

# **Diplomado En Programación Básica**

Universidad Autónoma de Chiapas  
Centro Mesoamericano de Física Teórica

Michael Steven Paucar Rojas

---

# **MATHEMATICA**

---



**WOLFRAM**

---

## 1. Introducción

El presente cuaderno constituye un recurso de apoyo para el aprendizaje de Mathematica orientado a la programación y al uso de sus principales funciones en contextos académicos y prácticos. El contenido se organiza de manera progresiva iniciando con operaciones básicas sobre listas, expresiones matemáticas y representaciones gráficas para avanzar hacia temas más complejos como manejo de entidades, conversiones de unidades, generación de visualizaciones interactivas y aplicaciones en análisis de datos.

El enfoque seguido combina teoría con ejemplos prácticos que buscan ilustrar no solo la sintaxis del lenguaje sino también la lógica detrás de cada comando. Se ha procurado mantener una estructura clara donde cada sección incluye subtítulos, descripciones y comentarios en el código para facilitar la comprensión. Esto permite que el material pueda ser utilizado tanto por estudiantes en formación como por interesados en explorar las capacidades del software en distintos escenarios.

Cabe señalar que el documento reúne apuntes propios sistematizados a partir del estudio y la práctica personal. Estos apuntes no reemplazan la documentación oficial de Mathematica pero sí constituyen un complemento útil para guiar el aprendizaje y servir como referencia en la resolución de ejercicios y proyectos futuros.

---

## 2. Tabla de contenidos

### 1. Introducción

### 2. Tabla de contenidos

### 3. Clase 1 — Introducción a Wolfram Mathematica

#### 3.1. Captura y análisis de imagen

### 4. Clase 2 — Comandos básicos, listas y entidades

#### 4.1. Comandos del sistema

#### 4.2. Comandos interactivos

#### 4.3. Entidades: países y banderas

#### 4.4. Exploración planetaria

#### 4.5. Conversiones de unidades y monedas

#### 4.6. Listas: creación y operaciones básicas

#### 4.7. Funciones para secuencias y combinación de listas

#### 4.8. Manipulación avanzada de listas

#### 4.9. Funciones adicionales sobre listas

### 5. Clase 3 — Gráficos, colores y funciones trigonométricas

#### 5.1. Gráficas estadísticas (barras y pastel)

#### 5.2. Selección y manipulación de datos para visualización

#### 5.3. Colores y estilos gráficos (paletas y transformaciones)

#### 5.4. Funciones matemáticas básicas y plots elementales

### 6. Clase 4 — Funciones Trascendentes

#### 6.1. Expansión de expresiones trigonométricas

#### 6.2. Números complejos

#### 6.3. Logaritmos

#### 6.4. Exponenciales

#### 6.5. Series

#### 6.6. Límites

#### 6.7. Funciones

#### 6.8. Derivadas

#### 6.9. Integrales

#### 6.10. Notación de Lagrange

#### 6.11. Integración Numérica

#### 6.12. Tablas

#### 6.13. Gráfica de Tablas

### 7. Clase 5 — Visualización Matemática Interactiva

#### 7.1. Gráficas Bidimensionales (2D)

#### 7.2. Gráficas Tridimensionales (3D)

### 7.3. Manipuladores Interactivos

## 8. Clase 6 — Álgebra Simbólica y Series Numéricas

### 8.1. Solución de ecuaciones

### 8.2. Manipulación algebraica

### 8.3. Series Numéricas

## 9. Clase 7 — Variable Compleja

### 9.1. Números Complejos

### 9.2. Conversión de la forma Polar a Rectangular

### 9.3. Conversión de la forma Rectangular a Polar

### 9.4. Gráficas de Números Complejos

## 10. Clase 8 — Álgebra Lineal

### 10.1. Definición y creación de matrices

### 10.2. Operaciones básicas con matrices

### 10.3. Acceso a elementos

### 10.4. Operaciones avanzadas con matrices y vectores

### 10.5. Programación básica en Mathematica

## 11. Clase 9 — Álgebra Lineal

### 11.1. Operadores condicionales

### 11.2. Condicional if

## 12. Tareas

### 12.1. Tarea 1 — Cálculos Numéricos y Funciones en Mathematic

### 12.2. Tarea 2 — Formato de Notebook

### 12.3. Tarea 3 — Aplicaciones de Funciones Trascendentes

### 12.4. Tarea 4 — Esferas 3D

### 12.5. Tarea 5 — Repaso general en Mathematica

### 12.6. Tarea 6 — Solución de ecuaciones

### 12.7. Tarea 7 — Variable Compleja


### 12.8. Tarea 8 — Reto Matrices

### 12.9. Tarea 9 — Aplicación del condicional if

## 13. Apéndice

### 13.1. Comandos comunes

## 10. Clase 8 — Álgebra Lineal

 2025/10/15

### ⚡ Introducción:

En esta clase se exploran los conceptos fundamentales de matrices y vectores en Wolfram Mathematica. Se abordan desde su definición y creación hasta operaciones más avanzadas, permitiendo trabajar con estructuras matemáticas esenciales en álgebra lineal. Se enseñan tanto las operaciones básicas como la manipulación de elementos específicos dentro de matrices y vectores. Además, se profundiza en operaciones avanzadas, como determinantes, trazas, inversas, y valores propios. Finalmente, se introduce la programación básica en Mathematica, abordando el uso de comandos simples y funciones para resolver problemas matemáticos y manipular datos de manera eficiente.

