

CAPÍTULO 7. CONDICIONALES MÚLTIPLES Y OPERADORES LÓGICOS

1. Introducción

En el capítulo anterior aprendimos a utilizar las estructuras condicionales `if` e `if-else`.

Estas estructuras permiten que un programa tome decisiones sencillas.

Por ejemplo:

- Determinar si un alumno ha aprobado o suspendido.
- Comprobar si una persona es mayor de edad.
- Verificar si una temperatura es positiva o negativa.
- Comprobar si una contraseña es correcta.

Observa el siguiente ejemplo:

```
edad = int(input("Edad: "))

if edad >= 18:
    print("Mayor de edad")

else:
    print("Menor de edad")
```

Este programa solo tiene dos posibles respuestas:

- Mayor de edad.
- Menor de edad.

Sin embargo, muchos problemas reales requieren más de dos alternativas.

Por ejemplo:

- Clasificar una nota como suspenso, aprobado, notable o sobresaliente.
- Clasificar una persona como menor de edad, adulta o jubilada.
- Determinar distintos descuentos según la edad.
- Clasificar una temperatura como fría, templada o calurosa.

Para resolver este tipo de problemas necesitaremos estructuras condicionales más avanzadas.

Además, aprenderemos a combinar varias condiciones mediante operadores lógicos.

En este capítulo estudiaremos:

- La estructura if-elif-else.
- Los operadores lógicos and, or y not.
- Los condicionales anidados.
- Los errores más frecuentes al trabajar con condiciones.

Estas herramientas permitirán que nuestros programas sean capaces de tomar decisiones mucho más complejas.

2. Repaso de la estructura if-else

Antes de estudiar nuevas estructuras es conveniente recordar cómo funciona un condicional sencillo.

La sintaxis general es:

```
if condicion:  
    instrucciones
```

```
else:  
    instrucciones
```

Python evalúa la condición.

Si la condición es verdadera (True):

Ejecuta el bloque if

Si la condición es falsa (False):

Ejecuta el bloque else

Ejemplo

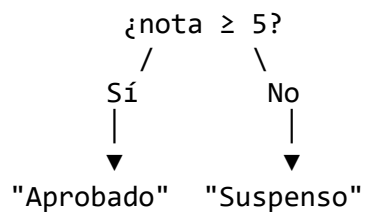
Determinar si una nota está aprobada.

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

```
if nota >= 5:  
    print("Aprobado")
```

```
else:  
    print("Suspenso")
```

Diagrama de flujo



Limitación del if-else

El problema de esta estructura es que solo permite elegir entre dos alternativas.

Observa la siguiente situación:

Suspenso
Aprobado
Notable
Sobresaliente

Ahora tenemos cuatro posibilidades.

La estructura if-else ya no es suficiente.

Necesitamos una nueva herramienta.

3. La estructura if-elif-else

La palabra reservada elif es una abreviatura de:

else if

Permite comprobar varias condiciones diferentes dentro de la misma estructura.

Su sintaxis general es:

```
if condicion1:  
    instrucciones
```

```
elif condicion2:  
    instrucciones
```

```
elif condicion3:  
    instrucciones
```

```
else:  
    instrucciones
```

Funcionamiento

Python evalúa las condiciones de arriba hacia abajo.

Cuando encuentra una condición verdadera:

1. Ejecuta ese bloque.
2. Ignora el resto de condiciones.
3. Continúa con el programa.

Por este motivo el orden de las condiciones es muy importante.

Primer ejemplo completo

Problema

Clasificar una nota.

Clasificación

Nota	Resultado
Menor que 5	Suspenso
Entre 5 y 7	Aprobado
Entre 7 y 9	Notable
Mayor o igual que 9	Sobresaliente

Análisis

Datos de entrada

Una nota.

Datos de salida

La calificación correspondiente.

