SEGUNDO PARCIAL-PROGRAMACIÓN 8/12/2025

CLASE #1

Consulta

- Diferencias o caracteristicas de los bucles: 1.- While 2.- Do-while 3.- for

---

Tarea Hacer las figuras mostrada en clase

- 7 figuras en google colab

FECHA DE ENTREGA:LUNES 15

Técnicas:

- Programación estructurada
- Programación orientada a objetos

---

1. Inicialización
2. Condición
3. Incremento/Decremento

---

end=para imprimir los valores de forma horizontal

Ejemplo

```
for i in range(3):-> i=0,1,2                (3*4 interacciones)
  for j in range(4):-> j=0,1,2,3
```

i,j

```
for i in range(3):
    print(f"i:{i}")
    for j in range(4):
        print(f"\tj:{j}")
```

```
i:0
        j:0
        j:1
        j:2
        j:3
i:1
        j:0
        j:1
        j:2
```

```
        j:3
i:2
        j:0
        j:1
        j:2
        j:3
```

i,j,k

```
for i in range(3):
    print(f"i:{i}")
    for j in range(4):
        print(f"\tj:{j}")
        for k in range (2):
            print(f"\t\tk:{k}")
```

```
i:0
        j:0
                k:0
                k:1
        j:1
                k:0
                k:1
        j:2
                k:0
                k:1
        j:3
                k:0
                k:1
i:1
        j:0
                k:0
                k:1
        j:1
                k:0
                k:1
        j:2
                k:0
                k:1
        j:3
                k:0
                k:1
i:2
        j:0
                k:0
                k:1
        j:1
                k:0
                k:1
        j:2
                k:0
                k:1
        j:3
                k:0
                k:1
```

SEGUNDO EJERCICIO

```
for i in range(5):
    for j in range(5):
        print("*", end="")
    print()
```

```
*****
*****
*****
*****
*****
```

```
for i in range(6):
    for j in range(6):
        print("♥", end="")
    print()
```

```
♥♥♥♥♥♥
♥♥♥♥♥♥
♥♥♥♥♥♥
♥♥♥♥♥♥
♥♥♥♥♥♥
♥♥♥♥♥♥
```

```
for i in range(6):
    for j in range(i):
        print("♥☺♦♣♠•◘○", end="")
    print()
```

```

♥☺♦♣♠•◘○
♥☺♦♣♠•◘○♥☺♦♣♠•◘○
♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○
♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○
♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○♥☺♦♣♠•◘○
```

TERCER EJERCICIO

```
n=int(input("ingrese n:"))
for i in range(n):
    for j in range(i+1):
        print("♥",end="\t")
    print()
```

```
ingrese n:8
♥
♥	♥
♥	♥	♥
♥	♥	♥	♥
♥	♥	♥	♥	♥
♥	♥	♥	♥	♥	♥
♥	♥	♥	♥	♥	♥	♥
```

♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥

> CONSULTA

En Python, **for** itera sobre secuencias (listas, rangos), ideal para un número conocido de veces; **while** repite mientras una condición sea verdadera, para un número desconocido de iteraciones; y no hay un **do-while** nativo, pero se simula con while True y break, asegurando que el bloque se ejecute al menos una vez antes de la comprobación, a diferencia del while normal que puede no correr nunca. **Bucle for** Propósito: Recorrer elementos de una secuencia (lista, tupla, cadena, range). Condición: Implícita; termina cuando no quedan elementos en la secuencia. Ideal para: Iteraciones definidas (ej. "haz esto 10 veces", "recorre cada nombre en esta lista").

```
# Ejemplo for
for i in range(3):
    print(f"Iteración {i}")
```

Bucle while Propósito: Repetir un bloque de código mientras una condición sea True. Condición: Se evalúa antes de cada iteración. Ideal para: Iteraciones indefinidas (ej. "espera una entrada válida del usuario", "lee archivos hasta el final"). Riesgo: Puede ser un bucle infinito si la condición nunca se vuelve False.

```
# Ejemplo while
contador = 0
while contador < 3:
    print(f"Contador es {contador}")
    contador += 1
```

Bucle do-while Propósito: Ejecutar el cuerpo del bucle al menos una vez, y luego verificar la condición. Implementación en Python: No existe de forma nativa, se emula con while True y una sentencia break. Ideal para: Tareas donde se necesita una primera ejecución incondicional (ej. leer una entrada y luego validarla).

```
# Ejemplo do-while simulado
while True:
    entrada = input("Escribe 'salir': ")
    if entrada == 'salir':
        break # Sale del bucle si la condición se cumple
    print("No escribiste 'salir'.")
```

for vs. while: for es para secuencias conocidas; while es para condiciones que pueden tardar en cambiar. while vs. do-while: while puede no ejecutarse nunca; do-while (simulado) se ejecuta al menos una vez.

10/12/2025 CLASE #2

Teoria: Codificacion mediante bloques

11/12/2025 CLASE #3

```
#PYTHON                                 #C++
for i in range(5)                       for(i=0,i<=5,i++)
    print(i)                            {

                                        }
```

DEBER #1

FIGURA 1

```
n=int(input("Ingrese el numero: "))
for i in range(n):
    for j in range(n):
        if j >= n - i - 1:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

```
Ingrese el numero: 7
            *
          * *
        * * *
      * * * *
    * * * * *
  * * * * * *
* * * * * * *
```

FIGURA 2

```
n = int(input("Ingrese el numero: "))
for i in range(n):
    for j in range(n):
        if i <= j:
            print("*", end=" ")
        else:
            print(" ", end=" ")
```

```
    print()
```

```
Ingrese el numero: 8
* * * * * * * *
  * * * * * * *
    * * * * * *
      * * * * *
        * * * *
          * * *
            * *
              *
```

FIGURA 3

```
n = int(input("Ingrese el numero: "))
for i in range(n):
    for j in range(n):
        if i + j < n:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

```
Ingrese el numero: 9
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

FIGURA 4

```
n = int(input("Ingrese el numero: "))
for i in range(n):
    print(" " * (n - i - 1) + "* " * (i + 1))
```

```
Ingrese el numero: 8
       *
      * *
     * * *
    * * * *
   * * * * *
  * * * * * *
 * * * * * * *
* * * * * * * *
```

FIGURA 5

```
n = int(input("Ingrese el numero: "))
for i in range(n):
    print("*" * (n - i) + "  " * i + "*" * (n - i))
for i in range(n - 2, -1, -1):
    print("*" * (n - i) + "  " * i + "*" * (n - i))
```

```
Ingrese el numero: 6
**********
*****  *****
****    ****
***      ***
**        **
*          *
**        **
***      ***
****    ****
*****  *****
**********
```

FIGURA 6

```
#solo numeros impares
n = int(input("Ingrese un número impar: "))
centro = n // 2
for i in range(centro + 1):
    print(" " * (centro - i) + "* " * (i + 1))
for i in range(centro - 1, -1, -1):
    print(" " * (centro - i) + "* " * (i + 1))
```

```
Ingrese un número impar: 9
    *
   * *
  * * *
 * * * *
* * * * *
 * * * *
  * * *
   * *
    *
```

FIGURA 7

```
n = int(input("Ingrese el numero: "))
for i in range(n):
    for j in range(n):
        if i == 0 or i == n-1 or j == 0 or j == n-1:
            print("*", end=" ")
        else:
            print(" ", end=" ")
```

```
    print()
```

```
Ingrese el numero: 10
* * * * * * * * * *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
* * * * * * * * * *
```

FIGURA EXTRA

```
n = int(input("Ingrese el numero: "))
luces = ["o", "O", "@"]
print(" " * (n - 1) + "⭐")
for i in range(n):
    print(" " * (n - i - 1), end="")
    for j in range(2 * i + 1):
        if (i + j) % 5 == 0:
            print(luces[(i + j) % len(luces)], end="")
        elif (i + j) % 7 == 0:
            print("~", end="")
        else:
            print("*", end="")
    print()
for i in range(3):
    print(" " * (n - 2) + "|||")
```

