Tarea de Algoritmos

Introducción

Los algoritmos son un conjunto de pasos ordenados y finitos que se siguen para resolver un problema y finalizar una tarea con instrucciones claras y siguiendo un orden.

Estos algoritmos estarán organizados en pasos y numerados para determinar mejor la posición,

con ejemplos reales en Python, teniendo en cuenta los ejemplos en las imágenes y en algunos casos el flujo del algoritmo, cabe resaltar que estos algoritmos no son perfectos y no se tomaron en cuenta muchos aspectos importantes del desarrollo, digamos que la idea fue hacerlos lo más sencillos posibles, lo cual hace que en la práctica al programar se enfrente a problemas.

Estos algoritmos fueron diseñados con fines didácticos y no son soluciones completas ni robustas. Se enfocan en representar la lógica básica y estructura de resolución, omitiendo aspectos importantes como:

* Validación de datos de entrada.
* Manejo de errores o excepciones.
* Optimización de rendimiento.
* Casos extremos o inputs inválidos.
* Uso de funciones o estructuras de control más avanzadas.

En la práctica real, al programar, nos enfrentamos a desafíos adicionales como:

* Evitar bucles infinitos o resultados incorrectos por inputs inesperados.
* Considerar límites de memoria o tiempo de ejecución.
* Escribir código reutilizable y mantenible
* Documentar adecuadamente la lógica.

La intención aquí es entender la lógica fundamental, y luego sí avanzar hacia soluciones más completas y profesionales. Aprender algoritmos así, paso a paso, es como aprender a construir una base sólida para luego levantar una buena arquitectura de software.

# Algoritmo para sumar dos números.

* *Algoritmo simple*:

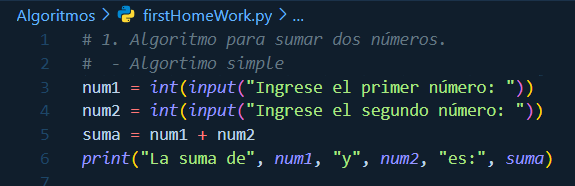
1. Iniciar.

2. Pedir un número entero y guardarlo en la variable *num1.*

3. Pedir otro número entero y guardarlo en la variable *num2.*

4. Sumar num1 + num2 y guardarlo en la variable sum.

5. Mostrar por pantalla o consola la variable sum.



* *Con funciones*:

1. Iniciar.

2. Pedir un número y guardarlo en la variable *num1.*

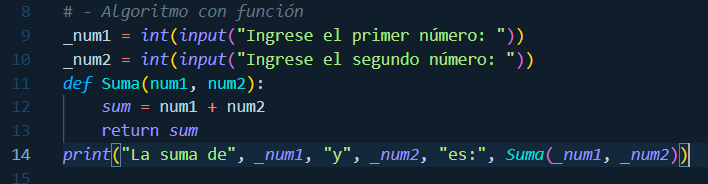
3. Pedir otro número y guardarlo en la variable *num2.*

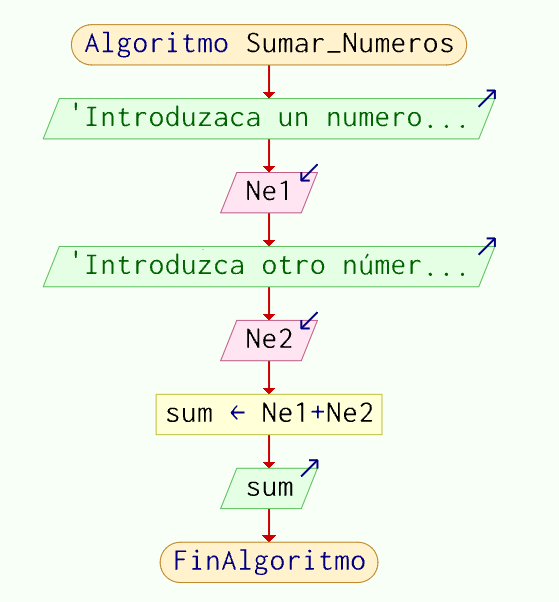
4. definir una función llamada *Suma* que reciba dos números como argumento y sume estos mismos y retorne el resultado.

