


ESTRUCTURA DE DECISIÓN O CONDICIONAL:

if





Al momento hemos resuelto problemas aplicando fórmulas simples y usando la estructura de secuencia

La mayoría de problemas a resolver en programación involucran estructuras de control

- Estructuras de decisión o estructuras condicionales: tomar decisiones
- Estructuras de repetición o iteración: realizar un mismo proceso varias veces

ESTRUCTURA

if

Usos

- Evaluar alternativas para tomar decisiones
- Revisar las restricciones del programa
- Dar soluciones a casos especiales

if (si condicional)

- ▣ Evalúa una condición que da como resultado un valor booleano (valor lógico):

True (VERDADERO: la condición es verdadera) o

False (FALSO: la condición es falsa)

- # Valor booleano de la evaluación de una condición: usado para tomar decisiones en los programas
- # Ejecución del programa continuará por uno y solamente uno de dos caminos:
 - Si la condición es verdadera el flujo de ejecución sigue en el bloque de instrucciones que asociemos al verdadero. Las instrucciones asociadas al falso no se ejecutan
 - Si la condición es falsa el flujo de ejecución sigue en el bloque de instrucciones que asociemos al falso . Las instrucciones asociadas al verdadero no se ejecutan

Sintaxis de la estructura de control **if**

if condición:

→ expresión que da un valor booleano: True o False

instrucciones

→ se ejecutan cuando la condición es verdadera

else:

instrucciones

→ se ejecutan cuando la condición es falsa

"... *aprendiendo:*

estudiar y

practicar ..."

OPERADORES RELACIONALES

Condición: compara dos expresiones utilizando operadores relacionales

Sean x , y expresiones

$==$	igualdad	$x == y$
------	----------	----------

$!=$	diferente	$x != y$
------	-----------	----------

$>$	mayor que	$x > y$
-----	-----------	---------

$<$	menor que	$x < y$
-----	-----------	---------

$>=$	mayor o igual que	$x >= y$
------	-------------------	----------

$<=$	menor o igual que	$x <= y$
------	-------------------	----------

⌘ Antes de aplicar un operador relacional primero se calculan los valores de las expresiones que se van a comparar

⌘ Ejemplos de condiciones:

a, b, nota, n, opcion = 5, -10, 90, 21, "10"

$4 > 10$?

$65 \leq 65$?

$(a + b) * 2 < 75$?

$n < a ** 2$?

$a ** 2 > b + 20$?

$nota \geq 70$?

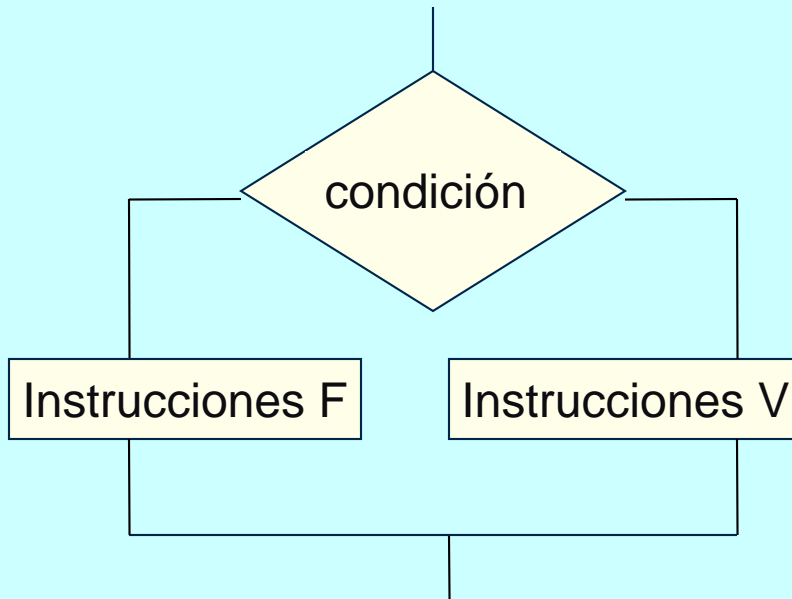
$n \% 10 == 0$?

$opcion \neq "10"$?

$a = 5$?