```
Ingrese el numero: 12
           ⭐
          o
         ***
        ***@*
       **@*~**
      *@*~**O**
     @*~**O***~o
    *~**O***~o***
   ~**O***~o****@~
  **O***~o****@~***
 *O***~o****@~***O**
O***~o****@~***O**~*o
***~o****@~***O**~*o***
          |||
          |||
          |||
```

CLASE #4 15/12/2025 Trabajamos en tinkercad-circuitos

CLASE #5 17/12/2025

Programacion modular

Estructura de datos: Una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella. Los sistemas de datos son muy importantes en los sistemas de computadoras. CONJUNTO DE DOS QUE SE NECESITA CAPTARLOS,ALMACENARLOS Y PROCESARLOS.

Clasificación:

- Datos simples: entero, real,logicos
- Datos estructurados:esaticos y dinamicos

Archivos y ficheros-> guardar

Datos: Conjunto de videos,imagenes,numeros,etc.

ARREGLOS (ARRAY)

*Tambien llamados vectores(una sola dimencion) vector fila o columna. **pregunta de examen**

*Areglos multidimencionales (dos o mas dimenciones) -> Matrices.

*3 Dimensiones -> tensores(conjunto de matrices)

- Son homogeneos, es decir, del mismo tipo de dato.
- Ordenados

NUMPY *LIBRRERIA DE PYTHON*

VECTOR FILA -> HORIZONTAL

VECTOR COLUMNA -> VERTICAL

LIBRERIAS

MATHPLOTLIB

PANDAS

ETRUCTURAS DE DATOS DE PYTHON PREGUNTA DE EXAMEN

- LISTAS []
- TUPLAS ()
- DICCIONARIOS {}

```
Lista=[3,1,4,2,5]
print(Lista[-1])
```

```
Lista=[3,1,4,2,5]
print(Lista[4])
```

```
Lista=[3,1,4,2,5]
suma=Lista[0]+Lista[1]+Lista[2]+Lista[3]+Lista[4]
print(f" la suma es {suma}")
```

CLASE #6 *18/12/2025*

```
v=[7,3,5,8,9,1,4]
suma=v[0]+v[1]+v[2]+v[3]+v[4]+v[5]+v[6]
print(f"la suma es {suma}")
```

```
v=[7,3,5,8,9,1,4]
suma=0
n=len(v)
print(n)
```

```
v=[7,3,5,8,9,1,4]
suma=0
n=len(v)
for i in range(n):
    suma=suma+v[i]
print(f"la suma es {suma}")
```

```
v=[7,3,5,8,9,1,4]
suma=sum(v)
n=len(v)
print(f"la suma es {suma}")
```

```
v=[7,3,5,8,9,1,4]
suma=min(v)
n=len(v)
print(f"la suma es {suma}")
```

append

```
v=[]
n=int(input("ingrese el numero:"))
for i in range(n):
    num=int(input(f"ingrese elemento[{i}]:"))
    v.append(num)
print(v)
```

Haz doble clic (o ingresa) para editar

```
v=[]
n=int(input("ingrese el numero:"))
for i in range(n):
    num=int(input(f"ingrese elemento[{i}]:"))
```

```
    v.append(num)
print(v)
mayor=v[0]
for i in range(1,n):
    if v[i]>=mayor:
        mayor=v[i]
print(f"el mayor {mayor}")
```

Haz doble clic (o ingresa) para editar

```
v=[]
n=int(input("ingrese el numero:"))
for i in range(n):
    num=int(input(f"ingrese elemento[{i}]:"))
    v.append(num)
print(v)
menor=v[0]
for i in range(1,n):
    if v[i]>=menor:
        nemor=v[i]
print(f"el menor {menor}")
mayor=v[0]
for i in range(1,n):
    if v[i]>=mayor:
        mayor=v[i]
print(f"el mayor {mayor}")
```

CLASE#7 5/01/2026

INDEXACIÓN

La indexación en programación es el proceso de organizar y estructurar datos (como en bases de datos, archivos o listas) usando "índices" (marcadores, palabras clave, o posiciones numéricas) para permitir una búsqueda y recuperación de información mucho más rápida y eficiente

PARTES DE UN VECTOR PREGUNTA DE EXAMEN

1.- Nombre

2.- Tamaño

3.- Tipo de dato

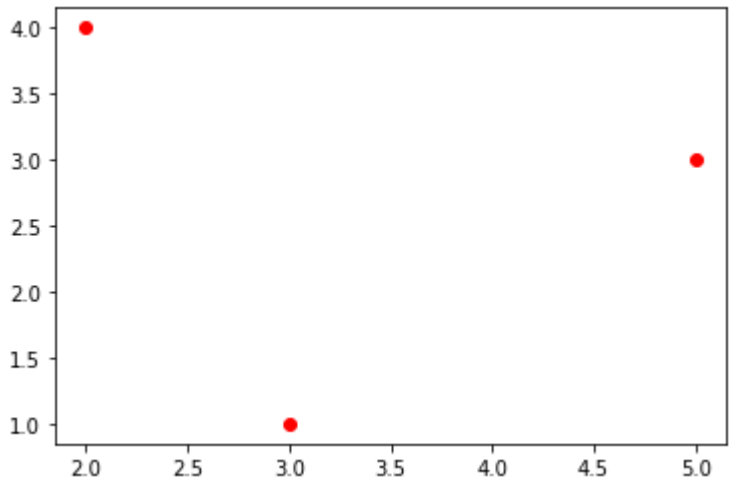
```
import random as rd
vec = []
n = int(input("Ingrese tamaño: "))
for i in range(n):
    num = rd.randint(0, 10)  # números enteros entre 0 y 10
    vec.append(num)
```

```
print(vec)
```

```
import random as rd
vec = []
n = int(input("Ingrese el tamaño: "))
for i in range(n):
    num = round(rd.random(), 3)
    vec.append(num)
print(vec)
```

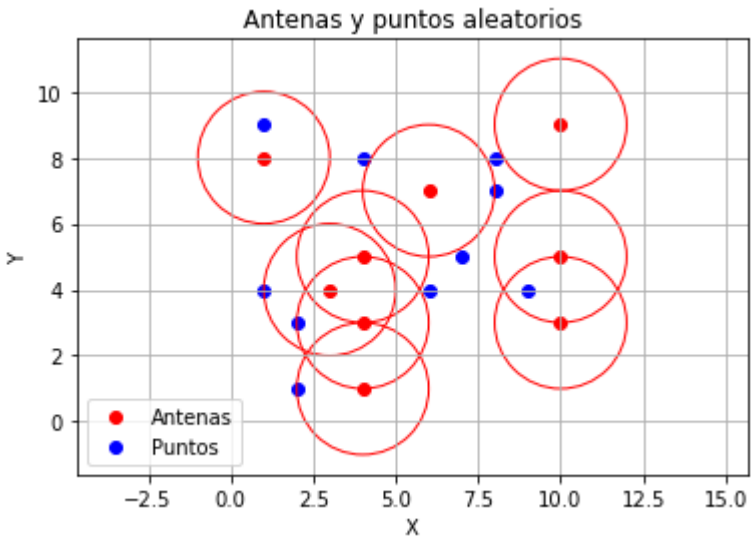
```
import matplotlib.pyplot as plt
x=[3,2,5]
y=[1,4,3]
plt.plot(x,y,"ro")
plt.show()
```

Grafica




```
import matplotlib.pyplot as plt
import random as rd
import math

# Cantidades
n_antenas = int(input("Ingrese número de antenas: "))
n_puntos = int(input("Ingrese número de puntos aleatorios: "))

# Vectores
antenas_x = []
antenas_y = []

puntos_x = []
puntos_y = []

# Generar antenas
for i in range(n_antenas):
    antenas_x.append(rd.randint(1, 10))
```

```
    antenas_y.append(rd.randint(1, 10))

# Generar puntos aleatorios
for i in range(n_puntos):
    puntos_x.append(rd.randint(1, 10))
    puntos_y.append(rd.randint(1, 10))

