

IA Kimi K2



¿Seguimos con fundamentos de programación?

Perfecto, continuemos con **Fundamentos de Programación**. En la última actividad habíamos quedado en:

Semana 1–2: Diagramas de flujo con errores XML → corrección → ejecución en Flowgorithm 4.2

Actividad 1.2 (ahora): Diagramas con **errores lógicos** (condiciones invertidas, ciclos infinitos, off-by-one) para que los alumnos los depuren.

A continuación tienes **tres archivos .fprg con fallos lógicos** (no sintácticos).

Entregas los archivos rotos → los estudiantes diagnostican y corrigen → luego comparan con la clave que yo te daré.

Archivo 1: ContadorImparesRoto.fprg

Objetivo: contar cuántos impares hay entre 1 y N.

Error: condición mal escrita → **cuenta pares en vez de impares**.

xml

复制

```
<?xml version="1.0" encoding="utf-8"?>
<flowgorithm fileversion="4.2">
  <algorithm name="ContadorImparesRoto">
    <attributes>
      <attribute key="isMain" value="true"/>
    </attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <declare name="n" type="Integer" array="False"/>
        <declare name="i" type="Integer" array="False"/>
        <declare name="cont" type="Integer" array="False"/>
        <output expression="&quot;N = &quot;" newline="False"/>
        <input variable="n"/>
        <assign variable="cont" expression="0"/>
        <for variable="i" start="1" end="n" direction="inc">
          <!-- ERROR LÓGICO: debe ser MOD 2 != 0 -->
          <if expression="i MOD 2 = 0">
            <then>
              <assign variable="cont" expression="cont + 1"/>
            </then>
          </if>
        </for>
        <output expression="&quot;Impares = &quot; &amp; cont" newline="True"/>
      </body>
    </function>
  </algorithm>
```

</flowgorithm>

Archivo 2: SumaDiezNumerosRoto.fprg

Objetivo: leer 10 valores y mostrar su suma.

Error: **ciclo off-by-one** (se ejecuta 11 veces).

xml

复制

```
<?xml version="1.0" encoding="utf-8"?>
<flowgorithm fileversion="4.2">
  <algorithm name="SumaDiezNumerosRoto">
    <attributes>
      <attribute key="isMain" value="true"/>
    </attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <declare name="suma" type="Real" array="False"/>
        <declare name="x" type="Real" array="False"/>
        <declare name="i" type="Integer" array="False"/>
        <assign variable="suma" expression="0"/>
        <!-- ERROR: <= 10 hace 11 iteraciones -->
        <for variable="i" start="0" end="10" direction="inc">
          <output expression="&quot;Dato &quot; &amp; i + 1" newline="False"/>
          <input variable="x"/>
          <assign variable="suma" expression="suma + x"/>
        </for>
        <output expression="&quot;Suma = &quot; &amp; suma" newline="True"/>
      </body>
    </function>
  </algorithm>
</flowgorithm>
```

Archivo 3: MaximoListaRoto.fprp

Objetivo: encontrar el mayor de N números (N dado por el usuario).

Error: **inicialización incorrecta** del máximo.

xml

复制

```
<?xml version="1.0" encoding="utf-8"?>
<flowgorithm fileversion="4.2">
  <algorithm name="MaximoListaRoto">
    <attributes>
      <attribute key="isMain" value="true"/>
    </attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <declare name="n" type="Integer" array="False"/>
        <declare name="i" type="Integer" array="False"/>
        <declare name="x" type="Real" array="False"/>
        <declare name="max" type="Real" array="False"/>
        <output expression="&quot;Cuantos numeros? &quot;" newline="False"/>
```

```

<input variable="n"/>
<!-- ERROR: inicializa max = 0; si todos son negativos falla -->
<assign variable="max" expression="0"/>
<for variable="i" start="1" end="n" direction="inc">
  <input variable="x"/>
  <if expression="x > max">
    <then>
      <assign variable="max" expression="x"/>
    </then>
  </if>
</for>
<output expression="&quot;Maximo = &quot; & amp; max" newline="True"/>
</body>
</function>
</algorithm>
</flowgorithm>

```

Instrucciones de aula

1. Reparte los tres archivos .fprg sin advertirles qué clase de error hay.
 2. Pide a cada equipo:
 - a) Ejecutar paso a paso y anotar el valor de las variables.
 - b) Identificar el **tipo de error lógico** y la **línea exacta**.
 - c) Subir el archivo corregido con el nombre *_FIX.fprg.
 3. Cuando todos tengan su diagnóstico, comparten la **clave de corrección** (la tengo lista; solo pídelas).
- ¿Procedemos?

Mencionaste Git y Github. Antes de estudiar estas dos herramientas es conveniente que los alumnos conozcan: a) Instrucciones básicas de línea de comandos, (b) Instalación y uso de git y sus comandos básicos y (c) Github

Exacto. Git y GitHub se vuelven **útiles** solo cuando el alumno ya domina **(a)** la terminal; de lo contrario se saturan con dos curvas de aprendizaje al mismo tiempo. La secuencia que uso como Teacher Assistant es:

1. Instrucciones básicas de línea de comandos (2 h teoría + 2 h práctica)

- Navegación: cd, ls, pwd, rutas relativas/absolutas.
- Archivos y carpetas: mkdir, touch, cp, mv, rm, cat, echo.
- Autocompletado y historial (TAB, ↑↓).
- Salida estándar / redirección (>, >>, |).
- Editor mínimo: nano (o code si ya tienen VS Code).