Algoritmo

1. Leer la nota.
 2. Comprobar si es menor que 5.
 3. Si no lo es, comprobar si es menor que 7.
 4. Si no lo es, comprobar si es menor que 9.
 5. En cualquier otro caso será sobresaliente.
-

Seudocódigo

Leer nota

Si nota < 5

 Escribir "Suspenso"

SiNoSi nota < 7

 Escribir "Aprobado"

SiNoSi nota < 9

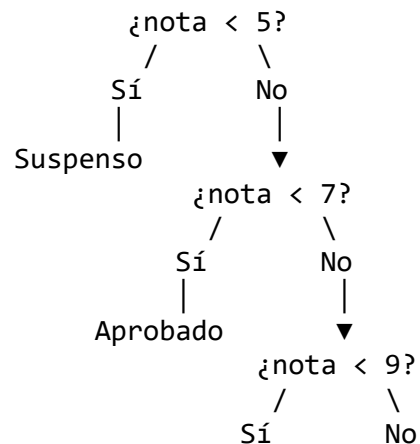
 Escribir "Notable"

SiNo

 Escribir "Sobresaliente"

FinSi

Diagrama de flujo



Notable Sobresaliente

Python

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

if nota < 5:
    print("Suspenso")

elif nota < 7:
    print("Aprobado")

elif nota < 9:
    print("Notable")

else:
    print("Sobresaliente")
```

Ejemplo de ejecución

Entrada:

8

Salida:

Notable

4. Importancia del orden de las condiciones

Cuando utilizamos una estructura if-elif-else, el orden de las condiciones es fundamental.

Python evalúa las condiciones de arriba hacia abajo.

En cuanto encuentra una condición verdadera:

Ejecuta ese bloque y abandona la estructura.

Las condiciones restantes ya no se comprueban.

Ejemplo correcto

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

```
if nota < 5:  
    print("Suspenso")
```

```
elif nota < 7:  
    print("Aprobado")
```

```
elif nota < 9:  
    print("Notable")
```

```
else:  
    print("Sobresaliente")
```

Si:

```
nota = 8
```

Python realiza las siguientes comprobaciones:

Condición	Resultado
nota < 5	False
nota < 7	False
nota < 9	True

Por tanto muestra:

Notable

Ejemplo incorrecto

Observa ahora el siguiente programa:

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

```
if nota < 10:  
    print("Aprobado")
```

```
elif nota < 5:  
    print("Suspenso")
```

Supongamos:

```
nota = 4
```

La primera condición:

```
nota < 10
```

es verdadera.

Python ejecuta:

```
print("Aprobado")
```

e ignora el resto de condiciones.

La salida será:

Aprobado

aunque la nota sea un 4.

¿Por qué ocurre?

Porque Python nunca llega a evaluar la segunda condición.

Cuando encuentra una condición verdadera abandona la estructura.

Regla práctica

Cuando utilices varias condiciones:

1. Piensa en todos los casos posibles.
 2. Ordena cuidadosamente las condiciones.
 3. Comprueba que ninguna condición impida evaluar las siguientes.
-

Ejemplo

Clasificación por edades:

- Menor de edad: menos de 18 años.
- Adulto: entre 18 y 64 años.
- Jubilado: 65 años o más.

```
edad = int(input("Edad: "))
```

```
if edad < 18:  
    print("Menor de edad")
```

```
elif edad < 65:  
    print("Adulto")
```

```
else:  
    print("Jubilado")
```

Observa que las condiciones están ordenadas de forma lógica y ninguna impide evaluar las siguientes.

Resumen de este bloque

En esta primera parte hemos aprendido:

- Las limitaciones de la estructura if-else.
- La estructura if-elif-else.
- Cómo funcionan los condicionales múltiples.
- La importancia del orden de las condiciones.
- Cómo clasificar información utilizando varias alternativas.

En el siguiente bloque estudiaremos los operadores lógicos:

- and
- or
- not

que nos permitirán combinar varias condiciones dentro de un mismo programa.

5. Operadores lógicos

Hasta ahora hemos trabajado con condiciones sencillas.

Por ejemplo:

edad >= 18

o

nota >= 5

Sin embargo, en muchos problemas necesitamos comprobar varias condiciones al mismo tiempo.

Por ejemplo:

- Tener más de 18 años y disponer de carnet de conducir.
- Obtener una nota superior a 8 y una asistencia superior al 90 %.
- Ser sábado o domingo.

- No haber introducido la contraseña correcta.

Para resolver este tipo de situaciones utilizaremos los operadores lógicos.

Los operadores lógicos permiten combinar varias condiciones para construir expresiones más complejas.

Python dispone de tres operadores lógicos principales:

Operador	Significado
and	Y
or	O
not	NO

5.1 Operador and

El operador and significa:

Y

Se utiliza cuando todas las condiciones deben cumplirse simultáneamente.

Ejemplo cotidiano

Para conducir un coche normalmente es necesario:

- Tener edad suficiente.
- Disponer de permiso de conducir.

Las dos condiciones deben cumplirse.

Ejemplo en Python