# Graficar antenas y puntos
plt.scatter(antenas_x, antenas_y, color='red', label='Antenas')
plt.scatter(puntos_x, puntos_y, color='blue', label='Puntos')

# Radio de cobertura
radio = 2

# Dibujar circunferencias en cada antena
for x, y in zip(antenas_x, antenas_y):
    circulo = plt.Circle((x, y), radio, color='red', fill=False)
    plt.gca().add_patch(circulo)

# Configuración del gráfico
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Antenas y puntos aleatorios")
plt.legend()
plt.axis('equal')  # Para que las circunferencias no se deformen
plt.grid()

plt.show()
```

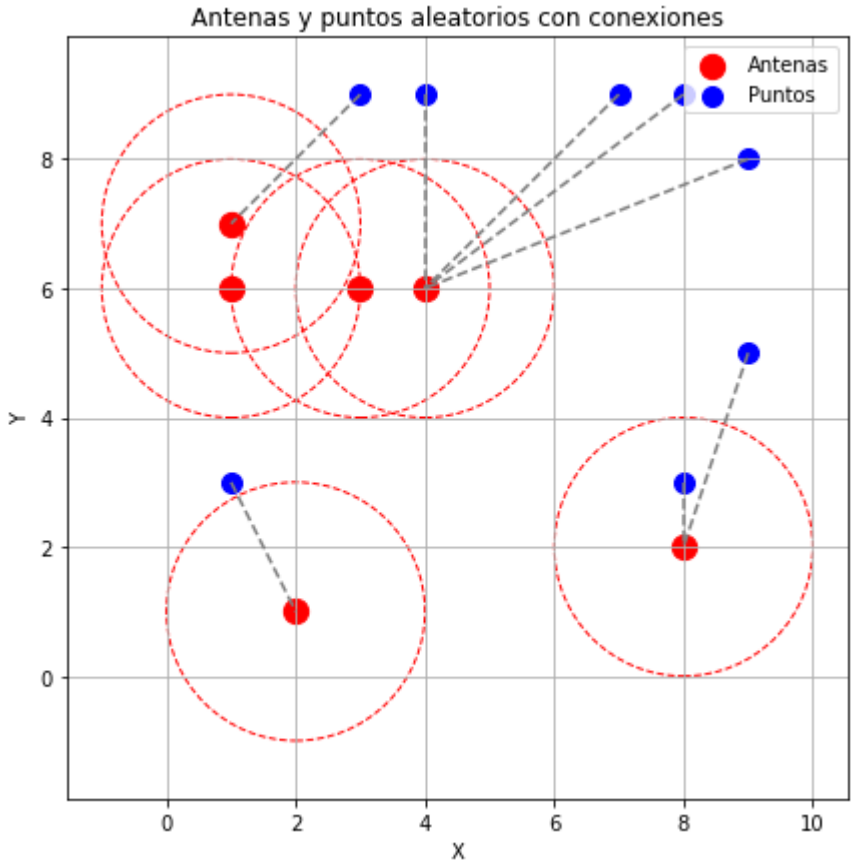


GRAFICA

```
import matplotlib.pyplot as plt
import random as rd
import math

# --- Cantidades ---
n_antenas = int(input("Ingrese número de antenas: "))
n_puntos = int(input("Ingrese número de puntos aleatorios: "))
```

```
# --- Vectores ---
antenas_x = []
antenas_y = []
puntos_x = []
puntos_y = []

# --- Generar antenas ---
for i in range(n_antenas):
    antenas_x.append(rd.randint(1, 10))
    antenas_y.append(rd.randint(1, 10))

# --- Generar puntos aleatorios ---
for i in range(n_puntos):
    puntos_x.append(rd.randint(1, 10))
    puntos_y.append(rd.randint(1, 10))

# --- Graficar antenas y puntos ---
plt.figure(figsize=(7,7))
plt.scatter(antenas_x, antenas_y, color='red', s=150, label='Antenas')
plt.scatter(puntos_x, puntos_y, color='blue', s=100, label='Puntos')

# --- Radio de cobertura ---
radio = 2
for x, y in zip(antenas_x, antenas_y):
    circulo = plt.Circle((x, y), radio, color='red', fill=False, linestyle='
    plt.gca().add_patch(circulo)

# --- Conectar cada punto con la antena más cercana ---
for px, py in zip(puntos_x, puntos_y):
    # Inicializar distancia mínima muy grande
    distancia_min = float('inf')
    antena_cercana = (0,0)

    # Buscar la antena más cercana
    for ax, ay in zip(antenas_x, antenas_y):
        distancia = math.sqrt((px - ax)**2 + (py - ay)**2)
        if distancia < distancia_min:
            distancia_min = distancia
            antena_cercana = (ax, ay)

    # Dibujar línea hacia la antena más cercana
    plt.plot([px, antena_cercana[0]], [py, antena_cercana[1]], color='gray',

# --- Configuración del gráfico ---
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Antenas y puntos aleatorios con conexiones")
plt.legend()
plt.axis('equal')
plt.grid(True)

plt.show()
```

GRAFICA

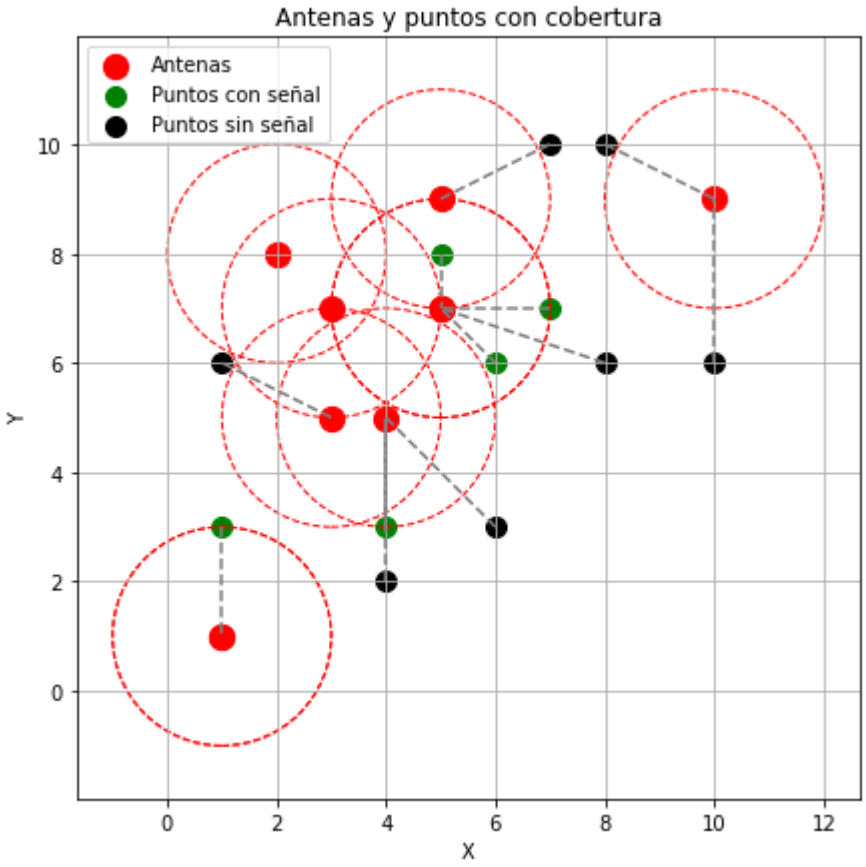



```
import matplotlib.pyplot as plt
import random as rd
import math

# --- Cantidades ---
n_antenas = int(input("Ingrese número de antenas: "))
n_puntos = int(input("Ingrese número de puntos aleatorios: "))

# --- Radio de cobertura ---
radio = 2

# --- Vectores ---
antenas_x = []
antenas_y = []
puntos_x = []
puntos_y = []

# --- Generar antenas ---
for i in range(n_antenas):
    antenas_x.append(rd.randint(1, 10))
    antenas_y.append(rd.randint(1, 10))

