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

Tareas

Instrucciones: En esta sección se agrupan las tareas asignadas.

Tarea 9 – Aplicación del condicional if

2025/10/20

1. **Definir una lista de 10 números. Usando el comando *if*, decir al usuario si el número 6 está en la lista.**
   - **Si está que diga la frase "El número está contenido en la lista".**
   - **Si no está, que lo agregue a la lista.**

```
# Si {6} no se encuentra en la lista:
```

In[•]:=
```
lista = Input["Ingrese una lista de 10
        entra
      números (por ejemplo: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}): "];
If[MemberQ[lista, 6],
si  ¿contenido en?
  Print["El número está contenido en la lista"], (lista = Join[lista, {6}];
  escribe                                                junta
   Print["El número no estaba en la lista, se ha agregado."])];
   escribe
Print["Lista actual: ", lista];
escribe
```

```
El número no estaba en la lista, se ha agregado.

Lista actual: {1, 7, 9, 3, 4, 5, 1, 11, 113, 20, 6}
```

```
# Si {6} se encuentra en la lista:
```

In[•]:=
```
lista = Input["Ingrese una lista de 10
        entra
      números (por ejemplo: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}): "];
If[MemberQ[lista, 6],
si  ¿contenido en?
  Print["El número está contenido en la lista"], (lista = Join[lista, {6}];
  escribe                                                junta
   Print["El número no estaba en la lista, se ha agregado."])];
   escribe
Print["Lista actual: ", lista];
escribe
```

```
El número está contenido en la lista

Lista actual: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

2. **Definir un número complejo $z = a + bi$**
   - **Si el módulo de este número es menor que 4, que diga la oración "el módulo es: ".**
   - **Si es mayor que 4, que diga "No hago nada".**

```
# Si el módulo es menor que 4
```

In[•]:=
```
z = Input["Ingrese un número complejo (por ejemplo: 2 + 3 I): "];
    entra                                                  número i
If[Abs[z] < 4, Print["El módulo es: ", Abs[z]], Print["No hago nada"]]
si  valor absoluto  escribe                  valor abs···  escribe
```

El módulo es: $\sqrt{13}$

```
# Si el módulo es mayor que 4
```

In[•]:=
```
z = Input["Ingrese un número complejo (por ejemplo: 2 + 3 I): "];
    entra                                                  número i
If[Abs[z] < 4, Print["El módulo es: ", Abs[z]], Print["No hago nada"]]
si  valor absoluto  escribe                  valor abs···  escribe
```

No hago nada

3. Definir la lista de números aleatorios.

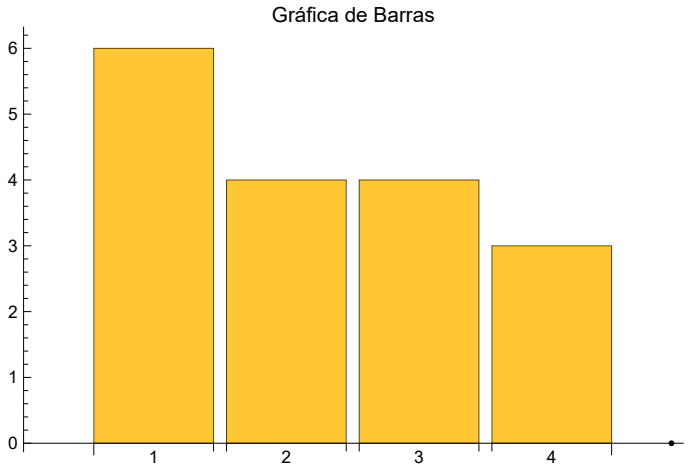
- **Si la longitud de esta lista es menor de 5, hacer una gráfica de barras.**
- **Si la longitud es mayor que 5, hacer una gráfica circular.**

```
# Si la longitud de la lista menor a 5
```

In[•]:=
```
Clear[lista];
borra
lista = RandomInteger[{1, 10}, RandomInteger[{2, 10}]];
        entero aleatorio       entero aleatorio
Print["Lista generada: ", lista];
escribe
If[Length[lista] < 5,
si  longitud
 BarChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica de Barras"],
 diagrama de barras  etiquetas de dia···  rango  longitud  etiqueta de representación
 PieChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica Circular"]]
 diagrama circular  etiquetas de dia···  rango  longitud  etiqueta de representación
```

```
Lista generada: {6, 4, 4, 3}
```

Out[•]=

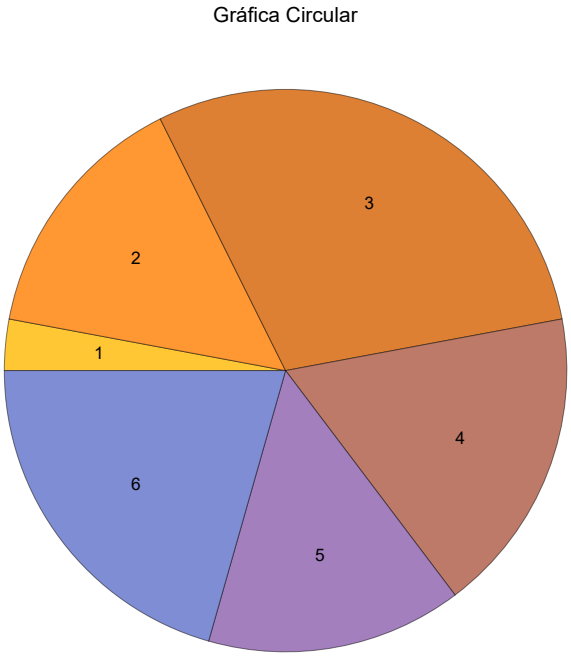