### 📋 Objetivos de la clase:

- Definir y crear matrices utilizando las funciones *MatrixForm*, *RandomReal*, y *Array*.
- Realizar operaciones básicas con matrices, como suma, resta, multiplicación y el cálculo de norma.
- Acceder a elementos específicos de una matriz y vector utilizando la sintaxis de doble corchete `[[i,j]]`, y modificar estos elementos con *ReplacePart* y *ReplaceAll*.
- Aplicar operaciones avanzadas como el cálculo de determinantes, traza, inversa, y valores propios mediante funciones como *Det*, *Trace*, *Inverse*, y *Eigenvalues*.
- Introducir la programación básica en Mathematica, manejando comandos de entrada y salida, y creando funciones personalizadas para resolver cálculos complejos de manera automática y eficiente.

## 10.1 Definición y creación de matrices

✎ **Explicación:** En Mathematica, las matrices se definen como listas de listas, donde cada sublista representa una fila. Se pueden crear directamente asignando valores o usando la función `Table`, que genera matrices programáticamente mediante índices de fila ( $i$ ) y columna ( $j$ ), permitiendo construir patrones específicos en los elementos.

*# Matriz 2x3 con dos filas y tres columnas*

```
In[ ]:= {{1, 2, 5}, {3, 4, 6}}
```

```
Out[ ]:= {{1, 2, 5}, {3, 4, 6}}
```

*# Forma 1: Convertir a, Forma Tradicional*

```
Out[ ]:= 
$$\begin{pmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{pmatrix}$$

```

*# Crea una matriz 5x5 donde cada elemento es el producto fila\*columna;  
i recorre filas de 1 a 5, j recorre columnas de 1 a 5*

```
In[ ]:= Matriz1 = Table[i*j, {i, 1, 5}, {j, 1, 5}]
```

```
Out[ ]:= 
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 4 & 8 & 12 & 16 & 20 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix}$$

```

*# Crea una matriz 5x5 donde cada elemento es i + j;  
i recorre filas de 1 a 5, j recorre columnas de 1 a 5*

```
In[ ]:= Matriz2 = Table[i + j, {i, 1, 5}, {j, 1, 5}]
```

```
Out[ ]:= 
$$\begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \\ 6 & 7 & 8 & 9 & 10 \end{pmatrix}$$

```

```
In[ ]:= Matriz2[[4, 4]]
```

```
Out[ ]:= 8
```

*# Forma 2: MatrixForm*

```
In[*]:= MatrixForm[{{1, 2, 5}, {3, 4, 6}}]
|forma de matriz
```

```
Out[*]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{pmatrix}$$

```

```
In[*]:= MatrixForm[{{-2, 5}, {0, 1}}]
|forma de matriz
```

```
Out[*]//MatrixForm=

$$\begin{pmatrix} -2 & 5 \\ 0 & 1 \end{pmatrix}$$

```

```
In[*]:= MatrixForm[{{1, -1}, {2, 3}}]
|forma de matriz
```

```
Out[*]//MatrixForm=

$$\begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}$$

```

```
# Asigna a uno una matriz 2x2 con elementos enteros
```

```
In[*]:= uno = {{-2, 5}, {0, 1}}
```

```
Out[*]=

$$\begin{pmatrix} -2 & 5 \\ 0 & 1 \end{pmatrix}$$

```

```
# Asigna a dos una matriz 2x2 con elementos enteros, incluyendo negativos
```

```
In[*]:= dos = {{1, -1}, {2, 3}}
```

```
Out[*]=

$$\begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}$$

```

## 10.2 Operaciones básicas con matrices

✎ **Explicación:** Mathematica permite realizar varias operaciones con matrices. La multiplicación elemento a elemento se hace con `*`, multiplicando cada par de elementos correspondientes. La multiplicación matricial estándar se realiza con `.` o `Dot`, siguiendo las reglas del álgebra lineal. Además, la función *Transpose* invierte filas y columnas, transformando la matriz y siendo esencial en muchas aplicaciones matemáticas y científicas.

```
# Multiplica elemento a elemento las matrices uno y dos
```

```
In[*]:= uno * dos
```

```
Out[*]=

$$\begin{pmatrix} -2 & -5 \\ 0 & 3 \end{pmatrix}$$

```

```
# Producto matricial estándar de las matrices uno y dos
```

```
In[ ]:= uno.dos
```

```
Out[ ]:=
```

$$\begin{pmatrix} 8 & 17 \\ 2 & 3 \end{pmatrix}$$

💎 **Nota:** *MatrixForm* solo da formato visual, no realiza operaciones. Por eso, la multiplicación debe hacerse con las matrices originales usando `.` y aplicar *MatrixForm* solo al resultado si se desea visualizar.

```
# Asigna a A una matriz 3x3 con elementos enteros
```

```
In[ ]:= A = {{1, 1, 1}, {3, 2, 1}, {1, 2, 3}}
```

```
Out[ ]:=
```

$$\begin{pmatrix} 1 & 1 & 1 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

```
# Asigna a B una matriz 3x3 con elementos enteros
```

```
In[ ]:= B = {{1, 2, 3}, {2, 4, 6}, {1, 2, 3}}
```

```
Out[ ]:=
```