Diagrama de flujo

Pseudocódigo



if condición:

Instrucciones V

else:

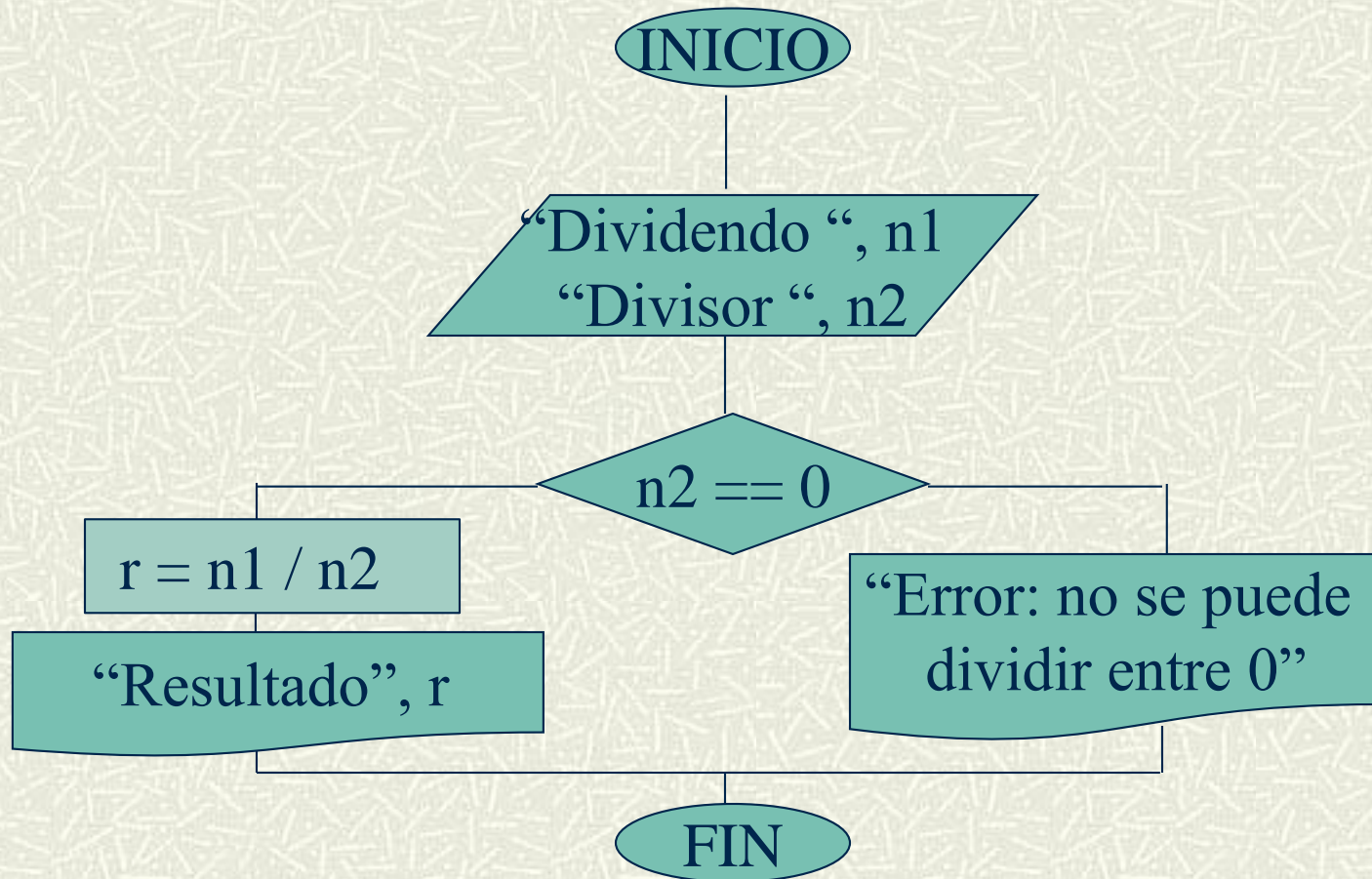
Instrucciones F

Las Instrucciones V deben indentarse con el if para indicar que pertenecen al verdadero. El bloque de las Instrucciones V termina cuando se encuentra la palabra else: u otra instrucción alineada con el if.

Las Instrucciones F deben indentarse con el else:

La primera instrucción alineada con el else: marca el fin de las Instrucciones F y de este if.

Ejemplo: Dar el resultado de una división aritmética, pero si el divisor es cero enviar un mensaje de error.