```
edad = 20 carnet = True
```

```
if edad >= 18 and carnet == True: print("Puede conducir")
```

```
---
```

```
## ¿Cómo funciona?
```

La expresión:

`edad >= 18 and carnet == True`

solo será verdadera cuando ambas condiciones sean verdaderas.

Problema resuelto

Problema

Determinar si un alumno puede obtener una beca.

Requisitos

- Nota media igual o superior a 8.
 - Asistencia igual o superior al 90%.
-

Algoritmo

4. Leer la nota.
 5. Leer la asistencia.
 6. Comprobar ambas condiciones.
 7. Mostrar el resultado.
-

Seudocódigo

Leer nota Leer asistencia

Si $\text{nota} \geq 8$ Y $\text{asistencia} \geq 90$

 Escribir "Beca concedida"

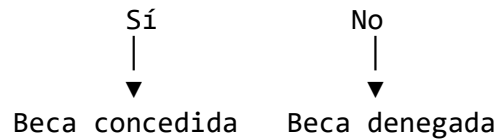
SiNo

 Escribir "Beca denegada"

FinSi

Diagrama de flujo

 {nota $\geq$ 8 y asistencia $\geq$ 90?
 /
 \



Python

```
nota = float(input("Nota:")) asistencia = float(input("Asistencia:"))  
if nota >= 8 and asistencia >= 90: print("Beca concedida")  
else: print("Beca denegada")
```

Ejemplo de ejecución

Entrada:

Nota: 8.5
Asistencia: 95

Salida:

Beca concedida

5.2 Operador or

El operador or significa:

0

Se utiliza cuando basta con que una de las condiciones sea verdadera.

Ejemplo cotidiano

Un alumno puede acceder a una actividad si:

- Es mayor de edad.
- O presenta una autorización familiar.

Solo es necesario cumplir una de las condiciones.

Ejemplo en Python

```
edad = 16 autorizacion = True
```

```
if edad >= 18 or autorizacion == True: print("Puede participar")
```

```
---
```

```
## Problema resuelto
```

```
### Problema
```

```
Determinar si un día pertenece al fin de semana.
```

```
---
```

```
### Algoritmo
```

1. Leer el día.
2. Comprobar si es sábado o domingo.
3. Mostrar el resultado.

```
---
```

```
### Seudocódigo
```

```
Leer día
```

```
Si día = "sábado" O día = "domingo"
```

```
    Escribir "Fin de semana"
```

```
SiNo
```

```
    Escribir "Día lectivo"
```

```
FinSi
```

Python

```
dia = input("Día:")
```

```
if dia == "sábado" or dia == "domingo": print("Fin de semana")
```

```
else: print("Día lectivo")
```

Ejemplos de ejecución

Entrada:

sábado

Salida:

Fin de semana

Entrada:

miércoles

Salida:

Día lectivo

5.3 Operador not

El operador not significa:

NO

Su función consiste en invertir el valor de una condición.

Ejemplo

Si:

llueve = True

entonces:

not llueve

vale:

False

Si:

```
llueve = False
```

entonces:

```
not llueve
```

vale:

```
True
```

Problema resuelto

Problema

Determinar si podemos salir a pasear.

Condición

Podemos salir si no está lloviendo.

Python