$$\{\{1, 2, 3\}, \{2, 4, 6\}, \{1, 2, 3\}\}$$

```
# Producto matricial estándar de las matrices A y B
```

```
In[ ]:= A.B
```

```
Out[ ]:=
```

$$\begin{pmatrix} 4 & 8 & 12 \\ 8 & 16 & 24 \\ 8 & 16 & 24 \end{pmatrix}$$

```
# Producto matricial estándar de las matrices B y A
```

```
In[ ]:= B.A
```

```
Out[ ]:=
```

$$\begin{pmatrix} 10 & 11 & 12 \\ 20 & 22 & 24 \\ 10 & 11 & 12 \end{pmatrix}$$

```
# Devuelve la matriz traspuesta de Matriz4, intercambiando filas por columnas
```



```
In[ ]:= Transpose[A]
|transposición
```

```
Out[ ]:=
```

$$\begin{pmatrix} 1 & 3 & 1 \\ 1 & 2 & 2 \\ 1 & 1 & 3 \end{pmatrix}$$

## 10.3 Acceso a elementos

✎ **Explicación:** Para obtener un elemento específico de una matriz, se usa la sintaxis de doble corchete `[[i, j]]`, donde *i* es la fila y *j* la columna. Esto facilita extraer o modificar valores particulares dentro de la matriz de forma precisa.

```
# Accede al elemento en la fila 2, columna 2 de la matriz A
```

```
In[ ]:= A[[2, 2]]
```

```
Out[ ]:=
```

2

```
# Accede al elemento en la fila 3, columna 3 de la matriz A
```

```
In[ ]:= A[[3, 3]]
```

```
Out[ ]:=
```

3

```
# Accede al elemento en la fila 2, columna 3 de la matriz B
```

```
In[ ]:= B[[2, 3]]
```

```
Out[ ]:=
```

6

### ■ 🎯 Reto en clase — Matrices

Realizar una matriz 3 x 5 con números consecutivos del [1 – 15] y otra matriz de 4 x 4 números consecutivos del [1 – 16].

```
# Crea una matriz 3x5 donde cada elemento es (fila - 1)*5 + columna;
i recorre filas de 1 a 3, j recorre columnas de 1 a 5
```

```
In[ ]:= Matriz3 = Table[(i - 1) * 5 + j, {i, 1, 3}, {j, 1, 5}]
|tabla
```

```
Out[ ]:=
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

```
# Crea una matriz 4x4 donde cada elemento es (fila - 1)*4 + columna;
i recorre filas de 1 a 4, j recorre columnas de 1 a 4
```

```
In[ ]:= Matriz4 = Table[(i - 1) * 4 + j, {i, 1, 4}, {j, 1, 4}]
```

tabla

Out[ ]:=

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

```
# Crea una matriz 4x4 donde cada elemento es (columna - 1)*4 + fila;
i recorre filas de 1 a 4, j recorre columnas de 1 a 4
# Otra forma de Matriz Transpuesta
```

```
In[ ]:= Matriz5 = Table[(j - 1) * 4 + i, {i, 1, 4}, {j, 1, 4}]
```

tabla

Out[ ]:=

$$\begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

## 10.4. Operaciones avanzadas con matrices y vectores

✧ **Explicación:** Las operaciones avanzadas con matrices y vectores incluyen cálculos como el determinante, traza, inversa, y el cálculo de valores propios y vectores propios. Estas operaciones son fundamentales en álgebra lineal y son utilizadas en diversas áreas de las ciencias y la ingeniería. Para calcular, por ejemplo, el determinante de una matriz  $A$ , se usa el comando  $Det[A]$ . Para obtener los valores propios y vectores propios de una matriz, se usa  $Eigenvalues[A]$  y  $Eigenvecs[A]$ , respectivamente. Además, las operaciones como la multiplicación de matrices ( $A \cdot B$ ), la suma ( $A + B$ ), y la resta ( $A - B$ ) permiten manipular matrices de manera sencilla, mientras que  $Norm[v]$  calcula la norma de un vector.

```
# OverBar no realiza la operación de Conjugados
```

```
In[ ]:= OverBar[3 + 4 I] * OverBar[3 + 7 I]
```

sobre barra    n...    sobre barra    núr

Out[ ]:=

$$\overline{3 + 4 i} \overline{3 + 7 i}$$

```
# Definir matriz de forma tradicional para cualquier operación
```

```
In[ ]:= Matriz5 = {{1, 2, 3}, {4, 5, 6}, {7, 9, 9}}
```

Out[ ]:=

$$\{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 9, 9\}\}$$

```
# Transpuesta de una matriz en forma estándar
```

```
In[ ]:= Transpose[Matriz5] // MatrixForm
      |transposición      |forma de matriz
```

```
Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 9 \\ 3 & 6 & 9 \end{pmatrix}$$

```

```
# Matriz de complejos con salida estándar
```

```
In[ ]:= m = {{1, 2 I, 3}, {3 + 4 I, 5, I}} // MatrixForm
      |número i      |número... |número... |forma de matriz
```

```
Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 i & 3 \\ 3 + 4 i & 5 & i \end{pmatrix}$$

```

```
# Definimos la matriz para operar
```

```
In[ ]:= m = {{1, 2 I, 3}, {3 + 4 I, 5, I}}
      |número i      |número... |número...
```