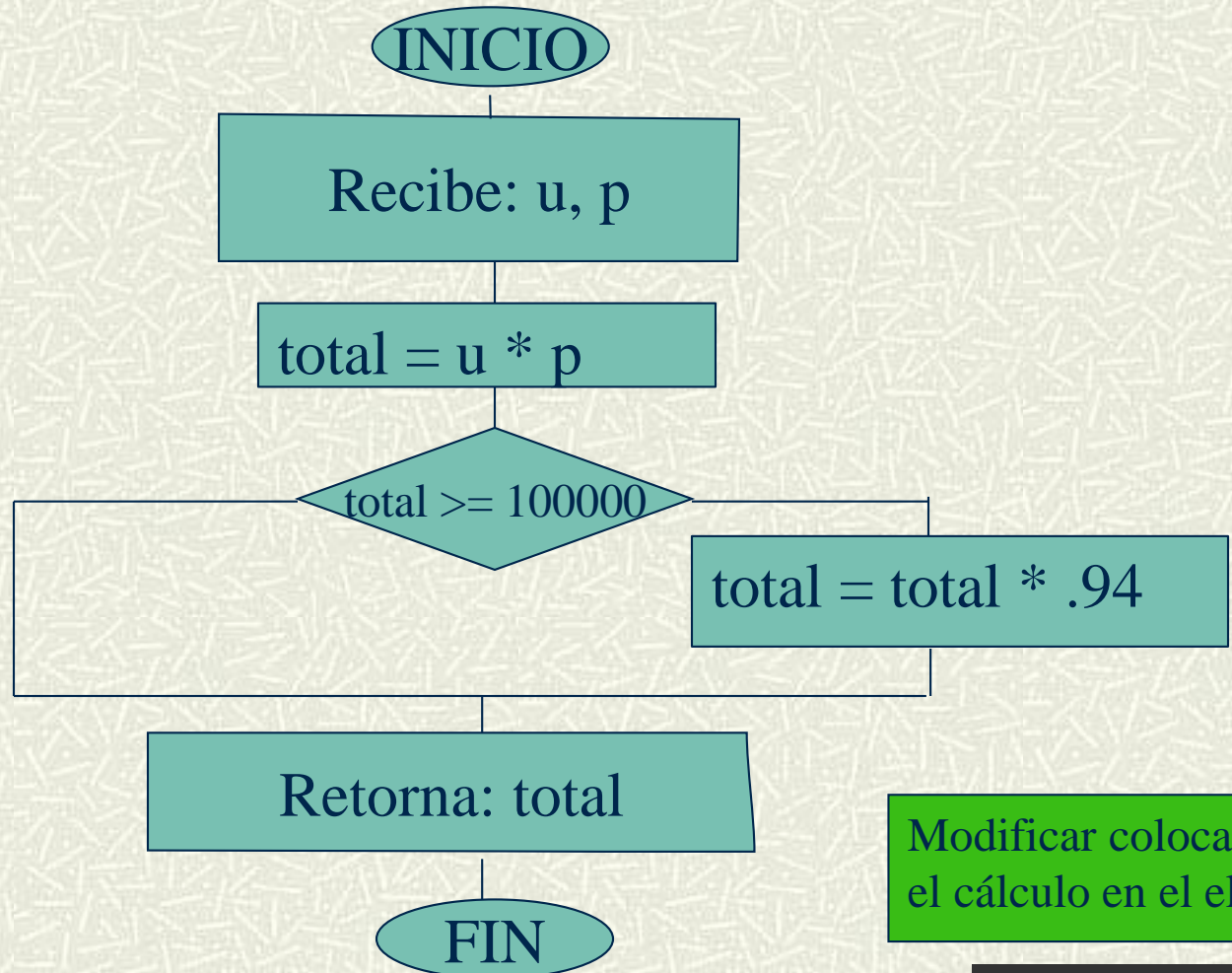


¿ corrida manual y pseudocódigo ?

Ejemplo: En el if hay instrucciones solo cuando la condición es verdadera o solo cuando es falsa. La función recibe las unidades compradas y el precio por unidad. Si el valor total es de 100000 o más se hace un descuento del 6%.