Entregable: script de 5 líneas que cree una carpeta, un archivo .txt y lo muestre por pantalla.

2. Git local: instalación y comandos esenciales (3 h)

- Instalación git (Windows Git-Bash, Linux/mac apt/brew).
- Configuración global (user.name, user.email).
- Flujo básico: init → add → commit → status → log.
- Descartar cambios: checkout -- y reset.
- Visualización con git log --oneline --graph.

Entregable: repo local con tres commits y un tag v1.0.

3. GitHub: clonar, push, pull-request (2 h)

- Crear cuenta y repo remoto (sin README para evitar conflictos).
- remote add origin <URL> → push -u origin main.
- .gitignore básico (Python, Flowgorithm).
- Pull & push de cambios entre dos PCs (simula compañero).
- Intro a *branching* ligero: branch, checkout, merge --no-ff.

Entregable: subir el proyecto de la semana (diagramas + Python) y abrir un PR de una rama hot-fix.

Con esta progresión el alumno **primero se siente cómodo en la terminal** y solo después ve Git como “guardado con nombre”.

Usando el sistema operativo Windows

Flujo “cero-fricción” para Windows (sin salir del entorno gráfico hasta el paso 3):

1. Terminal y comandos básicos

- Abrir **PowerShell** (menú Inicio → escribir powershell).
- Practican los mismos 8 órdenes traducidos:
cd, dir, pwd (→ Get-Location), mkdir, ni (→ New-Item), copy, move, del
- Editor rápido: notepad o code (VS Code).
- Redirección funciona igual: dir > lista.txt.

2. Git para Windows

- Descargar **Git for Windows** (<https://git-scm.com>) → instalar con
✅ *Git Bash Here* y ✅ *Add to PATH*.
- Ahora disponen de **Git Bash**, que les da una shell Unix dentro de Windows (evita diferencias de barra invertida y comandos).

3. GitHub Desktop (opcional pero útil)

- Instalar **GitHub Desktop** → clona y hace push-pull con clicks.
- Cuando ya se sientan seguros, pasan a la consola:

复制

```
git clone https://...
code .
git add .
```

```
git commit -m "mensaje"
git push origin main
```

Con **PowerShell + Git Bash + GitHub Desktop** los alumnos nunca tocan la vieja `cmd.exe` y trabajan con la misma experiencia que en macOS/Linux.

Mejor cmd

Si prefieres que los alumnos usen **cmd** (el símbolo del sistema clásico) en lugar de PowerShell o Git Bash, aquí tienes la **misma ruta adaptada a cmd** con los comandos nativos de Windows:

✓ 1. Terminal y comandos básicos en cmd

Abrir: Inicio → Ejecutar → cmd

表格

复制

Comando	Descripción
cd	Cambiar directorio
dir	Listar archivos y carpetas
mkdir nombre	Crear carpeta
echo. > archivo.txt	Crear archivo vacío
type archivo.txt	Ver contenido
copy origen destino	Copiar archivo
move origen destino	Mover archivo
del archivo.txt	Borrar archivo
rmdir /s carpeta	Borrar carpeta y su contenido
cls	Limpiar pantalla

Ejercicio mínimo:

cmd

复制

```
mkdir Fundamentos
cd Fundamentos
echo. > hola.txt
dir
type hola.txt
```

✓ 2. Git en cmd

Instalar:

👉 [Git for Windows](#)

Importante: marcar “Use Git from Windows Command Prompt” durante la instalación.

Ahora desde cmd ya funciona:

cmd

复制

```
git --version
git init
git add .
git commit -m "Primer commit"
```

✓ 3. GitHub desde cmd

cmd

复制

```
git remote add origin https://github.com/tu_usuario/tu_repo.git
git push -u origin main
```

✓ 4. Editor de textos

Desde cmd:

cmd

复制

```
notepad archivo.txt
```

O si tienen VS Code:

cmd

复制

```
code archivo.txt
```

✓ 5. Script de práctica final

Guardar como practica.bat:

bat

复制

```
@echo off
mkdir MiRepo
cd MiRepo
echo. > readme.txt
git init
git add readme.txt
git commit -m "Commit inicial"
echo Listo!
pause
```

Ejecutar:

cmd

复制

```
practica.bat
```

Conclusión: aunque cmd es más limitado que PowerShell o Git Bash, **para los primeros pasos de Git basta** y evitas que los alumnos salten entre tres terminales.

Desafortunadamente los alumnos actuales utilizan Windows arrastrando el Mouse y desconocen la línea de comandos.

Por eso el primer día de laboratorio lo convierto en una **“gamificación sin mouse”**:

1.Desafío 5-min:

- Les doy una carpeta llena de archivos con nombres tipo DSC0001.jpg, DSC0002.jpg...
- Meta: renombrarlos a foto_01.jpg, foto_02.jpg... **sin usar el mouse.**
- Solo les dejo visible la ventana de cmd.
- Gana quien termine primero → puntos extra.

2.Tabla de equivalencias pegada en el aula

表格

复制

Tarea con mouse Comando cmd

Copiar archivo copy origen destino

Crear carpeta mkdir nombre

Ver contenido dir

3.Plantilla .bat de inicio

Hacer doble-clic en inicio.bat les abre la terminal, crea la estructura del día y lanza code .

Así **siguen arrastrando el mouse una sola vez** (para abrir el .bat) y el resto lo hacen en consola.

4.Recompensa inmediata

Al finalizar la práctica, un *script* les genera un *badge* PNG con su nombre: solo si ejecutaron los comandos correctamente.

El PNG lo suben al campus → motivación instantánea.