```
Out[ ]:=
{{1, 2 i, 3}, {3 + 4 i, 5, i}}
```

```
# Definimos la matriz para operar
```

```
In[ ]:= m = {{1, 2 I, 3}, {3 + 4 I, 5, I}}
      |número i      |número... |número...
```

```
# Obtenemos la Conjugada y Transpuesta de m
```

```
In[ ]:= ConjugateTranspose[m] // MatrixForm
      |transpuesto conjugado      |forma de matriz
```

```
Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 3 - 4 i \\ -2 i & 5 \\ 3 & -i \end{pmatrix}$$

```

```
# Asignación a Matrix8
```

```
In[ ]:= Matrix8 = {{1.4, 2}, {3, -6.7}}
```

```
Out[ ]:=
{{1.4, 2}, {3, -6.7}}
```

```
..# Sacamos la inversa de Matriz Matrix8
```

```
In[ ]:= Matrix10 = Inverse[Matrix8]
      |matriz inversa
```

```
Out[ ]:=
{{0.435631, 0.130039}, {0.195059, -0.0910273}}
```

```
..# Producto escalar entre matrices
```

```
In[ ]:= Matrix8.Matrix10 // MatrixForm
      |forma de matriz
```

```
Out[ ]//MatrixForm=
      ( 1. -2.77556 × 10-17 )
      (-3.46494 × 10-17 1.)
```

```
..# Matriz Identidad 3x3
```

```
In[ ]:= IdentityMatrix[3]
      |matriz identidad
```

```
Out[ ]:=
      {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

```
..# Matriz Identidad 5x5
```

```
In[ ]:= IdentityMatrix[5] // MatrixForm
      |matriz identidad |forma de matriz
```

```
Out[ ]//MatrixForm=
      ( 1 0 0 0 0 )
      ( 0 1 0 0 0 )
      ( 0 0 1 0 0 )
      ( 0 0 0 1 0 )
      ( 0 0 0 0 1 )
```

```
# Diagonal principal con {1,2,3}
```

```
In[ ]:= DiagonalMatrix[{1, 2, 3}]
      |matriz diagonal
```

```
Out[ ]:=
      {{1, 0, 0}, {0, 2, 0}, {0, 0, 3}}
```

```
# Diagonal principal con {1,2,3,4,5}
```

```
In[ ]:= DiagonalMatrix[{1, 2, 3, 4, 5}] // MatrixForm
      |matriz diagonal |forma de matriz
```

```
Out[ ]//MatrixForm=
      ( 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 )
```

```
# Genera una matriz de 3 filas y 3 columnas, donde cada elemento es un número aleatorio entre 0 y 1.
```

```
In[ ]:= RandomReal[{0, 1}, {3, 3}] // MatrixForm
|real aleatorio |forma de matriz
```

```
Out[ ]:= MatrixForm=

$$\begin{pmatrix} 0.000360236 & 0.347271 & 0.195604 \\ 0.496464 & 0.260868 & 0.791208 \\ 0.199936 & 0.0627008 & 0.979587 \end{pmatrix}$$

```

# Matriz 2x2

```
In[ ]:= A = {{1, 2}, {3, 4}}
```

```
Out[ ]:=
{{1, 2}, {3, 4}}
```

# Tamaño de la matriz nxn

```
In[ ]:= Dimensions[A]
|dimensiones
```

```
Out[ ]:=
{2, 2}
```

# Traza, suma de los elementos de la diagonal principal

```
In[ ]:= Tr[A]
|traza
```

```
Out[ ]:=
5
```

# Determinante de una matriz

```
In[ ]:= Det[A]
|determinante
```

```
Out[ ]:=
-2
```

# calcula la norma Frobenius, que es la raíz cuadrada de la suma de los cuadrados de todos sus elementos

```
In[ ]:= Norm[A]
|norma
```

```
Out[ ]:=
 $\sqrt{15 + \sqrt{221}}$ 
```

Potencia de una matriz cuadrada

```
In[ ]:= A^n
```

```
Out[ ]:=
{{1, 2^n}, {3^n, 4^n}}
```

```
# Calcula los valores propios de la matriz A
```

```
In[ ]:= Eigenvalues[A]
         |autovalores
```

```
Out[ ]:=

$$\left\{ \frac{1}{2} (5 + \sqrt{33}), \frac{1}{2} (5 - \sqrt{33}) \right\}$$

```

```
# Calcula los vectores propios correspondientes a cada uno de los valores propios
```

```
In[ ]:= Eigenvectors[A]
         |autovectores
```

```
Out[ ]:=

$$\left\{ \left\{ \frac{1}{6} (-3 + \sqrt{33}), 1 \right\}, \left\{ \frac{1}{6} (-3 - \sqrt{33}), 1 \right\} \right\}$$

```

```
# Extrae la columna 1
```

```
In[ ]:= A[[All, 1]]
         |todo
```

```
Out[ ]:=
{1, 3}
```

```
# Extrae la fila 2
```

```
In[ ]:= A[[2, All]]
         |todo
```

```
Out[ ]:=
{3, 4}
```

```
@ Reemplaza el valor dentro de la matriz en posición específica
```

```
In[ ]:= A = ReplacePart[A, {1, 1} → 10]
         |sustituye una parte
```

```
Out[ ]:=
{{10, 2}, {3, 4}}
```

```
# Reemplaza todo los valores que coinciden por el nuevo valor, no necesita una posición específica
```

```
In[ ]:= A = A /. 10 → -10
```

```
Out[ ]:=
{{-10, 2}, {3, 4}}
```