# --- Generar puntos aleatorios ---
for i in range(n_puntos):
    puntos_x.append(rd.randint(1, 10))
    puntos_y.append(rd.randint(1, 10))
```

```
# --- Listas para puntos con y sin señal ---
puntos_con_senal = []
puntos_sin_senal = []

# --- Comprobar cobertura ---
for px, py in zip(puntos_x, puntos_y):
    tiene_senal = False
    for ax, ay in zip(antenas_x, antenas_y):
        distancia = math.sqrt((px - ax)**2 + (py - ay)**2)
        if distancia <= radio:
            tiene_senal = True
            break
    if tiene_senal:
        puntos_con_senal.append((px, py))
    else:
        puntos_sin_senal.append((px, py))

# --- Graficar antenas ---
plt.figure(figsize=(7,7))
plt.scatter(antenas_x, antenas_y, color='red', s=150, label='Antenas')

# --- Graficar puntos con señal (verde) y sin señal (negro) ---
if puntos_con_senal:
    x_senal, y_senal = zip(*puntos_con_senal)
    plt.scatter(x_senal, y_senal, color='green', s=100, label='Puntos con se

if puntos_sin_senal:
    x_no, y_no = zip(*puntos_sin_senal)
    plt.scatter(x_no, y_no, color='black', s=100, label='Puntos sin señal')

# --- Dibujar circunferencia de cobertura de antenas ---
for x, y in zip(antenas_x, antenas_y):
    circulo = plt.Circle((x, y), radio, color='red', fill=False, linestyle='
    plt.gca().add_patch(circulo)

# --- Conectar puntos con su antena más cercana ---
for px, py in zip(puntos_x, puntos_y):
    distancia_min = float('inf')
    antena_cercana = (0,0)
    for ax, ay in zip(antenas_x, antenas_y):
        distancia = math.sqrt((px - ax)**2 + (py - ay)**2)
        if distancia < distancia_min:
            distancia_min = distancia
            antena_cercana = (ax, ay)
    plt.plot([px, antena_cercana[0]], [py, antena_cercana[1]], color='gray',

# --- Configuración del gráfico ---
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Antenas y puntos con cobertura")
plt.legend()
plt.axis('equal')
plt.grid(True)

plt.show()
```

```
# --- Mostrar resultados ---
print(f"Puntos con señal: {len(puntos_con_senal)}")
print(f"Puntos sin señal: {len(puntos_sin_senal)}")
```

GRAFICA



CLASE#8 7/01/2026

```
import random as rd
vec = []
n = int(input("Ingrese el tamaño: "))
for i in range(n):
    num = round(rd.random(), 3)
    vec.append(num)
print("Horizontal:")
print(vec)
print("\nVertical:")
for elemento in vec:
    print(elemento)
```

```
Ingrese el tamaño: 5
Horizontal:
[0.205, 0.618, 0.061, 0.893, 0.945]

Vertical:
0.205
```

```
0.618
0.061
0.893
0.945
```

RESULTADO

```
Ingrese el tamaño: 4
Horizontal:
[0.125, 0.847, 0.392, 0.671]

Vertical:
0.125
0.847
0.392
0.671
```

Producto cruz

```
n = int(input("Ingrese el tamaño de los vectores: "))
if n != 3:
    print("El producto cruz solo se puede calcular para vectores de tamaño 3
else:
    A = []
    B = []
    print("\nIngrese los elementos del vector A:")
    for i in range(n):
        A.append(float(input(f"A[{i}]: ")))
    print("\nIngrese los elementos del vector B:")
    for i in range(n):
        B.append(float(input(f"B[{i}]: ")))
    producto_cruz = [
        A[1]*B[2] - A[2]*B[1],
        A[2]*B[0] - A[0]*B[2],
        A[0]*B[1] - A[1]*B[0]
    ]
    print("\nProducto cruz (A × B):")
    print(producto_cruz)
```

```
Ingrese el tamaño de los vectores: 8
El producto cruz solo se puede calcular para vectores de tamaño 3.
```

RESULTADO

```
Ingrese los elementos del vector A:
A[0]: 85
A[1]: 45
A[2]: 231
```

```
Ingrese los elementos del vector B:
B[0]: 44
B[1]: 98
B[2]: 24

Producto cruz (A × B):
[-21558.0, 8124.0, 6350.0]
```

```
import random as rd
vec = []
n = int(input("Ingrese tamaño del vector: "))
for i in range(n):
    vec.append(rd.randint(1, 10))
buscar = int(input("Ingrese el número a buscar: "))
contador = 0
for elemento in vec:
    if elemento == buscar:
        contador += 1
print("\nVector:", vec)
print(f"El número {buscar} se repite {contador} veces en el vector")
```

```
Ingrese tamaño del vector: 5
Ingrese el número a buscar: 9

Vector: [8, 9, 8, 10, 8]
El número 9 se repite 1 veces en el vector
```

```
import random as rd
n=int(input('Ingrese tamanio:'))
a=[]
#Insetar datos aleatorios en el vector
for i in range(n):
    num=rd.randint(0,10)
    a.append(num)
print(f'Vector Original:{a}')

#Metodo de Ordenamiento Burbuja
for i in range(1,n):
    for j in range(n-i):
        if a[j]>a[j+1]:
            aux=a[j]
            a[j]=a[j+1]
            a[j+1]=aux
print(f'Vector Ordenado:{a}')
```

```
Ingrese tamanio:8
Vector Original:[2, 5, 2, 1, 7, 8, 10, 3]
Vector Ordenado:[1, 2, 2, 3, 5, 7, 8, 10]
```

Ejercicio

```
import random as rd
```

```
# Tamaño fijo del vector
n = 5
a = []

# Insertar datos aleatorios en el vector
for i in range(n):
    num = rd.randint(0, 10)
    a.append(num)

print(f'Vector Original: {a}')

# Método de Ordenamiento Burbuja
for i in range(1, n):
    for j in range(n - i):
        if a[j] > a[j + 1]:
            aux = a[j]
            a[j] = a[j + 1]
            a[j + 1] = aux

print(f'Vector Ordenado: {a}')
```

CLASE#9 08/1/2026

Nombres de personas ordenadas alfabeticamente

```
n = int(input("Ingrese la cantidad de nombres: "))
nombres = []
for i in range(n):
    nombre = input(f"Ingrese el nombre {i + 1}: ")
    nombres.append(nombre)
print(f'\nNombres Originales: {nombres}')
# Método de Ordenamiento Burbuja (alfabético)
for i in range(1, n):
    for j in range(n - i):
        if nombres[j].lower() > nombres[j + 1].lower():
            aux = nombres[j]
            nombres[j] = nombres[j + 1]
            nombres[j + 1] = aux
print(f'Nombres Ordenados Alfabéticamente: {nombres}')
```

ORDENAMIENTO RÁPIDO (QUICKSORT)

PARA NUMEROS

```
def quicksort(lista):
    if len(lista) <= 1:
        return lista
    pivote = lista[0]
```

```
        menores = []
        mayores = []
        for i in range(1, len(lista)):
            if lista[i] <= pivote:
                menores.append(lista[i])
            else:
                mayores.append(lista[i])
        return quicksort(menores) + [pivote] + quicksort(mayores)
    n = int(input("Ingrese la cantidad de números: "))
    lista = []
    for i in range(n):
        num = int(input(f"Ingrese el número {i + 1}: "))
        lista.append(num)
    print("\nLista original:", lista)
    lista_ordenada = quicksort(lista)
    print("Lista ordenada:", lista_ordenada)
```

PARA NOMBRES