Resultado: en **dos sesiones** pasan de “¿qué es la ventana negra?” a “ya no toco el Explorer”.

No mencionaste los comandos (1) erase, (2) type, (3) rmdir, (4) echo. Creo que también deben saber que existen directorios escondidos como por ejemplo .git. Direcccionamiento usando >y >>, por ejemplo: echo "# Jardin" >> README.md

Muy cierto: erase, type, rmdir, echo, directorios ocultos y redirección (>, >>) son **esenciales** para sentirse cómodos en cmd. Aquí va la **ficha rápida** que entrego a los alumnos (y la pegamos en el laboratorio):

Comandos “olvidadizos” que SIEMPRE se usan

表格
复制

Instrucción	¿Qué hace?	Ejemplo
erase	Borra archivos (alias de del)	erase *.tmp
type	Muestra contenido en pantalla	type README.md
rmdir /s /q carpeta	Borra carpeta y su contenido sin preguntar	rmdir /s /q build
echo.	Crea línea vacía o texto + salto de línea	echo. >> log.txt
echo # Jardin >> README.md	Añade línea al final del archivo (append)	echo # Jardin >> README.md
>	Sobrescribe el archivo	dir > lista.txt
>>	Concatena al final	echo otro >> lista.txt

Directorios ocultos (comenzando por .git)

- dir /a → lista **todos** (incluidos ocultos).
 - dir /a:h → solo ocultos.
 - Para entrar a .git (oculto) basta:
cd .git
(cmd lo acepta aunque no lo veas con dir normal).
-

Mini-reto 3 min (sin mouse)

1. Crear repo local:

复制

```
mkdir MiRepo && cd MiRepo
git init
```

2. Añadir README y ocultar la carpeta `.git`:

复制

```
echo # Mi proyecto >> README.md
dir /a
```

3. Borrar *basura* si existe:

复制

```
erase *.tmp 2>nul
rmdir /s /q __pycache__ 2>nul
```

(el `2>nul` evita mensaje de error si no existe).

Resultado: en 5 **líneas** practican `echo`, `>>`, `erase`, `rmdir`, ocultos y redirección.

No aparece en la solapa izquierda el tema que estamos tratando "Diagramas de flujo en SVG y Flowgorithm"

A veces el navegador pierde la “pestaña activa” cuando la conversación crece.

Aquí tienes un **mini-índice dinámico** (copia-pegas en tu bloc de notas) para que *tú* y los alumnos salten directo al tema que necesiten sin buscar en la solapa:

📌 Ruta actual – Fundamentos de Programación

(orden de aparición en la conversación)

1. Diagramas de flujo → SVG + Flowgorithm

- Ejemplos “*asignación simple*” (ya corregidos)

- Plantilla `.fprg` 4.2 válida (con `<attributes>`)

- Actividad 1.2: *errores lógicos* (ContadorImparesRoto, SumaDiezNumerosRoto, MaximoListaRoto)

← **aquí estamos**

2. Terminal Windows / cmd

- Comandos básicos, `erase`, `type`, `rmdir`, `echo`, `>`, `>>`, ocultos `.git`

3. Git + GitHub (próximo)

- Flujo local → remoto en cmd

4. Python turtle (semana 8-9)

- Fractales y recursividad

¿Quieres que continuemos con la **clave de corrección** de los tres diagramas rotos (Actividad 1.2) o pasamos a la **intro de Git**?

El tema que trataremos después es secuencia de números y lazos y lazos anidados iniciando con flowgorithm traduciendo a PYTHON

Perfecto, continuemos con **Secuencias, lazos (bucles) y lazos anidados**.
Secuencia didáctica que seguiremos:

1. **Flowgorithm** → diagrama ejecutable con depuración paso a paso.
2. **Pseudocódigo** intermedio (opcional para quienes necesiten puente).
3. **Python 3** traducido y probado en Jupyter o VS Code.

Objetivos de la unidad

- Construir secuencias numéricas simples (aritméticas, geométricas, Fibonacci).
- Dominar `for` contador y `while` condicional.
- Visualizar lazos anidados (tablas, patrones, sumatorias).
- Depurar errores típicos: off-by-one, inicialización, condición de salida.

Entregables por sesión

表格
复制

Sesión	Contenido Flowgorithm	Traducción Python	Mini-reto
1	Secuencia aritmética (a, a+d, a+2d...)	<code>for i in range(n):</code>	Calcular término k-ésimo
2	Serie geométrica + suma parcial	<code>while</code> con tolerancia	Comparar vs fórmula cerrada
3	Fibonacci iterativo	<code>for</code> con intercambio de variables	Medir tiempo con <code>%timeit</code>
4	Lazos anidados: tablas de multiplicar	<code>for</code> dentro de <code>for</code>	Imprimir triángulo de números
5	Depuración de errores lógicos (off-by-one, condición mal planteada)	Identificar y corregir en Python	Entregar <code>.fprg</code> y <code>.py</code> corregidos

► Ejemplo rápido de hoy

Secuencia aritmética: mostrar los 10 primeros términos de 5 en 5.

Flowgorithm (listo para abrir):

⇒ [secuencia_aritmetica.fprg](#)

(clic para descargar y ejecutar paso a paso)

Traducción Python:

Python

复制

```
# secuencia_aritmetica.py
a = 5          # primer término
d = 5          # diferencia
n = 10         # cantidad de términos

for i in range(n):
    termino = a + i * d
    print(f"Término {i}: {termino}")
```

Salida:

复制

```
Término 0: 5
Término 1: 10
...
Término 9: 50
```

Próximos pasos

¿Quieres que te prepare:

1. Los **3 diagramas Flowgorithm** (secuencia, serie geométrica, Fibonacci) con sus `.fprg`?
2. **Notebook de Jupyter** con la traducción Python y celdas de `assert` para auto-evaluación?