💎 **Nota:** Se debe volver a definir la matriz original para que los valores puedan ser reemplaza ya sea por subíndices o por números.

```
# Definimos vector
```

```
In[*]:= v = {1, 2}
```

```
Out[*]=  
{1, 2}
```

```
# Producto escalar
```

```
In[*]:= A . v
```

```
Out[*]=  
{-6, 11}
```

```
# Producto escalar con Dot[]
```

```
In[*]:= A = {{1, 2}, {3, 4}, {5, 6}};
```

```
In[*]:= Dot[A, v] // MatrixForm  
|producto escalar |forma de matriz
```

```
Out[*]//MatrixForm=  

$$\begin{pmatrix} 5 \\ 11 \\ 17 \end{pmatrix}$$

```

```
In[*]:= Diagonal[A]  
|diagonal
```

```
Out[*]=  
{1, 4}
```

```
# Definimos vectores
```

```
In[*]:= v1 = {1, 2, 3}  
v2 = {4, 5, 6}
```

```
Out[*]=  
{1, 2, 3}
```

```
Out[*]=  
{4, 5, 6}
```

```
# Producto Cruz entre vectores
```

```
In[*]:= Cross[v1, v2]  
|producto vectorial
```

```
Out[*]=  
{-3, 6, -3}
```

## 10.5 Programación básica en Mathematica

✎ **Explicación:** El *input* en Mathematica puede ser tan simple como escribir una expresión matemática o invocar funciones integradas. Por ejemplo, si deseas calcular el valor de una expresión como  $3 + 5$ , simplemente escribes  $3 + 5$  y presionas

*Shift + Enter* para ejecutar. Para definir una variable o función, se usa la sintaxis de asignación, como  $x = 10$ . Además, el sistema de control de flujo permite crear programas más complejos con condicionales (If), bucles (For), y definiciones de funciones personalizadas. Las funciones básicas de entrada y salida son esenciales para empezar a construir programas más elaborados, mientras que la interactividad de Mathematica facilita la experimentación y el aprendizaje.

*# Ingreso de datos por el usuario*

```
In[*]:= numero = Input["Ingresa el número"]
      |entra
```

```
Out[*]:=
10
```

*# Programa para calcular el factorial, ingresamos un negativo*

```
In[*]:= numero = Input["Ingresa el número del cual quieres el factorial"];
      |entra
fact = numero!;
Print["El factorial de ", numero, " es ", fact];
      |escribe
```

El factorial de -6 es ComplexInfinity

*# Programa para calcular el factorial, con entrada válida*

```
In[*]:= numero = Input["Ingresa el número del cual quieres el factorial"]
      |entra
fact = numero!;
Print["El factorial de ", numero, " es ", fact];
      |escribe
Speak["El factorial de " <> ToString[numero] <> " es " <> ToString[fact]]
      |pronuncia          |convierte a cadena de caracteres    |convierte a cadena de
```

```
Out[*]:=
```

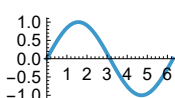
5

El factorial de 5 es 120

*# Programa para graficar la función Seno de 0 has n, ingresado por el usuario*

```
In[*]:= funcion = Input["Ingresa la funcion que dependa de x"];
      |entra
limite = Input["hasta que x"];
      |entra
plotfuncion = Plot[funcion, {x, 0, limite}];
      |representación gráfica
Print["La grafica de ", funcion, " es ", plotfuncion];
      |escribe
```

La grafica de Sin[x] es

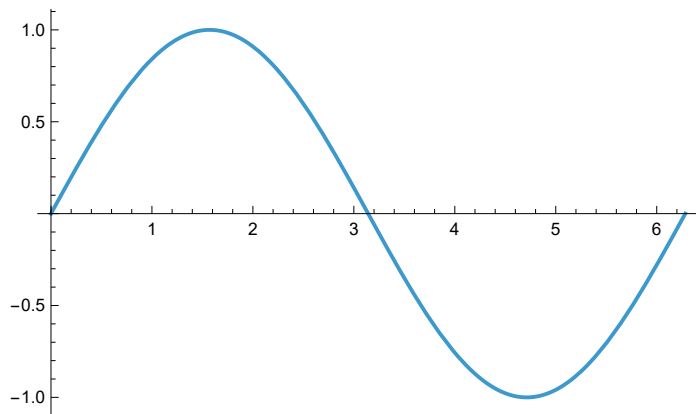




# Función para que la gráfica se formatee y sea mas grande

In[ ]:= plotfuncion

Out[ ]:=

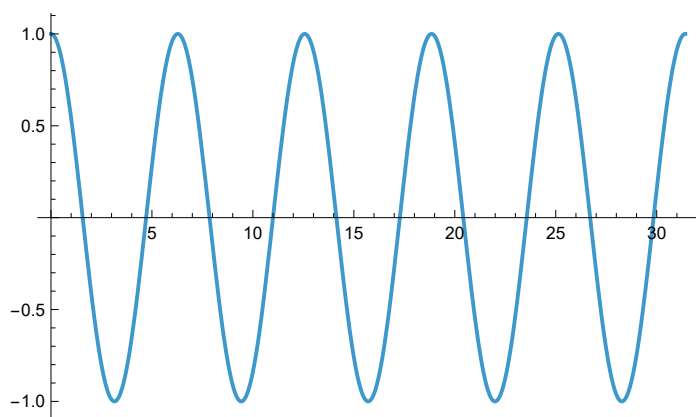


💎 **Nota:** Se usa `plotfuncion` para poder agrandar la imagen de salida en caso se tenga inconvenientes.