```
    def quicksort_nombres(lista):
        if len(lista) <= 1:
            return lista
        pivote = lista[0]
        menores = []
        mayores = []
        for i in range(1, len(lista)):
            if lista[i].lower() <= pivote.lower():
                menores.append(lista[i])
            else:
                mayores.append(lista[i])
        return quicksort_nombres(menores) + [pivote] + quicksort_nombres(mayores
    n = int(input("Ingrese la cantidad de nombres: "))
    nombres = []
    for i in range(n):
        nombre = input(f"Ingrese el nombre {i + 1}: ")
        nombres.append(nombre)
    print("\nNombres originales:", nombres)
    nombres_ordenados = quicksort_nombres(nombres)
    print("Nombres ordenados alfabéticamente:", nombres_ordenados)
```

12/01/2026 CLASE #10

TIPOS DE ARREGLOS pregunta de examen

bidimencionales -> tablas o matrices

multidimencionales -> Mas de tres dimenciones

```
M=[[5,1,3],[7,2,1]]
N=[[3,4,1,2,7],[5,4,8,2,0],[0,0,1,3,5]]
```

```
M=[[5,1,3],[7,2,1]]
print(M)
print("forma matriz")
for i in M:
    print(i)
```

14/01/2026 CLASE #11

PREGUNTA DE EXAMEN: COMO PONER UNA MATRIZ EN FORMA DE PAYTHON

A=[[1,2,1],[5,8,9]]

```
Comienza a programar o generar con IA.
```

```
import random as rd
A=[]
m=int(input("ingrese el numero de filas"))
n=int(input("ingrese el numero de columnas"))
for i in range(m):
    filas=[]
    for j in range(n):
        num=rd.randint(a=0,b=28)
        filas.append(num)
    A.append(filas)
print(A)
print("modo matriz")
for i in A:
    print(i)
```

```
ingrese el numero de filas2
ingrese el numero de columnas3
[[7, 10, 6], [28, 21, 18]]
modo matriz
[7, 10, 6]
[28, 21, 18]
```

```
import random as rd
A=[]
m=int(input("ingrese el numero de filas"))
n=int(input("ingrese el numero de columnas"))
for i in range(m):
    filas=[]
    for j in range(n):
        num=rd.randint(a=0,b=28)
        filas.append(num)
    A.append(filas)
    print(A)
```

```
ingrese el numero de filas8
ingrese el numero de columnas6
[[9, 22, 14, 7, 21, 7]]
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19]]
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5]]
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5], [26
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5], [26
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5], [26
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5], [26
[[9, 22, 14, 7, 21, 7], [20, 11, 16, 21, 18, 19], [19, 26, 22, 0, 24, 5], [26
```

```
import random as rd
A=[]
m=int(input("ingrese el numero de filas"))
n=int(input("ingrese el numero de columnas"))
for i in range(m):
    filas=[]
    for j in range(n):
        num=rd.randint(a=0,b=28)
        filas.append(num)
    A.append(filas)
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
```

```
ingrese el numero de filas3
ingrese el numero de columnas2
14      14
19      28
18      24
```

```
import random as rd
A = []
m = int(input("Ingrese el numero de filas: "))
n = int(input("Ingrese el numero de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        num = rd.randint(0, 28)
        filas.append(num)
    A.append(filas)
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
mayor = A[0][0]
pos_i = 0
pos_j = 0
for i in range(m):
    for j in range(n):
        if A[i][j] > mayor:
            mayor = A[i][j]
            pos_i = i
```

```
            pos_j = j
print("\nEl mayor numero es:", mayor)
print(f"Se encuentra en la posicion [{pos_i}][{pos_j}]")
```

```
Ingrese el numero de filas: 5
Ingrese el numero de columnas: 6
2       21      5       13      12      6
26      24      22      28      27      17
5       9       21      6       16      19
15      16      19      4       22      23
2       9       19      6       22      15

El mayor numero es: 28
Se encuentra en la posicion [1][3]
```

```
import random as rd
A = []
m = int(input("Ingrese el numero de filas: "))
n = int(input("Ingrese el numero de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        num = rd.randint(0, 28)
        filas.append(num)
    A.append(filas)
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
mayor = A[0][0]
menor = A[0][0]
mayor_i = mayor_j = 0
menor_i = menor_j = 0

for i in range(m):
    for j in range(n):
        if A[i][j] > mayor:
            mayor = A[i][j]
            mayor_i = i
            mayor_j = j
        if A[i][j] < menor:
            menor = A[i][j]
            menor_i = i
            menor_j = j
print("\nEl mayor numero es:", mayor)
print(f"Se encuentra en la posicion [{mayor_i}][{mayor_j}]")
print("\nEl menor numero es:", menor)
print(f"Se encuentra en la posicion [{menor_i}][{menor_j}]")
```

```
Ingrese el numero de filas: 3
Ingrese el numero de columnas: 4
9       18      19      3
16      17      17      19
17      24      24      26
```

```
El mayor numero es: 26
Se encuentra en la posicion [2][3]

El menor numero es: 3
Se encuentra en la posicion [0][3]
```

```
import random as rd
A = []
m = int(input("Ingrese el numero de filas: "))
n = int(input("Ingrese el numero de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        num = rd.randint(0, 28)
        filas.append(num)
    A.append(filas)
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
mayor = A[0][0]
menor = A[0][0]
suma = 0
mayor_i = mayor_j = 0
menor_i = menor_j = 0
for i in range(m):
    for j in range(n):
        suma += A[i][j]
        if A[i][j] > mayor:
            mayor = A[i][j]
            mayor_i = i
            mayor_j = j
        if A[i][j] < menor:
            menor = A[i][j]
            menor_i = i
            menor_j = j
print("\nEl mayor numero es:", mayor)
print(f"Se encuentra en la posicion [{mayor_i}][{mayor_j}]")
print("\nEl menor numero es:", menor)
print(f"Se encuentra en la posicion [{menor_i}][{menor_j}]")
print("\nLa suma de todos los elementos de la matriz es:", suma)
```

```
Ingrese el numero de filas: 6
Ingrese el numero de columnas: 3
20      8       28
6       15      19
15      19      13
19      22      19
4       22      10
2       5       17

El mayor numero es: 28
Se encuentra en la posicion [0][2]
```

```
El menor numero es: 2
Se encuentra en la posicion [5][0]

La suma de todos los elementos de la matriz es: 263
```

```
import random as rd
A = []
m = int(input("Ingrese el numero de filas: "))
n = int(input("Ingrese el numero de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        num = rd.randint(0, 28)
        filas.append(num)
    A.append(filas)
print("Matriz original:")
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
pares = []
for i in range(m):
    fila_pares = []
    for j in range(n):
        if A[i][j] % 2 == 0:
            fila_pares.append(A[i][j])
    pares.append(fila_pares)
print("\nMatriz con solo numeros pares:")
for i in range(m):
    for j in range(len(pares[i])):
        print(pares[i][j], end="\t")
    print()
```

```
Ingrese el numero de filas: 4
Ingrese el numero de columnas: 3
Matriz original:
21      1       9
9       27      17
1       13      6
27      10      1

Matriz con solo numeros pares:


6
10
```

```
import random as rd
A = []
m = int(input("Ingrese el numero de filas: "))
n = int(input("Ingrese el numero de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        filas.append(rd.randint(0, 28))
```

```
    A.append(filas)
print("Matriz original:")
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
pares = []
print("\nNumeros pares y sus posiciones:")
for i in range(m):
    for j in range(n):
        if A[i][j] % 2 == 0:
            pares.append(A[i][j])
            print(f"Numero {A[i][j]} en la posicion [{i}][{j}]")
print("\nVector con todos los numeros pares:")
print(pares)
```

```
Ingrese el numero de filas: 2
Ingrese el numero de columnas: 3
Matriz original:
11      0       13
15      25      9

Numeros pares y sus posiciones:
Numero 0 en la posicion [0][1]

Vector con todos los numeros pares:
[0]
```

```
import random as rd
A = []
m = int(input("Ingrese el número de filas: "))
n = int(input("Ingrese el número de columnas: "))
for i in range(m):
    fila = []
    for j in range(n):
        num = rd.randint(0, 28)
        fila.append(num)
    A.append(fila)
print("Matriz original:")
for fila in A:
    print(fila)
B = []
contador_unos = 0
contador_ceros = 0
for fila in A:
    fila_binaria = []
    for elemento in fila:
        if elemento == 0:
            fila_binaria.append(1)
            contador_unos += 1
        else:
            fila_binaria.append(0)
            contador_ceros += 1
    B.append(fila_binaria)
print("\nMatriz binaria invertida:")
```