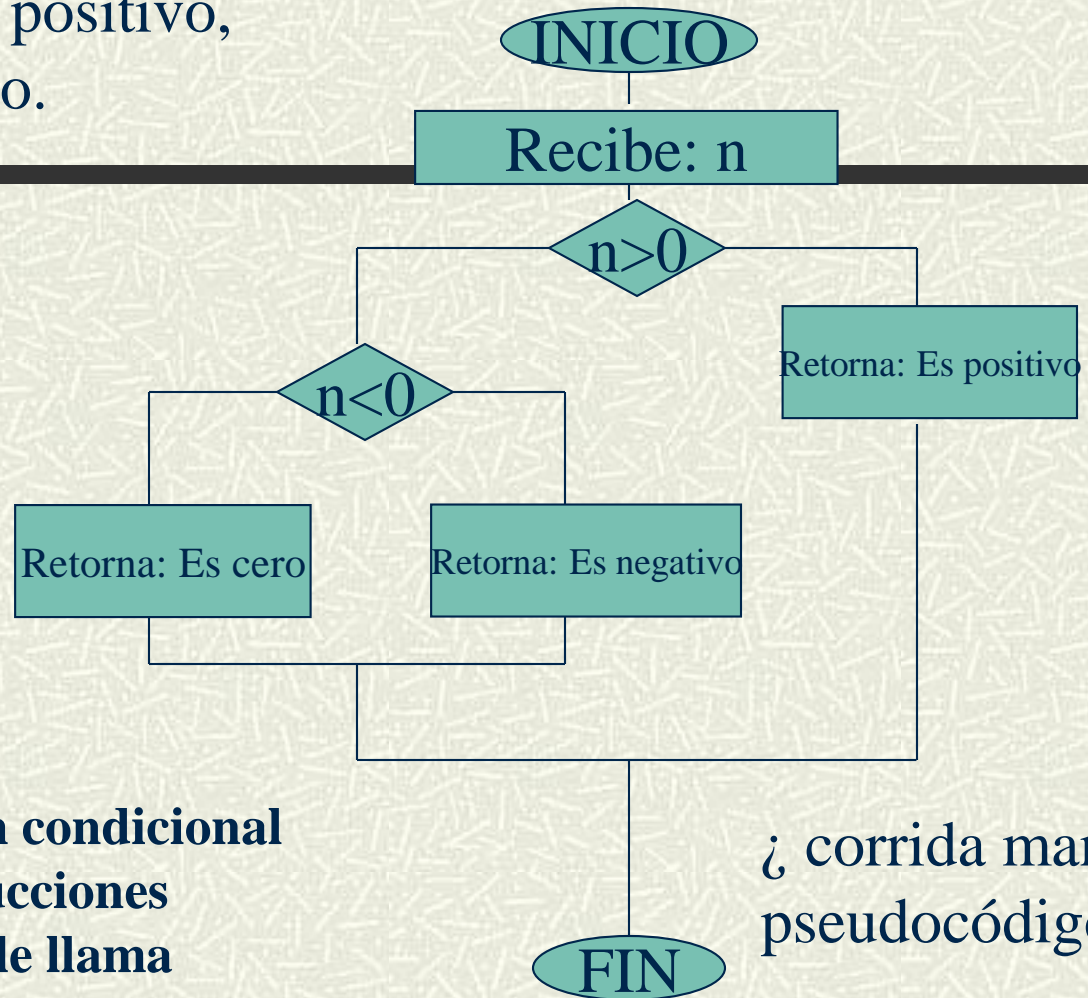
¿ Qué sabemos aquí
con respecto al valor
de la variable total ?

¿ corrida manual
y pseudocódigo ?



Modificar colocando
el cálculo en el else

Ejemplo: recibe un número y retorna si es positivo, negativo o cero.



Dentro de una instrucción condicional pueden haber otras instrucciones condicionales a lo cual se le llama condicional anidada o if anidado

¿ corrida manual y pseudocódigo ?

Codificar usando únicamente un return

Prueba de funciones:

■ FORMA 1:

- Hacer un programa principal para solicitar los datos de entrada y llamar a la función
- Hacer un programa principal que llame a la función con valores constantes, no se le piden al usuario

■ FORMA 2: Si no hacemos la sección del programa principal, las pruebas las podemos hacer desde el modo comando

- Traemos el programa fuente: File / Open / F5 y a partir de este momento las funciones quedan disponibles para ser usadas en dicho modo

EJERCICIOS (pseudocódigo y corrida manual)

