# Documento Explicativo para Sustentación: Módulo de Gramática

Este documento explica el funcionamiento del módulo de gramática implementado en Python para el proyecto final de Matemáticas Discretas II, con el objetivo de facilitar la sustentación.

## 📄 Estructura General del Módulo

El módulo de gramática permite:

1. Ingresar una gramática formal regular.
2. Verificar si una frase pertenece a esa gramática.
3. Generar frases válidas a partir de la gramática.

Todo esto se hace con estructuras básicas de Python: listas, diccionarios y conjuntos.

## 🔖 Representación de la Gramática

Una gramática se representa como un diccionario:

{  
 'V': {'S', 'A'}, # No terminales  
 'T': {'a', 'b'}, # Terminales  
 'S': 'S', # Símbolo inicial  
 'P': { # Producciones  
 'S': [['a', 'A'], ['b']],  
 'A': [['b']]  
 }  
}

## ⚙️ Funciones del Módulo

### 1. ingresar\_gramatica()

* El usuario ingresa:
  + No terminales (separados por espacio)
  + Terminales (separados por espacio)
  + Símbolo inicial
  + Reglas como: S -> a A
* El resultado es un diccionario con la gramática completa.

### 2. pertenece\_frase(frase, gramatica, max\_profundidad=10)

* Verifica si una frase (como "a b") puede ser generada desde el símbolo inicial.
* Usa **backtracking recursivo**:
  + Reemplaza no terminales por sus producciones.
  + Si la frase obtenida es igual a la de entrada, se acepta.
* Tiene un límite de profundidad para evitar ciclos infinitos.

### 3. generar\_frases(gramatica, n=10)

* Genera frases aleatorias a partir de la gramática.
* Usa elecciones aleatorias de producciones.
* Solo devuelve frases que contengan sólo terminales.

Ejemplo de uso:

frases = generar\_frases(g, n=10)  
for i, f in enumerate(frases):  
 print(f"Frase {i+1}: {f}")

## 📈 Argumentos para la Sustentación

Puedes decir lo siguiente:

* “Usé una estructura de diccionario que agrupa la información de la gramática de forma organizada.”
* “Para verificar frases, implementé un método recursivo que intenta derivar la frase desde el símbolo inicial, como en clase.”
* “Limito la profundidad de la recursión para evitar bucles infinitos, que podrían darse con gramáticas recursivas.”
* “No usé algoritmos como CYK porque prefiero entender y controlar el proceso de derivación paso a paso.”
* “Las frases generadas cumplen todas las reglas de la gramática porque parto desde el símbolo inicial y aplico producciones aleatorias.”

## 🌟 Conclusión

Este módulo cumple con los puntos de la rúbrica relacionados con:

* Ingreso de gramática.
* Verificación de pertenencia.
* Generación de frases válidas.

Está hecho con estructuras claras, es modular, y puedes explicar cada parte sin usar algoritmos avanzados. Así garantizas claridad y funcionalidad.