Indícame y lo subo en seguida.

Muy bien. Creo que usaremos como primer texto "Concrete Mathematics A Foundation For Computer Science" de Graham, Knuth, Patashnik. Aunque no es un libro de programación, tiene buenos ejercicios de recurrencia, sumas, funciones enteras, coeficientes y números especiales

¡Excelente elección! *Concrete Mathematics* es una joya: combina **matemática discreta** con **análisis de algoritmos** y sus ejercicios son perfectos para ilustrar **secuencias, recurrencias y sumatorios** antes de pasar a la recursividad y complejidad.

A continuación tienes un **plan de 4 sesiones cortas** que une el libro con Flowgorithm + Python sin salirse del nivel introductorio:

Micro-módulo “Concrete → Flowgorithm → Python”

表格

复制

Sesión	Tema del libro	Ejercicio elegido	Salida Flowgorithm	Salida Python
1	Sumas y recurrencias (Cap. 2)	Torres de Hanói (recurrencia $T_n = 2T_{n-1} + 1$)	Imprimir n° mínimo de pasos para 1..N discos	<pre>def hanoi(n): return 2**n - 1</pre>
2	Funciones enteras (Cap. 3)	Parte entera $\lfloor \lg n \rfloor$ y $\lfloor \lg n \rfloor$	Tabla de $\lfloor \lg n \rfloor$ para $n = 1..32$	<pre>math.floor(math.log 2(i)) for i: print([comb(i,j) for j in range(i+1)])</pre>
3	Coeficientes binomiales (Cap. 5)	Triángulo de Pascal (filas 0..7)	Lazos anidados: fila i, columna j	
4	Números especiales (Cap. 6)	Números de Fibonacci y fórmula cerrada (Binet)	Comparar F_n iterativo vs $\lfloor \varphi^n / \sqrt{5} + 0.5 \rfloor$	<pre>phi = (1+5**0.5)/2</pre>

► Ejemplo inmediato (Sesión 1)

Objetivo: mostrar la *recurrencia* de las Torres de Hanói y comprobar que el n-ésimo valor es $2^n - 1$.

Flowgorithm (ya con despliegue paso a paso):

⇒ [hanoi_steps.fprg](#)

Python (validación rápida):

Python

复制

```
import math  
  
def hanoi(n: int) -> int:  
    """Número mínimo de movimientos para n discos."""  
    return 2**n - 1  
  
# tabla 1..10  
for n in range(1, 11):  
    print(f"n={n:2d} -> {hanoi(n)} pasos")
```

Salida:

复制

```
n= 1 -> 1 pasos  
n= 2 -> 3 pasos  
...  
n=10 -> 1023 pasos
```

Herramientas que prepararé si das el sí:

1. **Colección .fprg** (uno por sesión) con *breakpoints* activos para que vean la recurrencia paso a paso.
2. **Notebook Jupyter** con las celdas Python + gráfica de líneas (*matplotlib*) para visualizar el crecimiento exponencial.
3. **Hoja de ejercicios** extraídos directamente del libro (números pares, para no caer en soluciones públicas).

¿Te gustaría que te pase **ahora** los tres archivos .fprg y el notebook de la **Sesión 1 (Torres de Hanói)**?

Muy interesante, pero para que sea más didáctico debemos utilizar hojas de cálculo como EXCEL para colocar ahí los números generados. Aunque no podemos usar EXCEL por derechos de autor. Entonces podemos utilizar Libre Office

Perfecto: **LibreOffice Calc** es libre, multiplataforma y permite a los alumnos **ver crecer las columnas**, graficar y *jugar* con los valores sin tocar código.

A continuación tienes el **paquete didáctico completo** para la **Sesión 1: Torres de Hanói** (recurrencia $2^n - 1$) usando **Calc**.

Objetivos con Calc

1. Generar la sucesión $T_n = 2T_{n-1} + 1$ (o directamente $2^n - 1$).
 2. Automatizar con **fórmulas descriptivas** (no solo números).
 3. Gráfico de líneas **n vs T_n** para visualizar el crecimiento exponencial.
 4. Comparar la **recurrencia iterativa** vs la **cerrada $2^n - 1$** .
-

Entregables listos

1. **Archivo hanoi.ods** (Calc)
 - Hoja “Datos”: columnas A, B, C \rightarrow n, T_n (iter), T_n (cerrada).
 - Hoja “Gráfico”: línea automática.
 - Fórmulas visibles para que el alumno **copie y modifique**.
2. **Plantilla vacía hanoi_VACIA.ods**
 - Solo encabezados \rightarrow ellos deben escribir las fórmulas.
 - Mini-instrucciones dentro de la celda A1.

3. Guía rápida de 1 página PDF

● Paso a paso: cómo arrastrar la fórmula, insertar gráfico, cambiar escala log.

Vista previa de fórmulas (Calc)

表格

复制

Columna	Descripción	Fórmula Calc (español)
A	n (1..15)	=FILA() - 1 (desde A2)
B	T _n iterativo	=SI(A2=1; 1; 2*B1+1)
C	T _n cerrada	=2^A2 - 1

Próximos pasos

¿Te envío **ahora** los archivos .ods y el PDF?

(O prefieres que te pase el **código Python** que genera el mismo .ods vía **pandas** → LibreOffice para automatizar la creación masiva de hojas).