- 1. Haga una función que reciba un número positivo entero y retorne el valor “El número es par” cuando sea par y el valor “El número es impar” cuando sea impar. Un número es par cuando el residuo de su división entera entre 2 es 0. En caso de que el número recibido no cumpla con las restricciones retornar el valor “Error”. Ejemplos del funcionamiento:

```
>>> par_impar(25)
El número es impar
>>> par_impar(200)
El número es par
>>> par_impar(-1)
Error
```

- # 2. Haga una función que reciba tres datos, los dos primeros son números y el tercero va a ser un string conteniendo un código de operación matemática (“+”, “-”, “/”, “*”) que se les va a aplicar. Retorne el resultado. Ejemplo del funcionamiento

```
>>> minicalculadora(5, 9, “*”)
45
```

Si el código de operación no es válido retorne un mensaje de error

```
>>> minicalculadora(5, 9, “x”)
Error: código de operación debe ser +, -, / o *
```

En las divisiones el divisor debe ser diferente de cero, de lo contrario retorne un mensaje de error

```
>>> minicalculadora(5, 0, “/”)
Error: el divisor debe ser diferente de cero
```

Uso del **pass**

- En caso de que se quiera tener un bloque sin instrucciones, se puede usar la instrucción **pass**, la cual indica que continúe la ejecución

```
if x >= abs(v):
```

```
    pass
```

```
else:
```

```
    instrucciones
```

```
if x >= abs(v):
```

```
    instrucciones
```

```
else:
```

```
    pass
```

```
if x >=abs(v):
```

```
    instrucciones
```

En este último caso el **else:** no tiene instrucciones: puede ponerse **pass** o quitar el **else:**

OPERADORES LÓGICOS

- # Usados para formar condiciones compuestas
 - Evaluar dos o más condiciones simples: con el uso de operadores lógicos. La evaluación de una condición compuesta va a ser True o False.

- # Hay 3 operadores lógicos
 - and
 - or
 - not

Operador lógico: **and** (y lógico)

- Para que el resultado de la expresión sea verdadero, todas sus condiciones deben ser verdaderas
- Si alguna condición es falsa la expresión es falsa

Ejemplo:

```
if x > 0 and y < 10:  
    instrucciones
```

Esta condición compuesta es verdadera solo cuando $x > 0$ sea verdadera y que $y < 10$ también sea verdadera

Operador lógico: **or** (o lógico)

- Para que el resultado de la expresión sea verdadero, al menos una condición debe ser verdadera
- Si todas las condiciones son falsas la expresión es falsa

Ejemplo:

```
if n == 1 or n > 5:
```

```
    instrucciones
```

Esta condición es verdadera cuando

$n == 1$ sea verdadera (n sea igual a 1) o que

$n > 5$ sea verdadera (n sea mayor a 5)

Operador lógico: **not** (no lógico)

- Este operador niega el resultado de la evaluación de una expresión booleana
- Si el resultado de la expresión booleana es verdadero antes de aplicar el **not**, entonces el resultado después de aplicar el **not** es falso
- Si el resultado de la expresión booleana es falso antes de aplicar el **not**, entonces el resultado después de aplicar el **not** es verdadero



Ejemplo:

if not (x > y):

¿ cuándo esta condición es verdadera ?

¿ cuándo esta condición es falsa ?

Orden de evaluación de operadores lógicos

not, and, or

Ejemplo: función para calcular área de un rectángulo.

Restricciones: validar datos de entrada

- Vamos a dar como un hecho que las entradas son numéricas
 - Validar las restricciones
 - Base y altura mayores que cero
 - Si las restricciones no se cumplen la salida va a ser un mensaje de error: “Error: los valores deben ser mayores a cero”

William Mata Rodriguez

```
def area_rectangulo(base, altura):
```

```
    """Función para calcular área de un rectángulo.
```

```
    Entradas: base y altura.
```

```
    Salidas: área del rectángulo.
```

```
    Restricciones: la base y la altura deben ser números mayores que
    cero."""
```

```
    if base > 0 and altura > 0:
```

```
        area = base * altura
```

```
        return area
```

```
    else:
```

```
        return "Error: los valores deben ser mayores a cero"
```

```
# programa principal
```

```
b = int(input("Base: "))
```

```
a = int(input("Altura: "))
```

```
area=area_rectangulo(b,a)
```

```
print(area)
```