```
# Si la longitud de la lista mayor a 5
```

In[•]:=
```
Clear[lista];
borra
lista = RandomInteger[{1, 10}, RandomInteger[{2, 10}]];
        entero aleatorio        entero aleatorio
Print["Lista generada: ", lista];
escribe
If[Length[lista] < 5,
si  longitud
 BarChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica de Barras"],
 diagrama de barras  etiquetas de dia···  rango  longitud  etiqueta de representación
 PieChart[lista, ChartLabels → Range[Length[lista]], PlotLabel → "Gráfica Circular"]]
 diagrama circular  etiquetas de dia···  rango  longitud  etiqueta de representación
```

```
Lista generada: {1, 5, 10, 6, 5, 7}
```

Out[•]=



4. Definir r_1, si:

- **Si es número, que lo divida entre 2.**
- **Si no que, expanda la expresión.**

```
# Ingreso la expresión 2/5, se divide por 2
```

```
r1 = Input["Ingrese un número o una expresión algebraica: "];
     entra
If[NumericQ[r1], Print["El resultado de dividir entre 2 es: ", r1 / 2],
si  ¿expresión numé··  escribe
   Print["La expresión expandida es: ", Expand[FullSimplify[r1]]]];
   escribe                              expand··  simplifica completamente
```

```
El resultado de dividir entre 2 es: 1/5
```

```
# Ingreso la expresión Cos[x]^2 + Sin[x]^2 se espera que la salida sea 1
```

```
# Se usa EXPAND y fullsimplify, porque no todas las expresiones que ingrese el usuario podrán expandirse
```

In[•]:=
```
Clear[r1];
borra
r1 = Input["Ingrese un número o una expresión algebraica: "];
     entra

If[NumericQ[r1],
si  ¿expresión numérica?
  Print["El resultado de dividir entre 2 es: ", r1 / 2],
  escribe
  Print["La expresión expandida es: ", Expand[FullSimplify[r1]]]];
  escribe                              expand··  simplifica completamente
```

```
La expresión expandida es: 1
```

5. Defina la función para la delta de Dirac:

$$\delta(x-c) = \begin{cases} \infty & \text{si } x = c \\ 0 & \text{si } x \neq c \end{cases}$$

Grafique la expresión obtenida de -10 a 10.

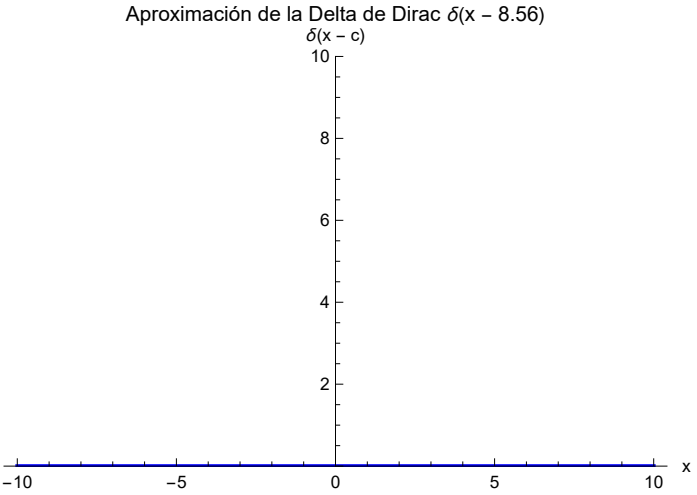
```
# Cumple con lo pedido por definición cuando x=c -> ∞;
  si no cumple, x != c -> 0
```

In[•]:=
```
c = Input["Ingrese el valor de c para la delta de Dirac δ(x - c): "];
    entra

delta[x_] := If[x == c, ∞, 0];
             si

Plot[delta[x], {x, -10, 10}, PlotRange → {0, 10},
representación gráfica       rango de representación
 PlotLabel → "Aproximación de la Delta de Dirac δ(x - " <> ToString[c] <> ")",
 etiqueta de representación                                convierte a cadena de cara
 AxesLabel → {"x", "δ(x - c)"}, PlotStyle → Blue, Exclusions → None]
 etiqueta de ejes                 estilo de repr···  azul  exclusiones  ninguno
```

Out[•]=




```
# Delta de Dirac como función gaussiana, para observar una campana
```

```
# Un valor pequeño para epsilon hace que el pico sea más estrecho y alto.
```

- **Para que pueda visualizarse la gráfica, usamos la delta de Dirac como función gaussiana:** $\delta_\epsilon(x-c) = \frac{1}{\sqrt{2\pi\epsilon}}\, e^{-\frac{(x-c)^2}{2\epsilon}}$

In[•]:=

```
c = Input["Ingrese el valor de c para la delta de Dirac (centro): "];
epsilon = Input["Ingrese el valor de epsilon (ejemplo: 0.01): "];
delta[x_] := (1 / Sqrt[2 Pi epsilon]) Exp[-(x - c)^2 / (2 epsilon)];

Plot[delta[x], {x, -10, 10}, PlotRange → All,
 PlotLabel → "Aproximación Gaussiana de la Delta de Dirac en x = " <> ToString[c],
 AxesLabel → {"x", "δ(x - c)"}, PlotStyle → Thick]
```

Out[•]=