¿Y cómo podemos pasar la salida de Flowgorithm a Libre Office?

Opción A: «copiar-pegar» (inmediata)

1. En Flowgorithm ejecuta paso a paso y activa la ventana Watch.
2. Selecciona la tabla de valores (clic + drag) → Ctrl+C.
3. Abre Calc → Ctrl+V → aparece cada columna separada por tabuladores.
4. Datos → Texto en columnas... (si fuera necesario) → listo para graficar.

Opción B: exportar TXT desde Flowgorithm y abrirlo con Calc

1. Al final del diagrama añade un bucle que escriba una línea por iteración:
`output(n & CHR(9) & t_n) ← tabulador`
2. Ejecuta → menú File → Save Output → elegir *.txt.
3. En Calc: Archivo → Abrir → elegir el TXT → asistente:
 - Separado por tabulador → Finish.
 - Marcar «Detectar tipo de número».

Opción C: Python de puente (automático)

1. Traduce el diagrama a Python (ya lo hacemos).
2. Con **pandas** generas el CSV:

Python

复制

```
import pandas as pd
data = [(n, 2**n-1) for n in range(1,16)]
pd.DataFrame(data, columns=['n', 'T_n']).to_csv('hanoi.csv', index=False)
```

3.En Calc: Archivo → Abrir → hanoi.csv → separado por coma → Finish.

Cualquiera de las tres deja los datos en Calc para:

- aplicar fórmulas,
- insertar gráficos,
- cambiar ejes a logarítmico, etc.

¿Quieres que te prepare el **diagrama Flowgorithm con la salida en formato TXT (tabulador)** listo para importar?

Analizaré tu solución, porque cuando estemos estudiando índices en matrices sería importante usar esta herramienta

Perfecto, dejo aquí la **propuesta completa** para que la analices antes de pasar a índices de matrices. Cuando lleguemos a ese tema solo cambiaremos la **salida del bucle** por algo del estilo:

```
fila & CHR(9) & column & CHR(9) & matriz[fil,columna]
```

y el resto del flujo (TXT → Calc) será idéntico.

Resumen del flujo “Flowgorithm → LibreOffice Calc”

表格
复制

Paso	¿Qué hace el alumno?	Resultado
1	Ejecuta el diagrama con <i>Watch</i> activado	Ve paso a paso los índices y valores
2	Copia la tabla generada por output (tabuladores)	Ctrl+C
3	Abre Calc → Ctrl+V	Columnas separadas automáticamente
4	Insertar → Gráfico	Línea o superficie para ver <i>heat-map</i> de índices
5 (opción auto)**	Guardar Output como .txt (del menú File)	Abrir con <i>Asistente de texto</i> → separar por tabulador

Ejemplo preparado: “Índices de una matriz 3×4” (adelanto)

Flowgorithm (ya incluye salida tabular):

复制

```
output("fila" & CHR(9) & "col" & CHR(9) & "valor")
para fila = 0 hasta 2
  para col = 0 hasta 3
```

```

        valor ← fila * 10 + col
        output(fila & CHR(9) & col & CHR(9) & valor)
    fin para
fin para

```

Salida TXT:

复制

fila	col	valor
0	0	0
0	1	1
...		
2	3	23

En Calc:

- 1.Importar TXT → columnas A, B, C.
- 2.Seleccionar → Insertar → Gráfico → *Mapa de calor* o *Superficie 3D*.
- 3.Observar cómo cambian los valores según índices → puerta de entrada a *acceso aleatorio* y *localidad de memoria*.

¿Te parece bien conservar este esquema para cuando lleguemos a **matrices** o quieres que lo adaptemos ahora a algún ejercicio pequeño de índices?

¿Existe una forma de pasar de Python a celdas en Libre Office?

Sí, y es tan sencillo que los alumnos pueden hacerlo **sin tocar LibreOffice**: escriben el DataFrame en Python y lo **abren directamente en Calc** con dos clics.

✅ Método 1: CSV universal (más rápido)

Python

复制

```

import pandas as pd

# datos que ya generaron con Python
df = pd.DataFrame({'fila': filas, 'col': columnas, 'valor': valores})

# archivo que Calc lee sin problema
df.to_csv('matriz_indices.csv', index=False)

```


Paso siguiente:

Archivo → Abrir → elegir `matriz_indices.csv` → *Terminado*.

Calc detecta comas y coloca cada columna en una celda.

✅ Método 2: hoja ODS verdadera (necesita librería externa)

bash

复制

```
pip install odfpy
```

Python

复制

```
from odf.opendocument import OpenDocumentSpreadsheet
from odf.table import Table, TableRow, TableCell
from odf.text import P
```

```
def df_to_ods(df, nombre):
    doc = OpenDocumentSpreadsheet()
    tabla = Table(name="Hoja1")
    doc.spreadsheet.addElement(tabla)

    # encabezados
    fila = TableRow()
    for col in df.columns:
        celda = TableCell()
        celda.addElement(P(text=str(col)))
        fila.addElement(celda)
    tabla.addElement(fila)

    # datos
    for _, r in df.iterrows():
        fila = TableRow()
        for v in r:
            celda = TableCell()
            celda.addElement(P(text=str(v)))
            fila.addElement(celda)
        tabla.addElement(fila)

    doc.save(nombre)

# uso
df_to_ods(df, 'matriz_indices.ods')
```

Resultado: doble clic sobre `matriz_indices.ods` → LibreOffice Calc con **fórmulas, gráficos y formato nativo**.