# Programa para graficar la función Coseno de  $\theta$  has  $n$ , ingresado por el usuario

```
In[ ]:= funcion = Input["Ingresa la funcion que dependa de x"];
          |entra
limite = Input["hasta que x"];
          |entra
plotfuncion = Plot[funcion, {x, 0, limite}];
          |representación gráfica
Speak["Aquí está la gráfica de la función que ingresaste"];
          |pronuncia
plotfuncion
```

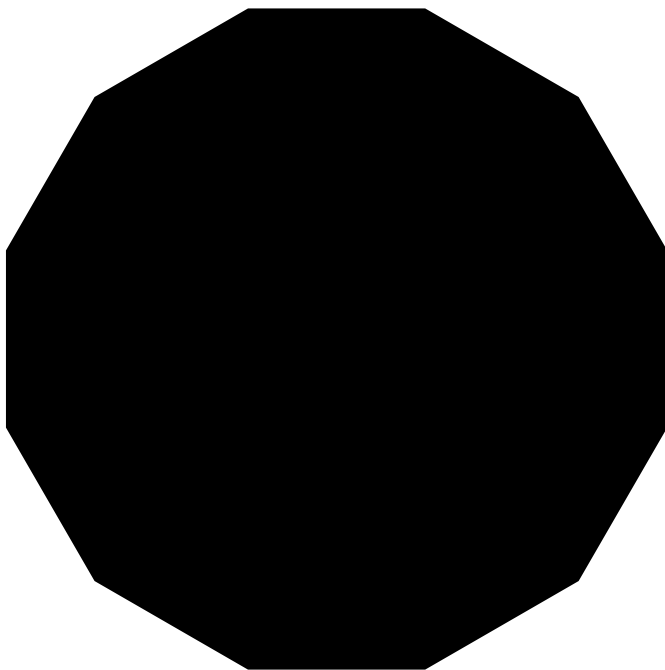
Out[ ]:=



# Programa para generar una figura con  $n$  lados

```
In[*]:= n = Input["¿Cuántos lados quieres que tenga el polígono?"];  
          |entra  
Graphics[RegularPolygon[n]]  
          |gráfico |polígono regular  
Speak["He dibujado un polígono de " <> ToString[n] <> " lados."  
      |pronuncia |convierte a cadena de caracteres
```

Out[\*]=



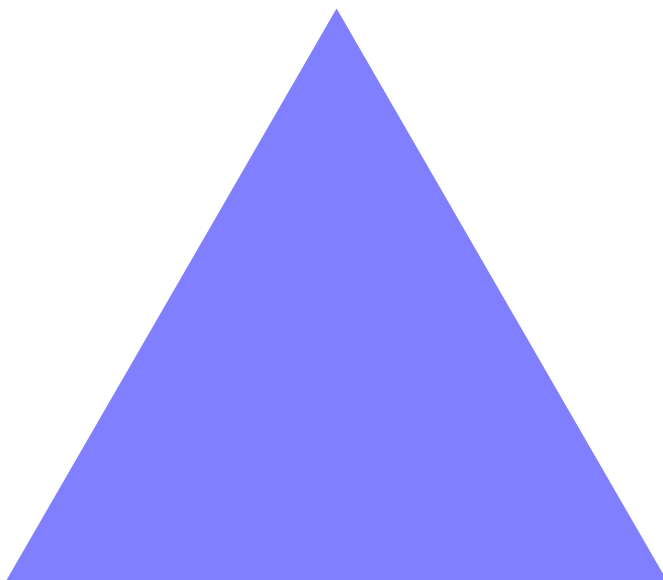
```
# Programa para generar una figura con n lados, con formato de color
```

```

In[ ]:= n = Input["¿Cuántos lados quieres que tenga el polígono?"];
Graphics[Style[RegularPolygon[n], Directive[Opacity[0.5], Blue]]]
Speak["He dibujado un polígono de " <> ToString[n] <> " lados."]

```

Out[ ]:=



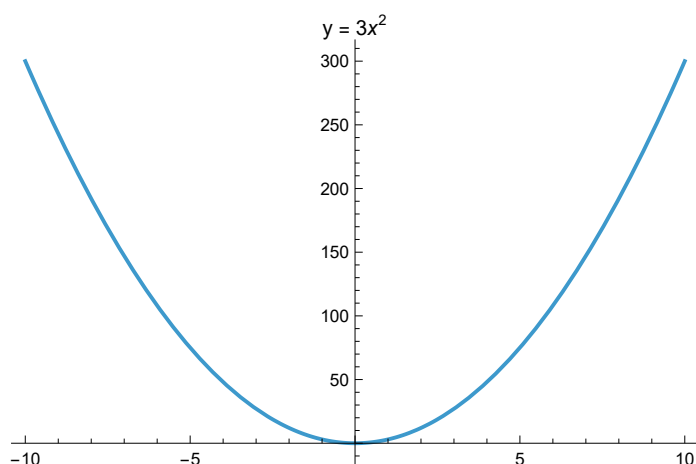
# Pedir al usuario un valor para el coeficiente  $a$  en una ecuación cuadrática y luego graficar

```

In[ ]:= a = Input["Ingresa el coeficiente a para y = ax^2:"];
Plot[a x^2, {x, -10, 10}, PlotLabel -> "y = " <> ToString[a] <> "x^2"]