```
for fila in B:
    print(fila)
print(f"\nCantidad de unos: {contador_unos}")
print(f"Cantidad de ceros: {contador_ceros}")
```

```
Ingrese el número de filas: 5
Ingrese el número de columnas: 7
Matriz original:
[3, 2, 10, 14, 0, 20, 14]
[14, 6, 24, 17, 16, 18, 4]
[9, 19, 24, 17, 9, 10, 2]
[4, 1, 22, 16, 18, 25, 0]
[7, 26, 11, 24, 8, 15, 23]

Matriz binaria invertida:
[0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0]

Cantidad de unos: 2
Cantidad de ceros: 33
```

```
import random as rd
matriz_nombres = []
filas = int(input("Ingrese el número de filas: "))
columnas = int(input("Ingrese el número de columnas: "))
for i in range(filas):
    fila = []
    for j in range(columnas):
        nombre = input(f"Ingrese el nombre para la posición ({i+1},{j+1}): "
        fila.append(nombre)
    matriz_nombres.append(fila)
print("\nMatriz de nombres:")
for fila in matriz_nombres:
    print(fila)
nombre_buscar = input("\nIngrese el nombre que desea buscar: ")
posiciones = []
for i in range(filas):
    for j in range(columnas):
        if matriz_nombres[i][j] == nombre_buscar:
            posiciones.append((i, j))
if posiciones:
    print(f"\nEl nombre '{nombre_buscar}' se encuentra en las posiciones:")
    for pos in posiciones:
        print(f"Fila {pos[0]+1}, Columna {pos[1]+1}")
else:
    print(f"\nEl nombre '{nombre_buscar}' no se encontró en la matriz.")
```

```
Ingrese el número de filas: 2
Ingrese el número de columnas: 2
Ingrese el nombre para la posición (1,1): pedro
Ingrese el nombre para la posición (1,2): juan
Ingrese el nombre para la posición (2,1): maria
```

```
Ingrese el nombre para la posición (2,2): maria

Matriz de nombres:
['pedro', 'juan']
['maria', 'maria']

Ingrese el nombre que desea buscar: maria

El nombre 'maria' se encuentra en las posiciones:
Fila 2, Columna 1
Fila 2, Columna 2
```

```
import random as rd

A = []
m = int(input("Ingrese el número de filas: "))
n = int(input("Ingrese el número de columnas: "))

for i in range(m):
    fila = []
    for j in range(n):
        num = rd.randint(0, 28)
        fila.append(num)
    A.append(fila)

print("\nMatriz original:")
for fila in A:
    print(fila)

contador = {}

for fila in A:
    for num in fila:
        if num in contador:
            contador[num] += 1
        else:
            contador[num] = 1
print("\nCantidad de repeticiones de cada número:")
for num, veces in sorted(contador.items()):
    print(f"Número {num} → {veces} veces")
```

```
Ingrese el número de filas: 5
Ingrese el número de columnas: 6

Matriz original:
[13, 18, 7, 0, 2, 9]
[0, 15, 21, 27, 17, 0]
[3, 3, 9, 6, 5, 7]
[2, 23, 8, 16, 20, 0]
[23, 3, 0, 6, 16, 3]

Cantidad de repeticiones de cada número:
Número 0 → 5 veces
Número 2 → 2 veces
Número 3 → 4 veces
Número 5 → 1 veces
Número 6 → 2 veces
```

```
Número 7 → 2 veces
Número 8 → 1 veces
Número 9 → 2 veces
Número 13 → 1 veces
Número 15 → 1 veces
Número 16 → 2 veces
Número 17 → 1 veces
Número 18 → 1 veces
Número 20 → 1 veces
Número 21 → 1 veces
Número 23 → 2 veces
Número 27 → 1 veces
```

Ejercio

```
import random as rd
A = []
m = int(input("Ingrese el número de filas: "))
n = int(input("Ingrese el número de columnas: "))
for i in range(m):
    filas = []
    for j in range(n):
        num = rd.randint(0, 28)
        filas.append(num)
    A.append(filas)
print("\nMatriz:")
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
todos = []
for i in range(m):
    for j in range(n):
        todos.append(A[i][j])
vector1 = []
for num in todos:
    if num not in vector1:
        vector1.append(num)
for i in range(len(vector1)):
    for j in range(i + 1, len(vector1)):
        if vector1[i] > vector1[j]:
            vector1[i], vector1[j] = vector1[j], vector1[i]
vector2 = []
for num in vector1:
    contador = 0
    for x in todos:
        if x == num:
            contador += 1
    vector2.append(contador)
print("\nVector 1 (valores sin repetir y ordenados):")
print(vector1)
print("\nVector 2 (veces que se repite cada valor):")
print(vector2)
```

```
Ingrese el número de filas: 5
Ingrese el número de columnas: 3

Matriz:
6       24      19
25      7       1
1       11      14
27      28      0
24      5       0

Vector 1 (valores sin repetir y ordenados):
[0, 1, 5, 6, 7, 11, 14, 19, 24, 25, 27, 28]

Vector 2 (veces que se repite cada valor):
[2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1]
```

19/01/2026 CLASE#13

```
import random as rd
n=int(input('Ingrese las filas:'))
m=int(input('Ingrese las columnas:'))
A=[]
#INSERTAR DATOS
for i in range(m):
    filas=[]
    for j in range(n):
        num=rd.randint(0,10)
        filas.append(num)
    A.append(filas)
#ENCONTRAR LA SUMA DE LA DIAGONAL PRINCIPAL
acu=0
if m==n:
    for i in range(m):
        for j in range(n):
            if i==j:
                acu=acu+A[i][j]
else:
    print('La matriz no es cuadrada')

#IMPRIMIR MATRIZ
f  or i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
print(f'La suma de la diagonal principal es: {acu}')
```

```
Ingrese las filas:3
Ingrese las columnas:3
4       7       4
4       3       7
6       9       6
La suma de la diagonal principal es: 13
```

```
import random as rd

n = int(input('Ingrese las filas: '))
m = int(input('Ingrese las columnas: '))
A = []

# INSERTAR DATOS
for i in range(n):
    filas = []
    for j in range(m):
        num = rd.randint(0, 10)
        filas.append(num)
    A.append(filas)

# SUMA DE LA DIAGONAL PRINCIPAL
suma = 0
if n == m:
    for i in range(n):
        suma += A[i][i]
else:
    print('La matriz no es cuadrada')

# IMPRIMIR MATRIZ
for fila in A:
    for valor in fila:
        print(valor, end="\t")
    print()

print(f'La suma de la diagonal principal es: {suma}')
```

```
Ingrese las filas: 5
Ingrese las columnas: 5
6       2       10      1       9
1       10      8       0       6
10      5       3       7       1
4       1       2       9       1
7       10      3       6       3
La suma de la diagonal principal es: 31
```

```
import random as rd

n = int(input('Ingrese las filas: '))
m = int(input('Ingrese las columnas: '))
A = []

# INSERTAR DATOS
for i in range(n):
    filas = []
    for j in range(m):
        num = rd.randint(0, 10)
        filas.append(num)
    A.append(filas)

# SUMA DE LA DIAGONAL (i + j = n - 1)
```

```
suma = 0
if n == m:
    for i in range(n):
        for j in range(n):
            if i + j == n - 1:
                suma += A[i][j]
else:
    print('La matriz no es cuadrada')

# IMPRIMIR MATRIZ
for fila in A:
    for valor in fila:
        print(valor, end="\t")
    print()

print(f'La suma de la diagonal es: {suma}')
```

```
Ingrese las filas: 4
Ingrese las columnas: 4
4       4       0       6
6       7       7       1
10      8       7       9
9       5       8       7
La suma de la diagonal es: 30
```