Sumar num1 + num2 y guardarlo en la variable sum.

5. Llamar y mostrar por pantalla o consola la función *Suma* y pasarle como argumento las variables *num1* y *num2*.

6. Fin





# Algoritmo para encontrar el mayor de tres números.

1. Iniciar.

2. Pedir al usuario 3 números *a*, *b*, *c* y guardarlos en dichas variables.

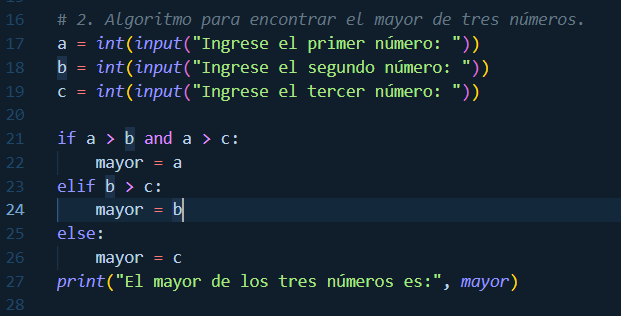
3. Comparar si *a* > *b* y *a* >*c* entonces *a* es el mayor.

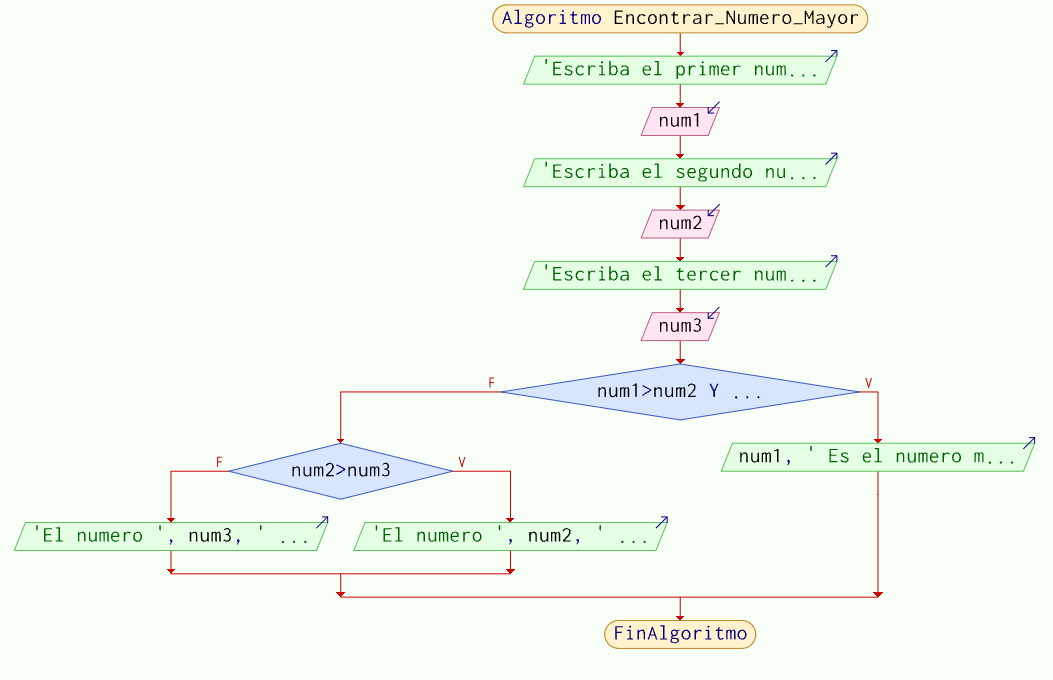
4. Si *b* >*c*, entonces *b* es el mayor.