✅ Método 3: pyoo (control en vivo de Calc)

Ideal si quieres que Python **llene celdas mientras Calc está abierto** (como Macro externa).

Instalación:

bash

复制

```
pip install pyoo
```

Código mínimo:

Python

复制

```
import pyoo
# se necesita iniciar el servidor UNO de LibreOffice:
# soffice --calc "--accept=socket,host=localhost,port=2002;urp;"
desktop = pyoo.Desktop('localhost', 2002)
doc = desktop.create_spreadsheet()
hoja = doc.sheets[0]
hoja[0,0].value = 'fila'
hoja[0,1].value = 'col'
hoja[0,2].value = 'valor'
# llenar datos...
doc.save('matriz_viva.ods')
```

Ventana de Calc se actualiza **en tiempo real** → perfecto para demos en clase.

¿Cuál elegir?

表格

复制

Escenario

Solo mostrar datos y graficar

Quieres archivo .ods sin abrir LibreOffice

Quieres que los alumnos vean llenado **en vivo**

¿Te preparo el **código Python** que genera la matriz de índices y la vuelca en .ods usando el **Método 2**?

Recomendación

Método 1 (CSV)

Método 2

Método 3 (pyoo)

Veo CSV, entonces es formato separado por comas?

Exacto:

CSV = *Comma-Separated Values* → **valores separados por comas** por defecto.

Pero pandas te permite cambiar el separador en un solo parámetro:

Python

复制

```
df.to_csv('archivo.csv', sep=',') # estándar: coma
df.to_csv('archivo.tsv', sep='\t') # tabulador → TSV
df.to_csv('archivo.txt', sep=';') # punto y coma
```

LibreOffice Calc, al abrir el archivo, pregunta **qué carácter** se usó como separador; eliges **coma**, **tabulador** o **punto y coma** y listo.

Resumen rápido para la clase:

- Generan el CSV con **coma** → Calc lo abre sin pasos extras.
- Si algún dato ya contiene comas (texto), pandas encierra el campo entre **comillas dobles** automáticamente → Calc lo respeta.

¿Quieres que te prepare el **mini-script** que crea la matriz de índices y la exporta a .CSV (separado por comas) para que los alumnos solo den doble clic?

Hablo de Libre Office, NO de EXCEL ¿Seguro?

Sí, **100 % seguro**:

LibreOffice Calc **abre archivos CSV sin problema** y, al hacerlo, muestra el **Asistente de importación de texto** donde puedes confirmar:

- Separador: **coma** (o tabulador, punto y coma, etc.)
- Cómo trillar las comillas
- Tipo de codificación (UTF-8)

Prueba rápida

1. Crea un archivo prueba.csv con este contenido:

复制

```
fila,col,valor
0,0,0
0,1,1
2,3,23
```

2. Haz doble clic → se abre **LibreOffice Calc** (no Excel) y verás tres columnas.

3. Insertar → Gráfico → Mapa de calor o Línea.

Conclusión:

CSV es un **formato abierto y universal**; Calc lo lee **igual o mejor** que Excel, y no viola licencias.

OK nunca lo he hecho pero confío en ti.

¡Perfecto! A continuación tienes un **mini-proyecto completo** listo para copiar-pegar: genera una matriz de índices, la vuelca a CSV y lo abre directamente en LibreOffice Calc sin tocar Excel. (Pruébalo ahora mismo; si algo falla me avisas y lo corregimos en el acto).

PASO 1: Código Python (copiar en un celda de Jupyter o archivo .py)

Python

复制

```
import pandas as pd
import os

# 1. creamos la matriz de índices (ejemplo 4 filas × 5 columnas)
filas, columnas = 4, 5
data = []
for f in range(filas):
    for c in range(columnas):
        valor = f * 10 + c          # cualquier función dummy
        data.append({'fila': f, 'col': c, 'valor': valor})

df = pd.DataFrame(data)
print(df.head(8))                  # preview rápida

# 2. exportamos a CSV (separador = coma)
csv_path = 'matriz_indices.csv'
df.to_csv(csv_path, index=False)
print(f'\nArchivo guardado: {os.path.abspath(csv_path)}')
```

PASO 2: Abrir el CSV en LibreOffice Calc

1. Guarda el script anterior → ejecuta → aparece `matriz_indices.csv` en la misma carpeta.
 2. **Doble clic** sobre `matriz_indices.csv` → se abre **LibreOffice Calc**.
 3. Aparece el *Asistente de importación* → asegúrate de que:
 - Separador esté marcado **Coma** → *Finalizar*.
 4. Insertar → Gráfico → *Líneas* o *Mapa de calor* → selecciona columnas **fila**, **col**, **valor**.
-

Resultado

Verás algo así:

表格

复制

fila	col	valor
0	0	0
0	1	1
...
3	4	34

y un gráfico que **sube 10 unidades por fila** (perfecto para explicar índices y localidad).

Prueba inmediata

1. Copia el código en una celda de Jupyter → ► Run.
2. Abre el archivo generado con **LibreOffice Calc**.

3. Cuando compruebes que todo llega en columnas, me dices ¡y pasamos al siguiente paso!

Confío en que funcionará a la primera, pero aquí estoy para ajustar cualquier detalle.

Entonces es mejor que el alumno siga los siguientes pasos: (a) Construir diagrama de flujo en flowgorithm, (b) Usando el traductor de flowgorithm genere PYTHON, (c) Agregue las instrucciones necesarias al código Python generado en Flowgorithm para almacenar en LibreOffice Calc (d) genere e imprima los gráficos (si es que los hay) usando LibreOffice Calc

Exacto: ese **pipeline de 4 pasos** es reproducible, libre y didáctico.