```
import random as rd

n = int(input('Ingrese las filas: '))
m = int(input('Ingrese las columnas: '))
A = []

# INSERTAR DATOS
for i in range(n):
    filas = []
    for j in range(m):
        num = rd.randint(0, 10)
        filas.append(num)
    A.append(filas)

# SUMA DE LA DIAGONAL (i + j = n - 1)
acu = 0
if n == m:
    for i in range(n):
        for j in range(n):
            if i + j == n - 1:
                acu = acu + A[i][j]
else:
    print('La matriz no es cuadrada')

# IMPRIMIR MATRIZ
for i in range(n):
    for j in range(m):
        print(A[i][j], end="\t")
    print()
```

```
print(f'La suma de la diagonal es: {acu}')
```

```
Ingrese las filas: 8
Ingrese las columnas: 8
8       9       0       9       9       6       7       0
9       1       8       10      8       8       9       0
9       9       5       3       5       2       0       8
1       1       3       1       4       4       8       10
1       2       7       8       8       9       3       9
9       0       10      1       9       2       10      7
2       3       6       8       6       2       3       8
10      3       10      7       0       7       9       5
La suma de la diagonal es: 46
```

```
import random as rd
n=int(input('Ingrese las filas:'))
m=int(input('Ingrese las columnas:'))
A=[]
#INSERTAR DATOS
for i in range(m):
    filas=[]
    for j in range(n):
        num=rd.randint(0,10)
        filas.append(num)
    A.append(filas)
#ENCONTRAR LA SUMA DE LA DIAGONAL PRINCIPAL
acu=0
acu2=0
if m==n:
    for i in range(m):
        for j in range(n):
            if i==j:
                acu=acu+A[i][j]
            if (i+j)==(n-1):
                acu2=acu2+A[i][j]
else:
    print('La matriz no es cuadrada')

#IMPRIMIR MATRIZ
for i in range(m):
    for j in range(n):
        print(A[i][j], end="\t")
    print()
print(f'La suma de la diagonal principal es: {acu}')
print(f'La suma de la diagonal secundaria es: {acu2}')
```

```
Ingrese las filas:2
Ingrese las columnas:2
2       2
8       4
La suma de la diagonal principal es: 6
La suma de la diagonal secundaria es: 10
```

PREGUNTA DE EXAMEN

Modulizar: utilizar funciones

RECICLAGE DE CODIGOS: No repetir muchas veces el mismo codigo

1.- for i in range (5): ---> cabecera

2.- aw=aw+i -----> cuerpo

def nombreFuncion(parametros)

3.- Llamar a la función

```
def mensaje():
    print("hola amigos")

mensaje()
mensaje()
mensaje()
```

```
hola amigos
hola amigos
hola amigos
```

```
def mensaje():
    num=int(input("ingrese un numero"))
    print(f"el numero que ingreso es {num}:")

mensaje()
mensaje()
mensaje()
```

```
ingrese un numero8
el numero que ingreso es 8:
ingrese un numero5
el numero que ingreso es 5:
ingrese un numero4
el numero que ingreso es 4:
```

21/1/2026 CLASE #14

```
def mensaje():
    num=int(input("ingrese numero: "))
    print(f"el numero es {num}")

mensaje()
mensaje()
mensaje()
```

```
ingrese numero: 5
el numero es 5
ingrese numero: 4
```

```
el numero es 4
ingrese numero: 5
el numero es 5
```

AREA DE UN TRIANGULO

```
def areaT (b,h):
    a=(b*h)/2
    return a

respuesta1=areaT(3,6)
print(respuesta1)
respuesta2=areaT(4,7)
print(respuesta2)
```

```
9.0
14.0
```

realizar una función que sume dos números como parámetros y que me retorne la suma

```
def sumar(a, b):
    p=a+b
    return a + b
resultado = sumar(5, 3)
print(resultado)
```

```
8
```

```
def areaT(a, b):
    p=(a*b)/2
    return p
num1=int(input("ingrese la base:"))
num2=int(input("ingrese la altura:"))
respuesta=areaT(num1,num2)
print(respuesta)

j=int(input("ingrese otra base:"))
k=int(input("ingrese otra altura:"))
respuesta2=areaT(j,k)
print(respuesta2)
```

```
def facto(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
num=int(input('Ingrese el Número:'))
```

```
respuesta=facto(num)
print(f'El factorial de {num} es {respuesta}')
```

```
Ingrese el Número:89
El factorial de 89 es 165079551609084610812169192624536193098396662364965418
```

```
def coeficiente_binomial(m, n):
    resultado = 1
    for i in range(1, n + 1):
        resultado = resultado * (m - i + 1) // i
    return resultado
print(coeficiente_binomial(5, 2))
```

```
10
```

```
def factorial(x):
    resultado = 1
    for i in range(1, x + 1):
        resultado *= i
    return resultado

def coeficiente_binomial(m, n):
    return factorial(m) // (factorial(n) * factorial(m - n))
print(coeficiente_binomial(5, 2))
```

```
10
```

```
def facto(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
num=int(input('Ingrese el Número:'))
respuesta=facto(num)
print(f'El factorial de {num} es {respuesta}')
m=5
n=2
coefBinomial=(facto(m))/(facto(n)*facto(m-n))
print(f'El coeficiente binomial es {coefBinomial}')
```

```
Ingrese el Número:9
El factorial de 9 es 362880
El coeficiente binomial es 10.0
```

Crear una funcion que tenga dos parametros y que me resuelva la siguiente formula r=a**b

```
def potencia(a, b):
    resultado = 1
    for i in range(b):
```

```
            resultado = resultado * a
        return resultado

    a = int(input("Ingrese la base (a): "))
    b = int(input("Ingrese el exponente (b): "))

    res = potencia(a, b)
    print(res)
```

```
Ingrese la base (a): 8
Ingrese el exponente (b): 4
4096
```

26/01/200/26 CLASE#15

```
a=int(input("Ingrese m:"))
b=int(input("Ingrese n:"))
respuesta=coefBinomial(a,b)
print(f"La respuesta es {respuesta}")
```

```
import Funciones as fc
a=int(input("Ingrese m:"))
b=int(input("Ingrese n:"))
respuesta=fc.coefBinomial(a,b)
print(f"La respuesta es {respuesta}")

num=int(input("Ingrese un numero:"))
fact=fc.fact(num)
print(f"El factorial de {num}!={fact}")
```

```
def fact(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
def coefBinomial(m,n):
    cB=fact(m)/(fact(n)*fact(m-n))
    return cB
```

PRINCIPAL.PY

```
import FUNCIONES as FUNCIONES
num = int(input("Ingrese tamaño de vector: "))
vector = FUNCIONES.crearVector(num)
print(vector)
print(f"El mayor es {FUNCIONES.mayorVector(vector)}")
num2 = int(input("Ingrese el tamaño de otro vector: "))
Vector2 = FUNCIONES.crearVector(num2)
print(Vector2)
print(f"El mayor es {FUNCIONES.mayorVector(Vector2)}")
print(f"El menor es {FUNCIONES.menorVector(Vector2)}")
```

```
n = int(input("Ingrese tamaño de los vectores: "))
vector1 = FUNCIONES.crearVector(n)
vector2 = FUNCIONES.crearVector(n)
print("Vector 1:", vector1)
print("Vector 2:", vector2)
print("Producto punto:", FUNCIONES.productoPunto(vector1, vector2))
```

FUNCIONES.PY

```
import random as rd
def crearVector(n):
    vec = []
    for _ in range(n):
        num = rd.randint(0, 10)
        vec.append(num)
    return vec


def mayorVector(vec):
    mayor = vec[0]
    for i in range(1, len(vec)):
        if vec[i] > mayor:
            mayor = vec[i]
    return mayor


def menorVector(vec):
    menor = vec[0]
    for i in range(1, len(vec)):
        if vec[i] < menor:
            menor = vec[i]
    return menor


def productoPunto(vec1, vec2):
    producto = 0
    for i in range(len(vec1)):
        producto += vec1[i] * vec2[i]
    return producto
```

28/01/2026 CLASE#16

Depuración del google colab

Ejemplo