5. Sino, *c* es el mayor

6. Mostrar el mayor.

7. Fin





# Algoritmo para calcular el factorial de un número.

* *simple.*

1. Iniciar.

2. Pedir al usuario un número y se guarda en la variable *n.*

3. Multiplicar el número por el número anterior a él.

4. Multiplicar el resultado por el número anterior a ese.

5. Seguir multiplicando hasta llegar a 1.

6. Fin.

* *Programming mode.*

1. Iniciar.

2. Pedir al usuario un número y se guarda en la variable *n.*

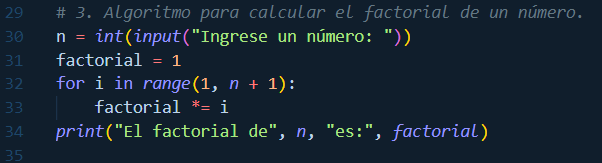
*3.* Inicializar *factorial = 1.*

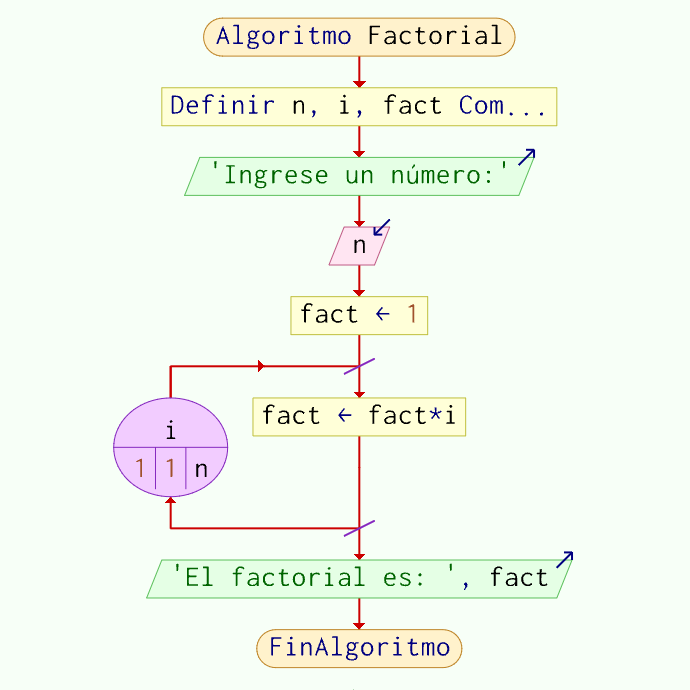
4. Iniciar un ciclo donde para *i* desde 1 hasta n, hacer:

*factorial* = *factorial* × *i.*

5. Mostrar el valor de *factorial*.

6. Fin.





# Algoritmo para verificar si un número es primo.

1. Iniciar.

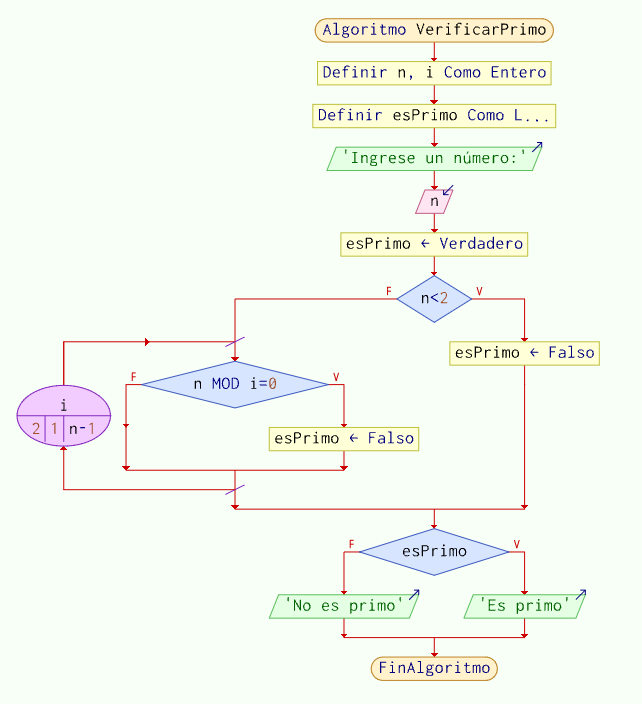
2. Pedir un número entero n.  
3. Si n < 2, no es primo.  
4. Para i desde 2 hasta n-1, hacer:

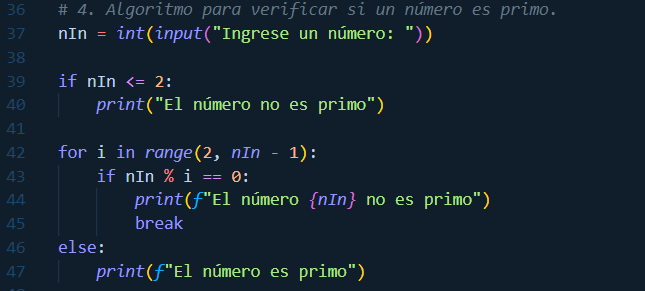
Si n % i == 0, entonces no es primo.

5. Si no se encontró ningún divisor, entonces es primo.

6. Mostrar el resultado.

7. Fin.

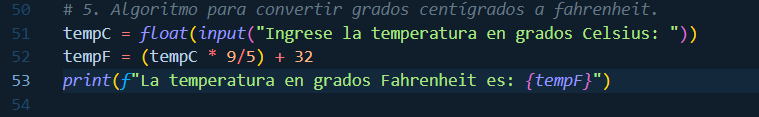


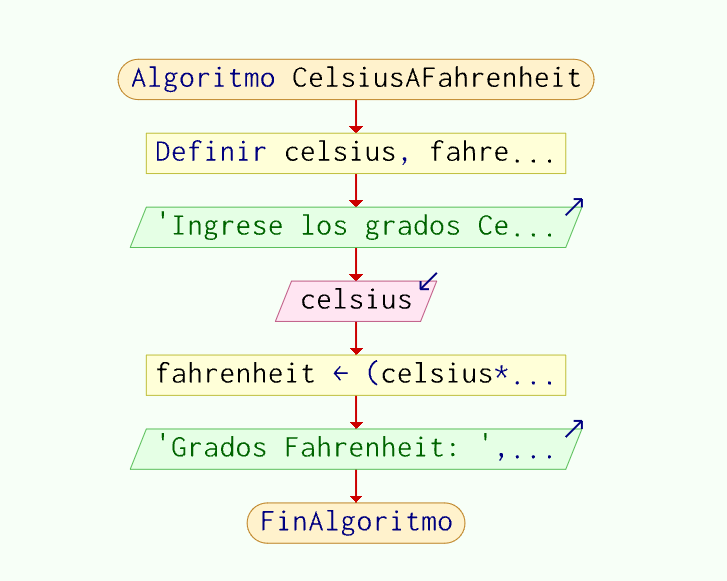


# Algoritmo para convertir de grados celsius a fahrenheit.

1. Iniciar.

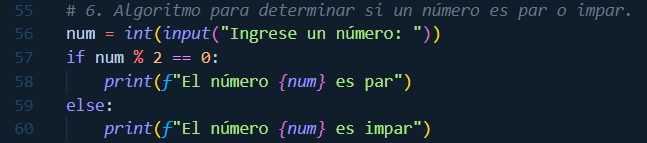
2. Pedir al usuario una temperatura en grados Celsius (tempC).  
3. Calcular tempF = (tempC × 9/5) + 32.  
4. Mostrar tempF.  
5. Fin.