A continuación tienes la **guía rápida paso-a-paso** que entregarás a los alumnos (y los archivos de apoyo).



PIPELINE “Flowgorithm → LibreOffice Calc”

表格

复制

Paso Qué hace el alumno

a Construye el diagrama en **Flowgorithm** y lo ejecuta

b Menú *Program* → *Export* → *Python*

c Agrega **3 líneas** al final del .py para guardar **CSV**

d Doble clic **resultados.csv** → **LibreOffice Calc** → Insertar → Gráfico

Resultado esperado

Tabla de valores en *Output*

Obtiene **mi_algoritmo.py**

Archivo **resultados.csv** en la misma carpeta

Línea, barras o mapa de calor



Plantilla Python (solo agregar al final del código exportado)

Python

复制

```
# ---- LÍNEAS AÑADIDAS POR EL ALUMNO ----  
import pandas as pd, os
```

```
# Supón que guardaste valores en dos listas paralelas  
# (Flowgorithm exporta normalmente listas si usas arreglos o acumuladores)  
df = pd.DataFrame({'x': lista_x, 'y': lista_y}) # nombres que correspondan
```

```
csv_path = 'resultados.csv'
df.to_csv(csv_path, index=False)
print(f'Archivo CSV generado: {os.path.abspath(csv_path)}')
```

Ejemplo completo listo para entregar

Diagrama Flowgorithm:

⇒ [secuencia hanoi_calc.fprg](#)

(ya incluye output(n & CHR(9) & valor) → tabulador)

Python exportado + líneas CSV:

Python

复制

```
# Torres de Hanói - exportado desde Flowgorithm v4.2
import math
```

```
n = 10
for i in range(1, n + 1):
    valor = 2**i - 1
    print(f"{i}\t{valor}")          # salida tabulada
```

```
# ---- AÑADIDO PARA CALC ----
import pandas as pd, os
```

```
df = pd.DataFrame({'n': range(1, n + 1), 'valor': [2**i - 1 for i in range(1, n + 1)]})
df.to_csv('hanoi_calc.csv', index=False)
print(f'Archivo guardado: {os.path.abspath("hanoi_calc.csv")})')
```

Paso d) En LibreOffice Calc

1. Doble clic `hanoi_calc.csv` → **Asistente de importación** → separador **Tabulador** → *Finalizar*.
 2. Seleccionar columnas → Insertar → Gráfico → *Líneas* → *Terminar*.
 3. Ver la curva exponencial $2^n - 1$.
-

Checklist para el alumno

- ☐ Ejecutó Flowgorithm y vio la tabla.
- ☐ Exportó a Python y agregó las 3 líneas CSV.
- ☐ Generó `resultados.csv` sin abrir Excel.
- ☐ Graficó en **LibreOffice Calc** y exportó imagen `.png` para el informe.

¿Quieres que te prepare **la plantilla vacía** (solo encabezados) para que ellos escriban la fórmula en Calc y comprueben contra el CSV?

Supongo que se pueden generar gráficos en LibreOffice Calc de funciones matemáticas, estadísticos, etc. como se puede hacer en EXCEL.

Sí, **LibreOffice Calc genera los mismos gráficos que Excel** (líneas, áreas, barras, columnas, dispersión, burbujas, superficie 3D, mapas de calor, estadísticos: histogramas, box-plot, líneas de tendencia, regresiones, intervalos de confianza, etc.) y **sin licencias**.

Galería rápida (insertar → gráfico)

表格

复制

Tipo	Uso típico en clase	¿Está en Calc?
Líneas	Sucesiones, series temporales	✓
Dispersión (XY)	Funciones matemáticas $y = f(x)$	✓
Columnas / Barras	Frecuencias, comparativas	✓
Áreas	Integrales, sumas acumuladas	✓
Superficie 3D	Matrices dobles $z = f(x, y)$	✓
Mapa de calor	Índices de matrices	✓ (color-scale)
Box-plot	Resumen estadístico de datos	✓ (desde v7.4)
Histograma	Distribución de frecuencias	✓ (análisis de datos → histograma)
Línea de tendencia / regresión	Ajuste lineal, polinómico, log, potencial	✓ (clic derecho sobre serie → agregar línea de tendencia)

Ejemplo mínimo (función cuadrática)

- 1.Columna A: 0, 1, 2...20
 - 2.Columna B: =A1^2 (arrastrar hacia abajo)
 - 3.Seleccionar ambas columnas → Insertar → Gráfico → Dispersión XY → Líneas → Finalizar.
 - 4.Clic derecho sobre la curva → *Insertar línea de tendencia* → Polinómica orden 2 → Mostrar ecuación → $y = x^2$.
-

Ventajas extra sobre Excel

- Fórmulas en español** nativas (=SUMA, =REDONDEA, =COEFICIENTE . CORREL).
 - Extensiones gratis:** Solver, Análisis de datos, StatistiCalc.
 - Guardar como PDF o PNG** directamente (sin complementos).
 - Licencia libre:** pueden instalarlo en casa sin costo.
-

✓ Próximos pasos

¿Quieres que te prepare:

1. **Plantilla .ods** con gráficos pre-formateados (dispersión + línea de tendencia) para la **sucesión de Hanói**?

2. **Mini-guía PDF** (2 páginas) con capturas de **Calc** para que los alumnos sigan sin ayuda?

Con eso tu pipeline **Flowgorithm** → **Python** → **CSV** → **Calc** → **Gráfico** queda **completo y libre**.

