array([[1, 2], [-1, 0]])
B = np.array([[-2, 9], [3, 5]])

C = A + B

print("Matrix A:")
print(A)
print("Matrix B:")
print(B)
print("La matriz C = A + B es:")
print(C)
```

```
Matrix A:
[[ 1  2]
 [-1  0]]
Matrix B:
[[-2  9]
 [ 3  5]]
La matriz C = A + B es:
[[-1 11]
 [ 2  5]]
```

b) 2A - B

In [22]:
```python
k = -2
A2 = k * A
print(A2)



# This will return dot product
D = A2 -B


# print resulted matrix
print(f"La matriz D = A2 -B es D:{D}")

```

```
[[-2 -4]
 [ 2  0]]
La matriz D = A2 -B es D:[[  0 -13]
 [ -1  -5]]
```

c) El producto AB

In [23]:
```python
A = np.array([[1, 2], [-1, 0]])
B = np.array([[-2, 9], [3, 5]])

E = [[0 for x in range(len(A))] for y in range(len(B))]

# explicit for loops
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):

            # resulted matrix
            E[i][j] += A[i][k] * B[k][j]

print(f"El producto de las matrices {A} y {B} es la matriz E {E}")
```

```
El producto de las matrices [[ 1  2]
 [-1  0]] y [[-2  9]
 [ 3  5]] es la matriz E [[4, 19], [2, -9]]
```

Otra manera más sencilla de hacerlo en Python sería:

In [24]:
```python
# This will return dot product
E = np.dot(A,B)

# print resulted matrix
print(f"La matriz E = A B es E:{E}")
```

```
La matriz E = A B es E:[[ 4 19]
 [ 2 -9]]
```

d) La matriz transpuesta A

In [25]:
```python
A = np.array([[1, 2], [-1, 0]])

TraA = A.transpose()

# print resulted matrix
print(f"La matriz Transpuesta de A: {A} es T:{TraA}")
```

```
La matriz Transpuesta de A: [[ 1  2]
 [-1  0]] es T:[[ 1 -1]
 [ 2  0]]
```

Problema 3:

Sea la matriz

$$A = \begin{bmatrix} 3 & 1 & -1 \\ 6 & 1 & -2 \\ 4 & -3 & 2 \end{bmatrix}$$

Calcule su determinante

In [26]:
```python
A_1 = np.array([[3, 1, -1], [6, 1, -2], [4, -3, 2]])


det_A_1 = np.linalg.det(A_1)

print(f"El determinante de la matgriz A_1 :{det_A_1}")
```

```
El determinante de la matgriz A_1 :-10.000000000000002
```

Problema 4:

Determinar los siguientes limites
a) $\lim_{x->3} x^3 - 2x^2 + 8x - 32$
b) $\lim_{x->9} \frac{\sqrt{x}-3}{x-9}$

a) $\lim_{x->3} x^3 - 2x^2 + 8x - 32$

In [27]:
```python
from sympy import Symbol, sqrt, limit

x = Symbol('x')
f1 = x**3 -2*x**2 + 8*x -32
limitx_3 = limit(f1, x, 3)
print(f"El límite de f(x):{f1} en x=3 es: {limitx_3}")
```

```
El límite de f(x):x**3 - 2*x**2 + 8*x - 32 en x=3 es: 1
```

b) $\lim_{x->9} \frac{\sqrt{x}-3}{x-9}$

In [28]:
```python
from sympy import Symbol, sqrt, limit
```

```
x = Symbol('x')
f2 = (sqrt(x) - 3)/(x - 9)
limitx_9 = limit(f2, x, 9)

print(f"El límite de f(x):{f2} en x=9 es: {limitx_9}")
```

```
El límite de f(x):(sqrt(x) - 3)/(x - 9) en x=9 es: 1/6
```

Problema 5:

Determinar los siguientes derivadas

a) $(3x^3 - 2)^2(x^2 - 4x + 4)$

b) $\frac{sin(x^2 - x)}{(x-1)^2}$

a) $(3x^3 - 2)^2(x^2 - 4x + 4)$

In [29]:

```
from sympy import symbols, expand

x = symbols('x')
polinomial_a = (3*x**3 - 2)**2 * (x**2 - 4*x + 4)
polinomial_a = expand(polinomial_a)

print(polinomial_a)
```

```
9*x**8 - 36*x**7 + 36*x**6 - 12*x**5 + 48*x**4 - 48*x**3 + 4*x**2 - 16*x + 16
```

In [30]:

```
from sympy import  diff


# calculate the derivative of polinomial_a
derivative_a = diff(polinomial_a, x)

print(f"La derivada de f(x) = {polinomial_a} es: {derivative_a}")
```

```
La derivada de f(x) = 9*x**8 - 36*x**7 + 36*x**6 - 12*x**5 + 48*x**4 - 48*x**3 + 4*x**2 - 16*x + 16 es: 72*x**7 -
252*x**6 + 216*x**5 - 60*x**4 + 192*x**3 - 144*x**2 + 8*x - 16
```

b) $\frac{sin(x^2 - x)}{(x-1)^2}$

In [31]:

```
from sympy import symbols, diff, sin

# Define the symbol x
x = symbols('x')

# Define the function f(x)
f2 = sin(x**2 - x)/(x - 1)**2

# Compute the derivative of f(x) with respect to x
dfdx_2 = diff(f2, x)

# Print the result
print(f"La derivada de f(x) = {f2} es: {dfdx_2 }")
```

```
La derivada de f(x) = sin(x**2 - x)/(x - 1)**2 es: (2*x - 1)*cos(x**2 - x)/(x - 1)**2 - 2*sin(x**2 - x)/(x - 1)**3
```

Problema 6: Calcular la tabla de verdad para la siguiente proposición compuesta: (P->Q) ^ (P ^ ¬Q)

P	Q	(P->Q)	(P->Q) ^ (P ^ ¬Q)	(P ^ ¬Q)	¬Q
V	V	V	F	F	F
V	F	F	F	V	V
F	V	V	F	F	F
F	F	V	F	F	V

Problema 7: Presentar la matriz de adyacencia y de incidencia, respectivamente, para los siguientes grafos:

In [32]:

```
#import image module
from IPython.display import Image

# get the image of graph A
Image(url="Modulo2-Actividad3-GRAFO_matriz_incidencia.png", width=200, height=200)
```

Out[32]:

In [33]:

```
import networkx as nx

nodes = [1, 2, 3, 4, 5, 6]
edges = [[1, 3], [1, 4], [2, 3], [2, 4], [3, 5], [4, 6], [5, 6]]

G_1 = nx.DiGraph()
G_1.add_nodes_from(nodes)
G_1.add_edges_from(edges)


incidence_matrix_G1 = -nx.incidence_matrix(G_1, oriented=True)
print(f"La matriz de incidencia de G1 es: {incidence_matrix_G1.toarray()}")
```

```
La matriz de incidencia de G1 es: [[ 1.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  1.  0.  0.  0.]
 [-1.  0. -1.  0.  1.  0.  0.]
 [ 0. -1.  0. -1.  0.  1.  0.]
 [ 0.  0.  0.  0. -1.  0.  1.]
 [ 0.  0.  0.  0.  0. -1. -1.]]
```

In [34]:

```
adj_matrix_G1 = nx.adjacency_matrix(G_1)
print(f"La matriz de adyacencia de G_1 es:\n{adj_matrix_G1.toarray()}")
```

```
La matriz de adyacencia de G_1 es:
[[0 0 1 1 0 0]
 [0 0 1 1 0 0]
 [0 0 0 0 1 0]
 [0 0 0 0 0 1]
 [0 0 0 0 0 1]
 [0 0 0 0 0 0]]
```

In [35]:

```

# get the image of graph B
Image(url="Modulo2-Actividad3-GRAFO_B_matriz_incidencia.png", width=200, height=200)
```

Out[35]:

In [36]:

```
import networkx as nx

nodes2 = [1, 2, 3, 4, 5, 6]
edges2 = [[1, 2], [1, 3], [1, 4], [1, 5], [2, 4], [3, 4], [3, 5],  [4, 5],  [4, 6],  [5, 6]]

G_2 = nx.DiGraph()
G_2.add_nodes_from(nodes2)
G_2.add_edges_from(edges2)


incidence_matrix_G2 = -nx.incidence_matrix(G_2, oriented=True)
print(f"La matriz de incidencia de G2 es: {incidence_matrix_G2.toarray()}")
```

```
La matriz de incidencia de G2 es: [[ 1.  1.  1.  1.  0.  0.  0.  0.  0.  0.]
 [-1.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0.  0.  1.  1.  0.  0.  0.]
 [ 0.  0. -1.  0. -1. -1.  0.  1.  1.  0.]
 [ 0.  0.  0. -1.  0.  0. -1. -1.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1. -1.]]
```

In [37]:

```
adj_matrix_G2 = nx.adjacency_matrix(G_2)
print(f"La matriz de adyacencia de G_2 es:\n{adj_matrix_G2.toarray()}")
```

```
La matriz de adyacencia de G_2 es:
[[0 1 1 1 1 0]
 [0 0 0 1 0 0]
 [0 0 0 1 1 0]
 [0 0 0 0 1 1]
 [0 0 0 0 0 1]
 [0 0 0 0 0 0]]
```