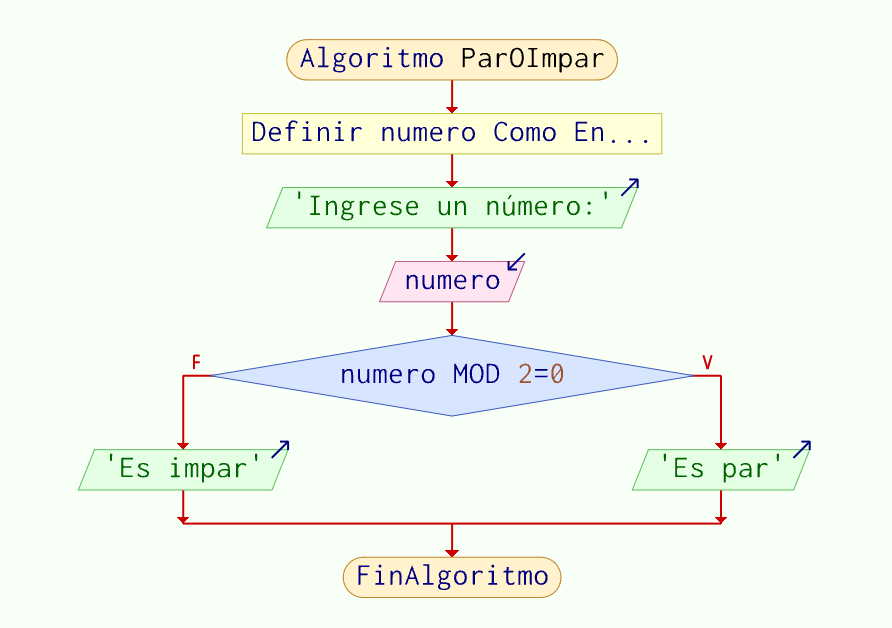




# Algoritmo para determinar si un número es par o impar.

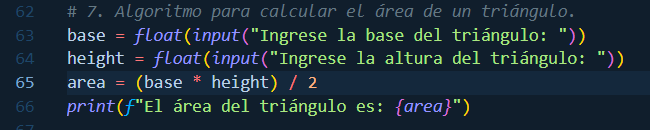
1. Iniciar.  
2. Pedir un número entero n.  
3. Si n % 2 == 0, entonces es par.  
4. Si no, es impar.  
5. Mostrar el resultado.  
6. Fin.