```
def areaT(b, h):
    a = (b * h) / 2
    return a
```

```
respuesta1 = areaT(3, 6)
print(respuesta1)

respuesta2 = areaT(4, 7)
print(respuesta2)
```

```
9.0
14.0
```

Depuración usando print

```
def areaT(b, h):
    print("Valor de b:", b)
    print("Valor de h:", h)

    a = (b * h) / 2
    print("Área calculada:", a)

    return a

respuesta1 = areaT(3, 6)
print("Resultado final:", respuesta1)

respuesta2 = areaT(4, 7)
print("Resultado final:", respuesta2)
```

```
Valor de b: 3
Valor de h: 6
Área calculada: 9.0
Resultado final: 9.0
Valor de b: 4
Valor de h: 7
Área calculada: 14.0
Resultado final: 14.0
```

28/01/2026 CLASE#17

FUNCIONES

```
def areat(b, h):
 a=(b*h)/2
 return a
num1=int(input("Ingrese la base:"))
num2=int(input("Ingrese la altura:"))
respuesta=areat(num1, num2)
print(respuesta)
```

```
Ingrese la base:5
Ingrese la altura:2
```

```
5.0
```

PRINCIPAL

```
import funciones as fc
num1=int(input("ingrese la base"))
num2=int(input("ingrese la altura"))
respuesta=fc.areat(num1,num2)
print(respuesta)
```

PRINCIPAL

```
import funciones as fc
num1=int(input("ingrese el tamaño del vector1"))
fc.crearVector(num1)
num2=int(input("ingrese el tamaño del vector2"))
fc.crearVector(num2)
```

FUNCIONES

```
import random as rd
def crearVector (n):
    vec=[]
    for i in range (n):
        num=rd.randint(0,10)
        vec.append(num)
    print(vec)
    return vec
```

29/01/2026 CLASE#18

vector--->vec---->len(vec)---->tamaño del vector

matriz--->A--->lent(A)--->fila de la matriz

---------->lent(A[0])--->columnas de la matriz

```
import random as rd
#funcion crear matriz con datos aleatoreos
def crearMatriz (m,n):
    A=[]
    for i in range (m):
        filas=[]
        for j in range(n):
            num=rd.randint(0,20)
            filas.append(num)
        A.append(filas)
```

```
        return A
A=[]
m=int(input("ingrese el numero de filas:"))
n=int(input("ingrese el numero de columnas"))
#creacion de matriz e insercion de datos
for i in range (m):
    filas=[]
    for j in range (n):
        num=rd.randint(0,20)
        filas.append(num)
    A.append(filas)
#impresion de la matriz
for i in range(m):
    for j in range (n):
        print(A[i][j], end="\t")
```

```
ingrese el numero de filas:3
ingrese el numero de columnas4
10      15      7       4       3       20      1       10      18      13
```

TODO

```
import random as rd
#Función Crear Matriz con datos
def crearMatriz(m,n):
    A = []
    for i in range(m):
        filas = []
        for j in range(n):
            num = rd.randint(0, 20)
            filas.append(num)
        A.append(filas)
    return A

#Función Imprimir Matriz
def imprimirMatriz(A):
    m=len(A)
    n=len(A[0])
    for i in range(m):
        for j in range(n):
            print(A[i][j],end="\t")
        print()
print('PRIMERA MATRIZ')
imprimirMatriz(crearMatriz(3,5))
print('SEGUNDA MATRIZ')
imprimirMatriz(crearMatriz(4,8))
```

FUNCIONES

```
import Funciones as fn
print("primera matriz")
```

```
fn.imprimirMatriz(fn.crearMatriz( 3 , 5))
print("segunda matriz")
fn.imprimirMatriz(fn.crearMatriz(4,8))
```

Pistas para el examen

no interaccion

20 preguntas de reactivos

funciones

contros repetitivas

vectores y matrices

primeras 10 una sola opcion, 10 siguientes mas de dos opciones

---

Practica 12pts

1 ejercicios

matriz

principal y funciones---> como en clase

pares impares

filas columnas

modulos

analisis de las filas,las columnas y las diagonales de la matriz

Repaso para el examen

PRINCIPAL

```
import FUNCIONES as FUNCIONES
num = int(input("Ingrese tamaño de vector: "))
vector = FUNCIONES.crearVector(num)
print(vector)
print(f"El mayor es {FUNCIONES.mayorVector(vector)}")
num2 = int(input("Ingrese el tamaño de otro vector: "))
Vector2 = FUNCIONES.crearVector(num2)
print(Vector2)
print(f"El mayor es {FUNCIONES.mayorVector(Vector2)}")
print(f"El menor es {FUNCIONES.menorVector(Vector2)}")
n = int(input("Ingrese tamaño de los vectores: "))
vector1 = FUNCIONES.crearVector(n)
vector2 = FUNCIONES.crearVector(n)
print("Vector 1:", vector1)
print("Vector 2:", vector2)
print("Producto punto:", FUNCIONES.productoPunto(vector1, vector2))
```

FUNCIONES

```
import random as rd
def crearVector(n):
    vec = []
    for _ in range(n):
        num = rd.randint(0, 10)
        vec.append(num)
    return vec


def mayorVector(vec):
    mayor = vec[0]
    for i in range(1, len(vec)):
        if vec[i] > mayor:
            mayor = vec[i]
    return mayor


def menorVector(vec):
    menor = vec[0]
    for i in range(1, len(vec)):
        if vec[i] < menor:
            menor = vec[i]
    return menor


def productoPunto(vec1, vec2):
    producto = 0
    for i in range(len(vec1)):
        producto += vec1[i] * vec2[i]
    return producto
```

PRINCIPAL

```
import FUNCIONES as FUNCIONES

num = int(input("Ingrese tamaño de vector: "))
vector = FUNCIONES.crearVector(num)
print("Vector:", vector)
print(f"El mayor es {FUNCIONES.mayorVector(vector)}")
print(f"El módulo del vector es {FUNCIONES.moduloVector(vector)}")

num2 = int(input("\nIngrese el tamaño de otro vector: "))
vector2 = FUNCIONES.crearVector(num2)
print("Vector 2:", vector2)
print(f"El mayor es {FUNCIONES.mayorVector(vector2)}")
print(f"El menor es {FUNCIONES.menorVector(vector2)}")
print(f"El módulo del vector es {FUNCIONES.moduloVector(vector2)}")

n = int(input("\nIngrese tamaño de los vectores para producto punto: "))
```

```
vector1 = FUNCIONES.crearVector(n)
vector2 = FUNCIONES.crearVector(n)
print("Vector 1:", vector1)
print("Vector 2:", vector2)
print("Producto punto:", FUNCIONES.productoPunto(vector1, vector2))

m = int(input("\nIngrese tamaño de la matriz cuadrada: "))
matriz = FUNCIONES.crearMatriz(m)

print("\nMatriz:")
FUNCIONES.imprimirMatriz(matriz)

print("\nSuma de filas:", FUNCIONES.sumaFilas(matriz))
print("Suma de columnas:", FUNCIONES.sumaColumnas(matriz))
print("Diagonal principal:", FUNCIONES.diagonalPrincipal(matriz))
```

FUNCIONES

```
import random
import math

def crearVector(n):
    return [random.randint(-10, 10) for _ in range(n)]

def mayorVector(v):
    return max(v)

def menorVector(v):
    return min(v)

def productoPunto(v1, v2):
    return sum(v1[i] * v2[i] for i in range(len(v1)))

def moduloVector(v):
    return round(math.sqrt(sum(x**2 for x in v)), 2)

def crearMatriz(n):
    return [[random.randint(0, 9) for _ in range(n)] for _ in range(n)]

def imprimirMatriz(M):
    for fila in M:
        for elemento in fila:
            print(elemento, end="\t")
        print()

def sumaFilas(M):
```