Lautaro Colque

Parte I: Arreglos

1- Elaborar un algoritmo que lea un vector, busque el mayor elemento en valor absoluto y muestre el resultado.

Ejercicio1.py

```
1  from algo1 import *
2  import math
3  import random
4
5  vector_a=Array(10,0)
6
7  for i in range(0,10):
8      vector_a[i]=random.randint(-10,10)
9
10
11 def max_abs_value(arreglo):
12     num=abs(arreglo[0])
13     for i in arreglo:
14         if abs(i) >= num:
15             num = abs(i)
16
17     return num
18
19 print(vector_a)
20 print(max_abs_value(vector_a))
21
```

Consola

```
>>> %Run Ejercicio1.py
  [-8, 10, -10, -3, 4, 2, -4, -3, 7, -8]
  10

>>>
```

Lautaro Colque

2-Elaborar un algoritmo que lea dos vectores, verifique si tienen la misma dimensión y los sume en un nuevo vector. Calcule la norma cuadrática de este último vector. Muestre el vector resultado y su norma cuadrática.

Ejercicio2.py

```
from algo1 import *
import math
import random

vector_a=Array(10,0)
vector_b=Array(10,0)

for i in range(0,10):
    vector_a[i]=random.randint(-10,10)
    vector_b[i]=random.randint(-10,10)

def sum_arrays(Arreglo_1, Arreglo_2):

    largo = len(Arreglo_1)
    array_retorno=Array(largo,0)

    if len(Arreglo_1) == len(Arreglo_2):
        for i in range (0,largo):
            array_retorno[i]=Arreglo_1[i]+Arreglo_2[i]
    else:
        print ("No es posible realizar la suma")
        return

    return array_retorno
```

```
def calcular_norma_cuadratica(Arreglo):

    norma_cuadratica=0
    for i in range (0,len(Arreglo)):
        norma_cuadratica=norma_cuadratica+math.pow(Arreglo[i],2)

    return math.sqrt(norma_cuadratica)


print("Primer vector: ",vector_a)
print("Segundo vector: ",vector_b)
print("Suma vectores: ",sum_arrays(vector_a, vector_b))
print("Norma cuadratica: ",calcular_norma_cuadratica(sum_arrays(vector_a, vector_b)))

```

Consola

```
>>> %Run Ejercicio2.py

  Primer vector:  [4, -8, 2, 8, 0, 8, -9, 10, -4, -10]
  Segundo vector:  [2, 8, -10, 0, 8, -6, -6, 5, -6, -2]
  Suma vectores:  [6, 0, -8, 8, 8, 2, -15, 15, -10, -12]
  Norma cuadratica:  30.430248109405877

>>>
```

Lautaro Colque

3-Elaborar un algoritmo que lea una matriz y un vector, y que verifique si es posible la multiplicación. En el caso de ser posible realice la operación correspondiente, caso contrario, que muestre el mensaje “dimensiones incorrectas”.

Ejercicio1.py * | Ejercicio2.py | Ejercicio3.py

```
from algo1 import *
import math
import random

vector_a=Array(5,0)
matriz_a=Array(5,Array(5,0))

for i in range(0,5):
    vector_a[i]=random.randint(-10,10)
    for j in range(0,5):
        matriz_a[i][j]=random.randint(-10,10)


def producto_vector_matriz(Arreglo, Matriz):
    if len(Arreglo)!=len(Matriz):
        print("No es posible realizar el producto, dimensiones inconrrectas.")
        return

    columnas= len(Matriz[0])
    producto=Array(columnas,0)
    for i in range(0,columnas):
        suma=0
        aux=0
        for j in range (0, len(Arreglo)):
            aux=Arreglo[j]*Matriz[j][i]
            suma=suma+aux

        producto[i]=suma

    return producto

print("vector:")
print(vector_a)
print("matriz:")
for i in matriz_a:
    print (i)
print("producto:")
print(producto_vector_matriz(vector_a, matriz_a))
```

Consola

```
>>> %Run Ejercicio3.py
  vector:
  [3, 10, -3, 9, 9]
  matriz:
  [-7, 1, 3, 8, -2]
  [-7, 3, -9, -9, -4]
  [8, -3, -9, -7, 4]
  [-10, 4, -9, 6, -2]
  [-5, -3, 5, 1, 5]
  producto:
  [-250, 51, -90, 18, -31]

>>>
```

Lautaro Colque

4-Elaborar un algoritmo que lea dos matrices, calcule la diferencia de las mismas y almacene el resultado en una tercera matriz.

Ejercicio1.py * | Ejercicio2.py | Ejercicio3.py | Ejercicio4.py

```python
from algo1 import *
import math
import random

matriz_a=Array(5,Array(5,0))
matriz_b=Array(5,Array(5,0))

for i in range(0,5):
    for j in range(0,5):
        matriz_a[i][j]=random.randint(-10,10)
        matriz_b[i][j]=random.randint(-10,10)


def diferencia_matrices(Matriz_A, Matriz_B):
    if (len(Matriz_A)!=len(Matriz_B))  or (len(Matriz_A[0])!=len(Matriz_B[0])):
        print("No es posible hacer la diferencia de matrices, dimensiones incorrectas.")
        return

    filas = len(Matriz_A)
    columnas = len(Matriz_A[0])

    matriz_resultado=Array(filas,Array(columnas,0))

    for i in range (0,filas):
        for j in range (0,columnas):
            matriz_resultado[i][j]=Matriz_A[i][j]-Matriz_B[i][j]

    return matriz_resultado

print("matriz A:")
for i in matriz_a:
    print (i)

print("matriz B:")
for i in matriz_b:
    print (i)

print("diferencia: ")
for i in diferencia_matrices(matriz_a, matriz_b):
    print (i)
```