```
llueve = False
```

```
if not llueve: print("Podemos salir")
```

```
else: print("Mejor nos quedamos en casa")
```

```
---
```

```
## Resultado
```

```
Podemos salir
```

6. Tablas de verdad

Las tablas de verdad permiten comprender exactamente cómo funcionan los operadores lógicos.

Tabla de verdad de and

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

Interpretación

La expresión:

A and B

solo es verdadera cuando:

A y B son verdaderas simultáneamente.

Tabla de verdad de or

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

Interpretación

La expresión:

A or B

es verdadera cuando al menos una condición es verdadera.

Tabla de verdad de not

A	not A
True	False
False	True

Interpretación

El operador:

not A

invierte el valor lógico de la condición.

Resumen de este bloque

En esta segunda parte hemos aprendido:

- Qué son los operadores lógicos.
- Cómo utilizar and.
- Cómo utilizar or.
- Cómo utilizar not.
- Cómo interpretar las tablas de verdad.
- Cómo combinar varias condiciones dentro de un mismo programa.

En el siguiente bloque estudiaremos los **condicionales anidados**, realizaremos el seguimiento paso a paso de programas con múltiples condiciones y resolveremos problemas más complejos.

7. Condicionales anidados

Hasta ahora hemos utilizado estructuras condicionales simples y múltiples.

Sin embargo, en algunos problemas una decisión depende de otra decisión previa.

En estos casos utilizaremos:

Condicionales anidados

Un condicional anidado es una estructura condicional situada dentro de otra estructura condicional.

Ejemplo cotidiano

Supongamos que queremos clasificar una persona según su edad.

Las categorías son:

- Menor de edad: menos de 18 años.
- Adulto: entre 18 y 64 años.
- Jubilado: 65 años o más.

Podríamos resolver el problema mediante un condicional múltiple.

Sin embargo, también podemos pensar así:

8. ¿Es mayor de edad?
9. Si es mayor de edad, ¿tiene al menos 65 años?

Observa que la segunda pregunta solo tiene sentido si la primera respuesta es afirmativa.

Estructura general

```
if condicion1:

    if condicion2:
        instrucciones

    else:
        instrucciones

else:
    instrucciones
```

Problema resuelto

Problema

Clasificar una persona según su edad.

Algoritmo

10. Leer la edad.
11. Comprobar si es mayor o igual que 18.
12. Si es menor de 18 años, mostrar “Menor de edad”.
13. Si tiene 18 años o más:
 - Comprobar si tiene 65 años o más.

- Si es así, mostrar “Jubilado”.
 - En caso contrario, mostrar “Adulto”.
-

Seudocódigo

Leer edad

Si edad $\geq$ 18

 Si edad $\geq$ 65

 Escribir "Jubilado"

 SiNo

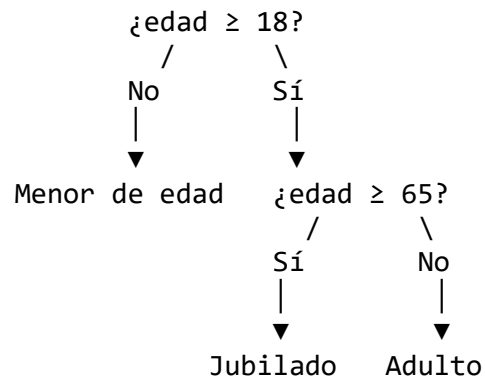
 Escribir "Adulto"

SiNo

 Escribir "Menor de edad"

FinSi

Diagrama de flujo



Python

```
edad = int(input("Edad: "))
```

```
if edad >= 18:
```

```
    if edad >= 65:
        print("Jubilado")
```

```
    else:
        print("Adulto")

else:
    print("Menor de edad")
```

Ejemplos de ejecución

Entrada:

15

Salida:

Menor de edad

Entrada:

40

Salida:

Adulto

Entrada:

70

Salida:

Jubilado

8. Problemas resueltos paso a paso

A continuación resolveremos varios problemas completos utilizando condicionales múltiples y operadores lógicos.

Problema 1. Clasificación de notas

Enunciado

Clasificar una nota según la siguiente tabla:

Nota	Calificación
Menor que 5	Suspenso
Entre 5 y 7	Aprobado
Entre 7 y 9	Notable
Mayor o igual que 9	Sobresaliente

Algoritmo

14. Leer la nota.
 15. Comprobar los distintos intervalos.
 16. Mostrar la calificación correspondiente.
-

Python

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

if nota < 5:
    print("Suspenso")

elif nota < 7:
    print("Aprobado")

elif nota < 9:
    print("Notable")

else:
    print("Sobresaliente")
```

Problema 2. Sistema de becas

Enunciado

Conceder una beca si:

- Nota ≥ 8

- Asistencia ≥ 90 %
-

Algoritmo

1. Leer nota.
 2. Leer asistencia.
 3. Comprobar ambas condiciones.
 4. Mostrar el resultado.
-

Python

```
nota = float(input("Nota: "))
asistencia = float(input("Asistencia: "))

if nota >= 8 and asistencia >= 90:
    print("Beca concedida")

else:
    print("Beca denegada")
```

Problema 3. Acceso a una atracción

Enunciado

Una persona puede acceder si:

- Tiene al menos 12 años.
 - Mide al menos 1.40 m.
-

Python

```
edad = int(input("Edad: "))
altura = float(input("Altura: "))

if edad >= 12 and altura >= 1.40:
    print("Puede subir")

else:
    print("No puede subir")
```

Problema 4. Acceso mediante autorización

Enunciado

Una persona puede acceder si:

- Tiene 18 años o más.
 - O dispone de autorización.
-

Python

```
edad = int(input("Edad: "))
autorizacion = input("¿Tiene autorización? ")

if edad >= 18 or autorizacion == "si":
    print("Acceso permitido")

else:
    print("Acceso denegado")
```

Problema 5. Clasificación de temperaturas

Enunciado

Clasificar una temperatura.