Ejemplo: función para formar un número a partir de 3 dígitos

Restricciones: validar datos de entrada

- Vamos a dar como un hecho que las entradas son números enteros pero vamos a poner las siguientes restricciones:
 - Cada dígito va a estar en el rango entre 0 y 9
 - Si las restricciones no se cumplen la salida va a ser un mensaje de error

William Mata Rodriguez

```
def formar_numero(d1, d2, d3):
```

```
    """Función para formar un número a partir de tres dígitos.
```

```
    Entradas: tres dígitos, el primero corresponde a las unidades,  
              el segundo corresponde a las decenas y el tercero a las centenas.
```

```
    Salidas: número que representa los dígitos de entrada.
```

```
    Restricciones: cada dígito va a estar en el rango de 0 a 9."""
```

```
    if d1 >=0 and d1 <= 9:
```

```
        if d2 >= 0 and d2 <= 9:
```

```
            if d3 >= 0 and d3 <= 9:
```

```
                num = d1 + d2*10 + d3*100
```

```
                return num
```

```
            else:
```

```
                return "Error en el tercer dígito"
```

```
        else:
```

```
            return "Error en el segundo dígito"
```

```
    else:
```

```
        return "Error en el primer dígito"
```

```
# programa principal
```

```
digito1=int(input("Dígito 1: "))
```

```
digito2=int(input("Digito 2: "))
```

```
digito3=int(input("Digito 3: "))
```

```
print("Resultado:",formar_numero(digito1, digito2, digito3))
```

Condicionales ligadas **elif**

- # Se pueden usar cuando hay mas de dos alternativas que deben verificarse
- # **elif** es una abreviación de las instrucciones

else:

if

Estructura general:

if condición:

Instrucciones

elif condición:

Instrucciones

elif condición:

Instrucciones

else:

Instrucciones

El último **else** es opcional.

- # Cada condición se verifica en orden
- # Si la primera condición es falsa, se verifica la siguiente y así sucesivamente
- # Si alguna condición es verdadera se ejecuta el bloque de instrucciones correspondiente
- # Aunque varias condiciones pueden ser verdaderas, solamente se ejecutará la primera que lo sea

Ejemplo: función para determinar estado de aprobación

- # Se requiere determinar el estado de aprobación de un estudiante en un curso. La función va a recibir una nota que va a ser un número entero. La función debe validar que esa nota este en el rango de 0 a 100, de lo contrario la salida va a ser “Nota no es válida”.
- # Las posible salidas si la entrada esta bien serán:
 - “Aprobado” si la nota esta en el rango de 70 a 100
 - “Reposición” si la nota esta en el rango de 60 a 69
 - “Reprobado” si la nota es menor que 60.

Paso 1: entender el problema

El problema consiste en determinar cuál de los tres posibles estados se debe asignar al estudiante.

Definir

entradas

salidas

restricciones



Paso 2: desarrollar (pensar) un algoritmo

Hacerlo en pseudocódigo tipo “Python”.

Paso 3: codificar el algoritmo

Al usar un pseudocódigo tipo Python para desarrollar el algoritmo nos queda listo el programa en el lenguaje de programación

William Mata Rodriguez

```
def determinar_estado_aprobacion(nota):
```

```
    """Función para determinar el estado de aprobación de un estudiante.
```

```
    Entradas: nota.
```

```
    Salidas: Aprobado si la nota esta en el rango de 70 a 100.
```

```
             Reposición si la nota esta en el rango de 60 a 69
```

```
             Reprobado si la nota esta en el rango de 0 a 59.
```

```
    Restricciones: la nota es un número entero en el rango de 0 a 100."""
```

```
    if nota >= 70 and nota <= 100:
```

```
        resultado = "Aprobación"
```

```
    elif nota >= 60 and nota <= 69:
```

```
        resultado = "Reposición"
```

```
    elif nota <=59 and nota >= 0:
```

```
        resultado = "Reprobado"
```

```
    else:
```

```
        resultado = "Nota no es válida"
```

```
    return resultado
```

```
# programa principal
```

```
nota=int(input("Nota: "))
```

```
print(determinar_estado_aprobacion(nota))
```


Paso 4: Probar y evaluar el programa

- Use diferentes valores para probar que todas las condiciones son manejadas por el programa

RECORDAR: EL PROGRAMA DEBE SER GENERALISTA,
FUNCIONAR PARA CUALQUIER CONJUNTO DE
VALORES



*"... aprendiendo:
estudiar y
practicar*

..."