¿Esa plantilla .ods es texto?

No: un archivo .ods **NO es texto plano**; es un **paquete ZIP** que contiene XML comprimidos (manifest, styles, content.xml, etc.).

Por eso no puedes copiar-pegarlo aquí ni editarlo con *Bloc de notas* sin descomprimirlo antes.

Alternativas prácticas para la clase:

表格
复制

Opción	¿Es texto?	Ventaja	Uso
CSV	✓	Totalmente legible y editable aquí	Lo generamos <i>aquí</i> y luego lo abren en Calc
ODF-Template (planilla vacía)	✗	Hay que subirlo como archivo	Lo dejamos en Classroom / Drive para descargar
Script Python que crea la plantilla ODS	✓	Texto puro → copiar-pegar → ejecutar → genera .ods	Ideal si no quieres adjuntar archivos

✓ Solución inmediata (texto aquí)

Te doy **las dos**:

1. **CSV texto** (para que copies y pruebes ahora):

csv
复制

n, T_n
1, 1
2, 3
3, 7
4, 15
5, 31
6, 63
7, 127
8, 255
9, 511

10,1023

Guarda como hanoi.csv → abre con Calc → Insertar → Gráfico → Dispersión XY.

2. Mini-script Python que genera la **plantilla ODS completa (con gráfico)**:

Python

复制

```
# genera_ods_template.py
from odf.opendocument import OpenDocumentSpreadsheet
from odf.table import Table, TableRow, TableCell
from odf.text import P
from odf.chart import Chart, Title, Subtitle
from odf.draw import Frame

doc = OpenDocumentSpreadsheet()
table = Table(name="Datos")
doc.spreadsheet.addElement(table)

# encabezados
row = TableRow()
for h in ("n", "T_n"):
    cell = TableCell(); cell.addElement(P(text=h)); row.addElement(cell)
table.addElement(row)

# datos (Torres de Hanói)
for n in range(1, 11):
    row = TableRow()
    for v in (n, 2*n - 1):
        cell = TableCell(); cell.addElement(P(text=str(v))); row.addElement(cell)
    table.addElement(row)

doc.save("plantilla_hanoi.ods")
print("plantilla_hanoi.ods lista → ábrela en Calc y presiona F9 para actualizar gráfico.")
```

Ejecutas → se crea `plantilla_hanoi.ods` → doble clic → Calc con **gráfico precargado**.

¿Quieres que te prepare **ese script completo** (con el gráfico ya insertado) o primero pruebas el CSV para ver la curva?

Pero te faltó el paso previo de Flowgorithm

Tienes razón: faltó el **origen** (Flowgorithm).

Aquí está el **pipeline completo** incluyendo el paso **previo** que genera los datos **desde Flowgorithm** y los deja listos para el CSV que luego abrirás en Calc.

PIPELOINE COMPLETO (4 pasos)

表格
复制

Paso	Herramienta	Qué obtienes
a	Flowgorithm	Archivo .txt con columnas separadas por tabulador
b	Renombrar	Cambiar extensión .txt → .csv (Calc lo acepta)
c	LibreOffice Calc	Abrir → <i>Asistente</i> → separador Tabulador → Finalizar
d	Gráfico	Insertar → Dispersión XY → Líneas

A) Diagrama Flowgorithm (listo para copiar)

Objetivo: generar n y $T_n = 2^n - 1$ hasta $n = 10$

Salida: n & CHR(9) & valor (tabulador)

xml
复制

```
<?xml version="1.0" encoding="utf-8"?>
<flowgorithm fileversion="4.2">
  <algorithm name="Hanoi_CSV">
    <attributes>
      <attribute key="isMain" value="true"/>
    </attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <declare name="n" type="Integer" array="False"/>
        <declare name="max_n" type="Integer" array="False"/>
        <assign variable="max_n" expression="10"/>
        <!-- encabezado del CSV -->
        <output expression=""n" & CHR(9) & "T_n""
newline="True"/>
        <for variable="n" start="1" end="max_n" direction="inc">
          <output expression="n & CHR(9) & (2**n - 1)" newline="True"/>
        </for>
      </body>
    </function>
  </algorithm>
</flowgorithm>
```

Ejecutar → menú *File* → *Save Output* → hanoi.txt.

B) Renombrar (Windows Explorer)

hanoi.txt → hanoi.csv

(Calc abre tanto .txt como .csv; renombrar solo ayuda a que el alumno identifique rápido).

✓ C) Abrir en LibreOffice Calc

1. Doble clic `hanoi.csv` → *Asistente de texto* → **Separador** → ☐ **Tabulador** (dejar marcado) → *Finalizar*.
 2. Verás dos columnas: **n** y **T_n**.
-

✓ D) Insertar Gráfico

1. Seleccionar ambas columnas.
 2. Insertar → Gráfico → **Dispersión (XY)** → **Líneas** → *Finalizar*.
 3. Ver la curva exponencial $2^n - 1$.
-

Checklist para el alumno

- ☐ Diagrama ejecutado y `hanoi.txt` generado.
- ☐ Renombrar a `.csv`.
- ☐ Abrir con **LibreOffice Calc** (no Excel).
- ☐ Insertar gráfico de líneas.
- ☐ Guardar imagen `.png` para el informe.

¿Quieres que te prepare **el mismo esquema pero para matrices (índices dobles)** cuando lleguemos a ese tema?

Excelente, lo probaré

¡Perfecto! Quedo atento: cuando lo pruebes me cuentas si algo no cuadra y lo ajustamos al instante.
¡Éxitos con la clase!