Temperatura	Clasificación
Menor de 10	Fría
Entre 10 y 25	Templada
Mayor que 25	Calurosa

Python

```
temperatura = float(input("Temperatura: "))

if temperatura < 10:
    print("Fría")

elif temperatura <= 25:
    print("Templada")
```

```
else:
    print("Calurosa")
```

9. Seguimiento de programas con condicionales

Una de las mejores formas de comprender un programa consiste en analizar paso a paso cómo se evalúan las condiciones.

Ejemplo

```
nota = 8

if nota < 5:
    print("Suspenso")

elif nota < 7:
    print("Aprobado")

elif nota < 9:
    print("Notable")

else:
    print("Sobresaliente")
```

Seguimiento

Paso	Condición evaluada	Resultado
1	nota < 5	False
2	nota < 7	False
3	nota < 9	True

Al encontrar una condición verdadera:

Python ejecuta el bloque correspondiente e ignora el resto de condiciones.

Resultado

Notable

Otro ejemplo

```
edad = 15
autorizacion = True
```

```
if edad >= 18 or autorizacion == True:
    print("Acceso permitido")
```

Seguimiento

Condición	Resultado
edad ≥ 18	False
autorizacion == True	True

Operador:

False or True

Resultado:

True

Salida:

Acceso permitido

Resumen de este bloque

En esta tercera parte hemos aprendido:

- Qué son los condicionales anidados.
- Cómo resolver problemas complejos mediante decisiones encadenadas.
- Cómo realizar el seguimiento paso a paso de un programa.
- Cómo analizar la evaluación de las condiciones.
- Cómo aplicar los operadores lógicos en situaciones reales.

En el siguiente y último bloque estudiaremos:

- Errores frecuentes.
- Actividades básicas.
- Actividades intermedias.
- Actividades de algoritmia.
- Actividades avanzadas.

- Reto final.
- Resumen del capítulo.

10. Errores frecuentes

Cuando comenzamos a trabajar con estructuras condicionales es normal cometer algunos errores.

Conocerlos nos ayudará a detectarlos y corregirlos con facilidad.

Error 1. Confundir = con ==

Es probablemente el error más frecuente entre los principiantes.

Incorrecto:

```
nota = 5
```

```
if nota = 5:  
    print("Correcto")
```

¿Qué ocurre?

El símbolo:

=

se utiliza para asignar valores.

Por ejemplo:

```
nota = 5
```

significa:

Guardar el valor 5 en la variable nota.

Forma correcta

Para comparar valores debemos utilizar:

==

Ejemplo:

```
nota = 5
```

```
if nota == 5:  
    print("Correcto")
```

Error 2. Orden incorrecto de las condiciones

Incorrecto:

```
nota = 4
```

```
if nota < 10:  
    print("Aprobado")  
  
elif nota < 5:  
    print("Suspenso")
```

¿Qué ocurre?

La primera condición:

```
nota < 10
```

es verdadera.

Por tanto Python muestra:

Aprobado

y nunca llega a comprobar la segunda condición.

Solución

Ordenar correctamente las condiciones.

```
if nota < 5:  
    print("Suspenso")  
  
elif nota < 10:  
    print("Aprobado")
```

Error 3. Utilizar varios if independientes

Incorrecto:

```
nota = 8
```

```
if nota >= 5:  
    print("Aprobado")
```

```
if nota >= 7:  
    print("Notable")
```

Resultado

Aprobado
Notable

¿Por qué ocurre?

Porque ambos if son independientes.

Python evalúa los dos.

Solución

Utilizar:

```
if - elif - else
```

Error 4. Condiciones imposibles

Incorrecto:

```
edad = 20
```

```
if edad < 10 and edad > 30:  
    print("Correcto")
```

¿Qué ocurre?

Ninguna persona puede tener simultáneamente:

menos de 10 años
y
más de 30 años

La condición siempre será falsa.