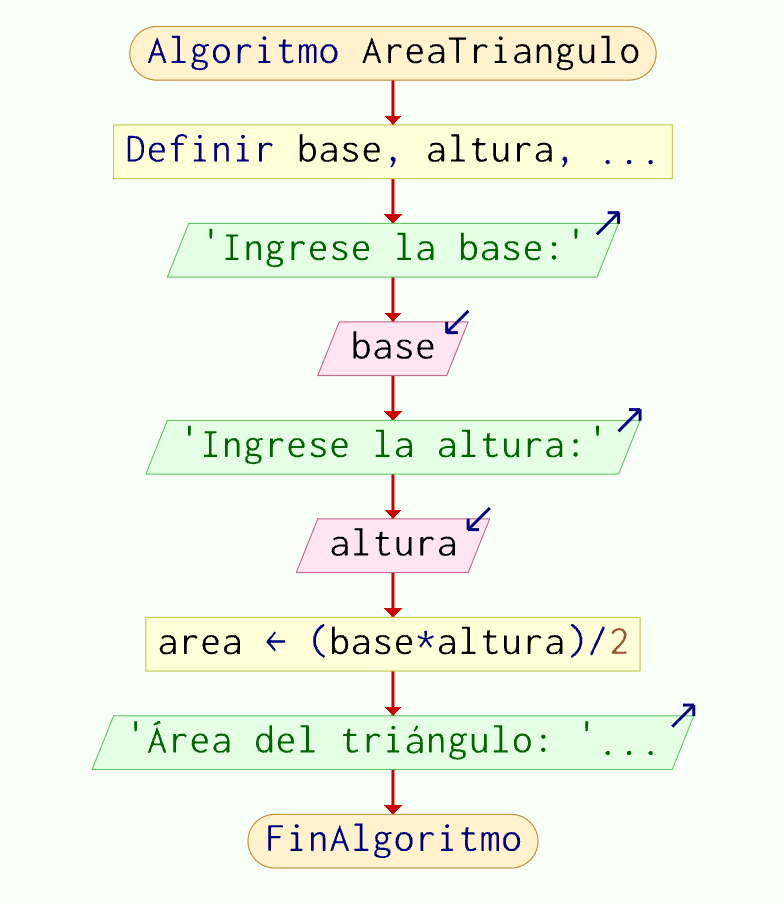




# Algoritmo para calcular el área de un triángulo.

1. Iniciar.  
2. Pedir la base del triángulo (base).  
3. Pedir la altura del triángulo (altura).  
4. Calcular área = (base × altura) / 2.  
5. Mostrar área.  
6. Fin.



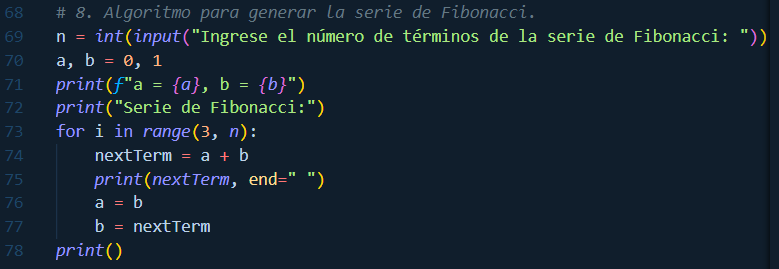


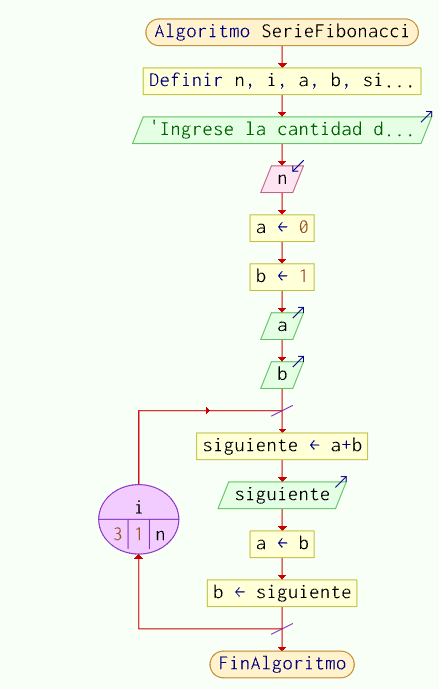
# Algoritmo para generar la serie de Fibonacci.

1. Iniciar.  
2. Pedir al usuario la cantidad de términos n.  
3. Inicializar a = 0, b = 1.  
4. Mostrar a y b.  
5. Para i desde 3 hasta n, hacer:

* siguiente = a + b
* Mostrar siguiente
* a = b
* b = siguiente

6. Fin.



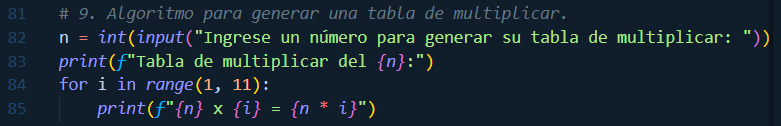


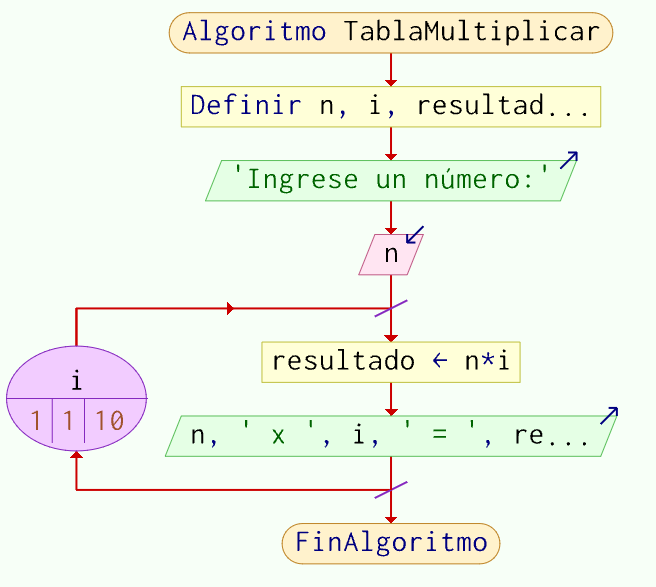
# Algoritmo para generar una tabla de multiplicar.

1. Iniciar.  
2. Pedir un número n.  
3. Para i desde 1 hasta 10, hacer:

* Calcular resultado = n × i
* Mostrar n × i = resultado

4. Fin.





# Algoritmo para calcular el promedio de una lista de números.

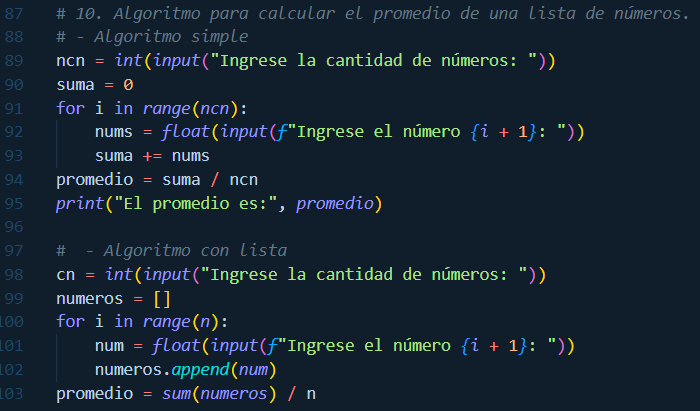
1. Iniciar.  
2. Pedir la cantidad de números ncn.  
3. Inicializar suma = 0.  
4. Repetir ncn veces:

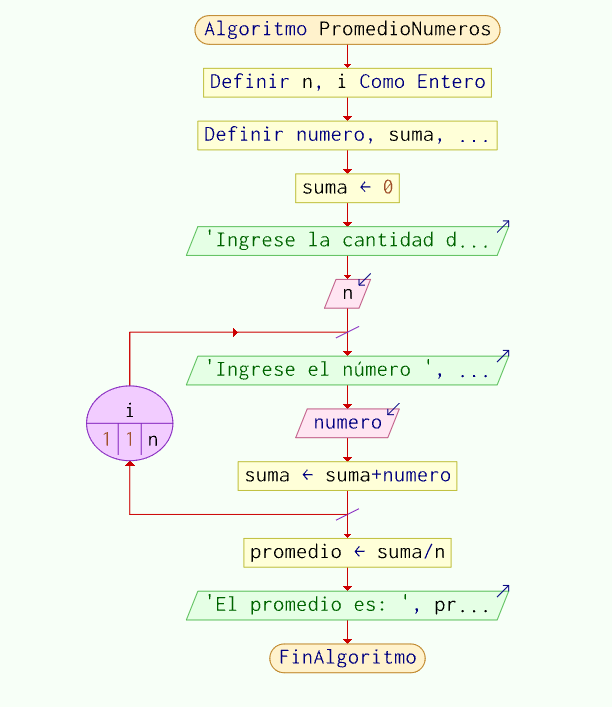
* Pedir un número y sumarlo a suma.

5. Calcular promedio = suma / ncn.

6. Mostrar promedio.

7. Fin.





# Algoritmo para calcular el área de un círculo.

1. Iniciar.  
2. Pedir el valor del radio r.  
3. Calcular área = π × r × r (usar 3.1416 si no se tiene π).  
4. Mostrar área.

5. Fin.