```

Out[ ]:=



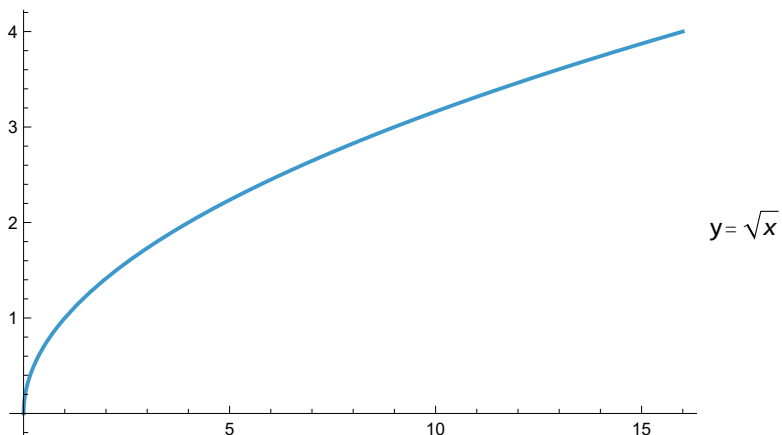
💎 **Nota:** `ToString[n]` se usa cuando quieres explícitamente convertir el valor de la variable  $n$  en texto para concatenarlo con otros textos. Es útil si estás construyendo una cadena de caracteres.

# Pedir al usuario un valor para el coeficiente  $n$  en una raíz y luego graficar

```
In[ ]:= n = Input["Ingrese un número para calcular su raíz cuadrada: "];
Print["La raíz cuadrada de ", n, " es ", Sqrt[n]];
Plot[Sqrt[x], {x, 0, n}, PlotLegends -> "y=√x"]
```

La raíz cuadrada de 16 es 4

Out[ ]:=



💎 **Nota:** ,  $n$  se usa en funciones como *Print* para pasar valores directamente, y *Mathematica* se encarga de convertir esos valores a su representación de texto en la salida automáticamente.

#### ■ 🌀 **Reto en clase — Tabla de multiplicar**

Hacer que el usuario ingrese un usuario y se imprima la tabla de multiplicar hasta el 12, ejemplo (2, 4 , ..., 24).

```
In[ ]:= n = Input["Ingrese el número de la tabla: "];
        |entra
Print["Tabla del ", n];
        |escribe
Column[Table[Times[n, i], {i, 1, 12}]]
        |columna |tabla |multiplicación
```

Tabla del 5

Out[ ]:=

```
5
10
15
20
25
30
35
40
45
50
55
60
```

### ■ **Reto en clase — Matrices**

*Hacer una matriz 2x2 ingresada por el usuario: sacar el determinante y generar un botón interactivo para mostrar la respuesta.*

# Ingreso de elementos de la matriz por el usuario

```
In[ ]:= a11 = ToExpression[InputString["Ingrese el valor de a11: "]];
        |convierte en ex... |cadena de caracteres de entrada
a12 = ToExpression[InputString["Ingrese el valor de a12: "]];
        |convierte en ex... |cadena de caracteres de entrada
a21 = ToExpression[InputString["Ingrese el valor de a21: "]];
        |convierte en ex... |cadena de caracteres de entrada
a22 = ToExpression[InputString["Ingrese el valor de a22: "]];
        |convierte en ex... |cadena de caracteres de entrada
matriz = {{a11, a12}, {a21, a22}};
determinante = Det[matriz];
              |determinante
Button["Mostrar Determinante", Print["El determinante es: ", determinante]]
        |botón |escribe
```

Out[ ]:=

Mostrar Determinante

El determinante es: -2

# Ingreso de elementos de la matriz por el usuario

```
In[ ]:= matrix = Input["Ingrese la matriz (por ejemplo, {{1, 2}, {3, 4}}): "];
          |entra
determinante1 = Det[matrix];
              |determinante
Button["Mostrar Determinante", Print["El determinante es: ", determinante1]]
      |botón                                     |escribe
```

Out[ ]:=

Mostrar Determinante

El determinante es: -7

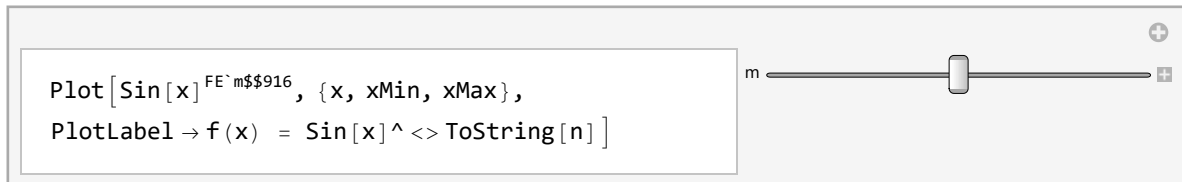
### ■ **Reto en clase — Gráfica interactiva**

*Graficar una función en  $\text{Sin}[x]^n$ , donde  $n$  sea ingresada por el usuario y además  $f(x)$  se manipule de 0 hasta  $n$ .*

# Función Seno, el usuario ingresa máximo i mínimo de la función así como el  $n$

```
In[ ]:= xMin = Input["Ingrese el límite inferior de x (xMin): "];
          |entra
xMax = Input["Ingrese el límite superior de x (xMax): "];
        |entra
n = Input["Para Sin[x]^n. Ingrese [n]: "];
    |entra      |seno
Manipulate[Plot[Sin[x]^m, {x, xMin, xMax},
               |manipula      |repr... |seno
               PlotLabel -> "f(x) = Sin[x]^<> ToString[n] ", {m, 0, n, 1}]
          |etiqueta de representación |seno      |convierte a cadena de caracteres
```

Out[ ]:=



Plot::p1ln: Limiting value xMin in {x, xMin, xMax}  
is not a machine-sized real number.