Error 5. Errores de indentación

Incorrecto:

```
if nota >= 5:  
print("Aprobado")
```

Correcto

```
if nota >= 5:  
    print("Aprobado")
```

Regla práctica

Cuando un programa no funcione correctamente:

5. Comprueba las condiciones.
 6. Comprueba los operadores lógicos.
 7. Comprueba la indentación.
 8. Comprueba el orden de los bloques.
-

11. Actividades básicas

Actividad 1

Solicita una nota y muestra:

- Suspenso.
 - Aprobado.
-

Actividad 2

Solicita una edad y determina si la persona es:

- Menor de edad.
 - Mayor de edad.
-

Actividad 3

Solicita una temperatura y muestra:

- Fría.
 - Templada.
 - Calurosa.
-

Actividad 4

Solicita un número y determina si es:

- Positivo.
 - Negativo.
 - Cero.
-

Actividad 5

Solicita dos números y muestra cuál es mayor.

Actividad 6

Solicita tres números y muestra cuál es el mayor.

12. Actividades intermedias

Actividad 7

Clasificar una nota como:

- Suspenso.
 - Aprobado.
 - Notable.
 - Sobresaliente.
-

Actividad 8

Determinar si un alumno obtiene una beca.

Condiciones:

- Nota ≥ 8 .
 - Asistencia $\geq 90\%$.
-

Actividad 9

Determinar si una persona puede acceder a una atracción.

Condiciones:

- Edad ≥ 12 .
 - Altura ≥ 1.40 m.
-

Actividad 10

Determinar si un usuario puede acceder a una plataforma.

Condiciones:

- Tener 18 años o más.
 - O disponer de autorización.
-

Actividad 11

Clasificar una persona como:

- Menor de edad.
 - Adulto.
 - Jubilado.
-

13. Actividades de algoritmia

Para las actividades anteriores:

9. Diseña el algoritmo.
10. Escribe el pseudocódigo.
11. Dibuja el diagrama de flujo.

12. Implementa la solución en Python.

Actividad 12

Clasificación de notas.

Actividad 13

Sistema de becas.

Actividad 14

Acceso a una atracción.

Actividad 15

Clasificación de edades.

14. Actividades avanzadas

Actividad 16

Diseña un sistema de descuentos.

Condiciones:

- Menores de 12 años: 50 %.
 - Mayores de 65 años: 30 %.
 - Resto: sin descuento.
-

Actividad 17

Diseña un sistema de acceso a una biblioteca.

Condiciones:

- Mayor de edad.
- O autorización familiar.

Actividad 18

Clasifica una temperatura en cinco categorías:

- Muy fría.
 - Fría.
 - Templada.
 - Calurosa.
 - Muy calurosa.
-

Actividad 19

Determina el precio de una entrada según:

- Edad.
 - Tipo de usuario.
-

Actividad 20

Diseña un sistema de clasificación deportiva.

15. Reto final

Gestión de alumnos

Diseña un programa que solicite:

- Nombre del alumno.
- Nota media.
- Porcentaje de asistencia.

El programa deberá:

Paso 1

Determinar la calificación:

Nota	Resultado
Menor que 5	Suspenso

Nota	Resultado
Entre 5 y 7	Aprobado
Entre 7 y 9	Notable
Mayor o igual que 9	Sobresaliente

Paso 2

Determinar si obtiene beca.

Condiciones:

- $\text{Nota} \geq 8$.
 - $\text{Asistencia} \geq 90\%$.
-

Paso 3

Mostrar:

Nombre del alumno
Calificación
Situación de la beca

Trabajo previo

Antes de programar:

13. Diseña el algoritmo.
 14. Escribe el pseudocódigo.
 15. Dibuja el diagrama de flujo.
 16. Implementa la solución en Python.
-

16. Resumen

En este capítulo hemos aprendido:

- La estructura if-elif-else.
- La importancia del orden de las condiciones.
- Los operadores lógicos and, or y not.
- Las tablas de verdad.
- Los condicionales anidados.

- Cómo combinar varias condiciones.
- Cómo resolver problemas mediante decisiones múltiples.
- Los errores más frecuentes al trabajar con condicionales.

Los condicionales permiten que un programa tome decisiones.

Sin embargo, hasta ahora todas las instrucciones se ejecutaban una única vez.

En el próximo capítulo estudiaremos los **bucles while**, una estructura que permitirá repetir instrucciones automáticamente mientras se cumpla una determinada condición.