Consola

```
>>> %Run Ejercicio4.py
 matriz A:
 [-7, -6, -6, 1, 8]
 [8, 1, -1, 3, -4]
 [-6, 6, 3, -4, -9]
 [-1, -1, -5, 5, 2]
 [-3, -8, -2, 7, -9]
 matriz B:
 [10, 7, 7, 3, 2]
 [-4, -10, 10, -7, -5]
 [-10, 7, 2, -6, 1]
 [7, -3, -2, 1, -5]
 [5, -5, -2, 8, -2]
 diferencia:
 [-17, -13, -13, -2, 6]
 [12, 11, -11, 10, 1]
 [4, -1, 1, 2, -10]
 [-8, 2, -3, 4, 7]
 [-8, -3, 0, -1, -7]

>>>
```

Lautaro Colque

5-Elaborar un algoritmo que lea una matriz y determine si es triangular superior. En caso afirmativo el algoritmo debe calcular el determinante de dicha matriz.

Ejercicio5.py

```
from algo1 import *
import math
import random

matriz_a=Array(3,Array(3,0))

for i in range(0,3):
    for j in range(0,3):
        matriz_a[i][j]=random.randint(0,1)

def matriz_triangular_superior(Matriz):
    filas = len(Matriz)
    columnas = len(Matriz[0])

    if filas!=columnas:
        print("Ingrese una matriz cuadrada")
        return

    for i in range (1,filas):
        for j in range(i):
            if Matriz[i][j]!=0:
                return False
    return True

def det_MTS(Matriz):

    determinante=1
    for j in range (0,len(Matriz)):
        determinante=determinante*Matriz[j][j]

    return determinante


for i in matriz_a:
    print (i)

if matriz_triangular_superior(matriz_a):
    print("Es una Matriz Triangular Superior")
    print("Determinante: ",det_MTS(matriz_a))
else:print("No es Matriz Triangular Superior")
```

Consola

```
>>> %Run Ejercicio5.py
  [1, 1, 0]
  [0, 0, 1]
  [0, 0, 0]
  Es una Matriz Triangular Superior
  Determinante:  0
```

Lautaro Colque

PARTE II: TAD Conjuntos

check_duplicates(Array)

set.py

```python
from algo1 import *
import math

"""Funcion verificar duplicados"""
def check_duplicates(Array):

    largo=len(Array)

    for i in range(0,largo):
        suma=0
        for j in range(0,largo):
            if Array[i]==Array[j]:
                suma=suma+1
                if suma==2:
                    print("Existen elementos duplicados!")
                    return True

    return False
```

Create_Set(Array)

set.py

```python
"""Funcion crear set"""
def create_set(Arreglo):
    if check_duplicates(Arreglo)==False:
        print("El array ingresado ya es un TAD Set")
        return Arreglo

    largo=len(Arreglo)
    for i in range(0, largo):
        suma=0
        for j in range(0, largo):
            if Arreglo[i]==Arreglo[j]:
                suma=suma+1
        if suma>=2:
            Arreglo[i]=None

    nuevo_largo=0
    for k in Arreglo:
        if k!=None:
            nuevo_largo=nuevo_largo+1

    arreglo_retorno=Array(nuevo_largo,0)

    aux=0
    for l in range (0, largo):
        if Arreglo[l]!=None:
            arreglo_retorno[aux]=Arreglo[l]
            aux=aux+1

    return arreglo_retorno
```

Lautaro Colque

Union(Array S,Array T)

set.py

```
"""Funcion union arreglos"""
def union(Array_S, Array_T):
    if check_duplicates(Array_S)==True or check_duplicates(Array_T)==True: return


    largo_1=len(Array_S)
    largo_2=len(Array_T)
    largo_nuevo=largo_1+largo_2
    arreglo_retorno=Array(largo_nuevo,0)

    for i in range(0, largo_nuevo):

        if i<largo_1:
            arreglo_retorno[i]=Array_S[i]
        else:
            arreglo_retorno[i]=Array_T[i-largo_1]

    arreglo_retorno=create_set(arreglo_retorno)

    return arreglo_retorno
```

Intersection(Array S,Array T)

set.py

```
"""Funcion interseccion arreglos"""
def intersection(Array_S, Array_T):
    if check_duplicates(Array_S)==True or check_duplicates(Array_T)==True: return

    largo=len(Array_S)+len(Array_T)
    aux_index=0

    arreglo_retorno=Array(largo,0)

    for i in range (0, len(Array_S)):
        for j in range (0, len(Array_T)):
            if Array_S[i]==Array_T[j]:
                arreglo_retorno[aux_index]=Array_S[i]
                aux_index=aux_index+1

    arreglo_retorno=create_set(arreglo_retorno)

    return arreglo_retorno
```

Difference(Array S, Array T)

set.py

```
"""Funcion diferencia arreglos"""
def difference(Array_S, Array_T):
    if check_duplicates(Array_S)==True or check_duplicates(Array_T)==True: return

    largo=len(Array_S)+len(Array_T)
    aux_index=0

    arreglo_retorno=Array(largo,0)


    for i in range (0, len(Array_S)):
        contador=0
        for j in range (0, len(Array_T)):
            if Array_S[i]==Array_T[j]:
                contador=contador+1

        if contador==0:
            arreglo_retorno[aux_index]=Array_S[i]
            aux_index=aux_index+1

    arreglo_retorno=create_set(arreglo_retorno)

    return arreglo_retorno
```