Plot::p1ln: Limiting value xMin in {x, xMin, xMax}  
is not a machine-sized real number.

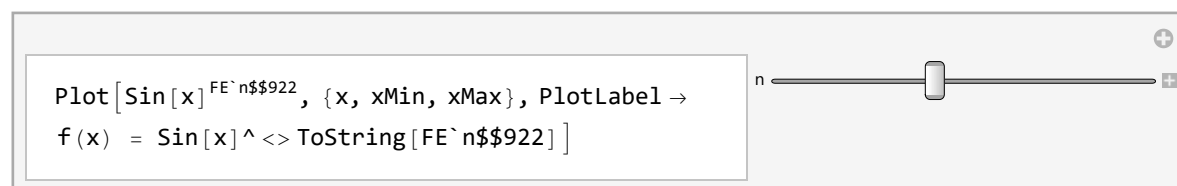
# Función Seno, el usuario ingresa máximo i mínimo de  $n$

```

In[*]:= xMin = Input["Ingrese el límite inferior de x (xMin): "];
          |entra
xMax = Input["Ingrese el límite superior de x (xMax): "];
          |entra
n = Input["Para Sin[x]^n. Ingrese [n]: "];
          |entra      |seno
Manipulate[Plot[Sin[x]^n, {x, xMin, xMax},
               |manipula |repr... |seno
             PlotLabel -> "f(x) = Sin[x]^" <> ToString[n]], {n, xMin, xMax, 1}]
          |etiqueta de representación |seno      |convierte a cadena de caracteres

```

Out[\*]=



```

Plot::p1ln: Limiting value xMin in {x, xMin, xMax}
is not a machine-sized real number.
Plot::p1ln: Limiting value xMin in {x, xMin, xMax}
is not a machine-sized real number.
Plot::p1ln: Limiting value xMin in {x, xMin, xMax}
is not a machine-sized real number.
Plot::p1ln: Limiting value xMin in {x, xMin, xMax}
is not a machine-sized real number.

```

- **🔗 Reto en clase — Números Complejos**  
Hacer que el usuario ingrese un número complejo y devuelva su conjugado.

# Programa para calcular el conjugado de un número ingresado por el usuario

```

complejo = Input["Ingrese un número complejo (por ejemplo, 3 + 4 I): "];
            |entra                                     |número i
conjugado = Conjugate[complejo];
            |conjugado
Print["El conjugado de ", complejo, " es ", conjugado]
|escribe

```

El conjugado de  $-2 + 7i$  es  $-2 - 7i$

- **🔗 Reto en clase — Gráfica de tabla**  
Hacer un programa donde el usuario ingrese una lista y de como resultado la gráfica de la lista ingresada.

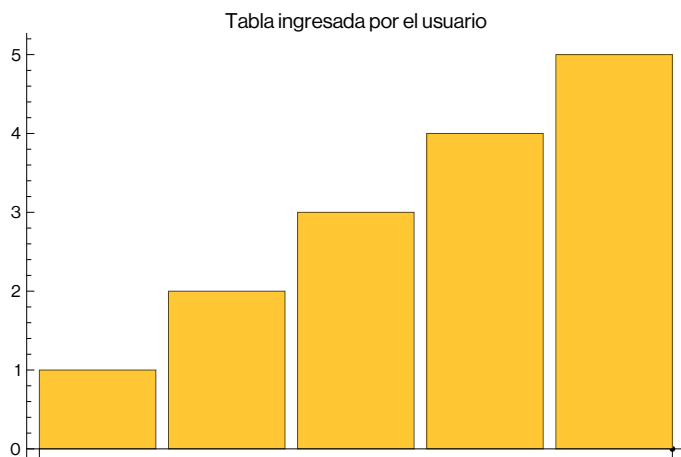
# Ingreso de tabla por el usuario para graficar barras

```

In[ ]:= numeros = Input["Ingrese una lista de números (por ejemplo, {1, 2, 3, 4}): "];
      |entra
      BarChart[numeros, PlotLabel -> HoldForm[Tabla ingresada por el usuario],
      |diagrama de barras |etiqueta de r... |forma sin evaluación
      LabelStyle -> {FontFamily -> "Neue Haas Grotesk Display Pro"}]
      |estilo de etiqueta |familia de tipo de letra |muestra

```

Out[ ]:=





---

## Tareas

⚡ **Instrucciones:** En esta sección se agrupan las tareas asignadas.

### Tarea 8 — Reto Matrices

📅 2025/10/15

Se solicita subir los ejercicios de clase, asegurándose de que cumplan con el formato indicado, incluyendo color de fondo, comentarios y títulos .

***Ver Clase 8 (Retos en clase)***