SUBINDICADOR # 5

Integra los operadores lógicos y su aplicación en el diseño de condiciones compuestas, con la finalidad de emplearlos en el desarrollo de programas de control condicional.

1. Introducción:

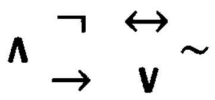
El estudiante debe diseñar condicionales basadas en las necesidades del ejercicio que simulan a las políticas o restricciones en un proceso de una organización; las condicionales compuestas otorgan un mejor criterio de solución a esos procesos complejos para asegurar la calidad de la información que se genera.

MÓDULO 3: Valores booleanos, Instrucciones if-elif-else, Bucles while y for, Control de flujo, Operaciones lógicas y bit a bit, Listas y arreglos.

2. Condicionales compuestas.

a. Manejo de los operadores lógicos and y or.

Los operadores lógicos nos permiten trabajar con valores de tipo booleano. Un valor booleano es un tipo que solo puede tomar valores True o False, por ello estos operadores nos permiten realizar diferentes operaciones con estos tipos, y su resultado será otro booleano.



Estos operadores son los siguientes:

Operador	Descripción	Ejemplo
and	Conjunción / Y lógico	**p** and **q**
or	Disyunción / O lógico	**p** or **q**
not	Negación	not **p**

Para operar los ejemplos se debe hacer uso de las tablas de verdad, que se muestran a continuación:

Conjunción(and)

p	q	p ^ q
V	V	V
V	F	F
F	V	F
F	F	F

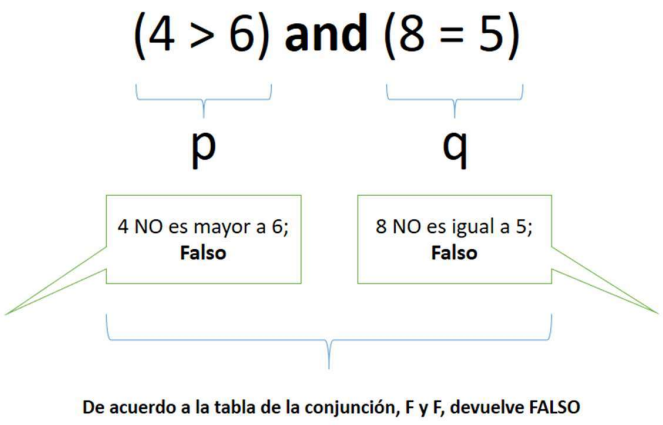
Disyunción(or)

p	q	p v q
V	V	V
V	F	V
F	V	V
F	F	F

b. Diseño de condicionales compuestas

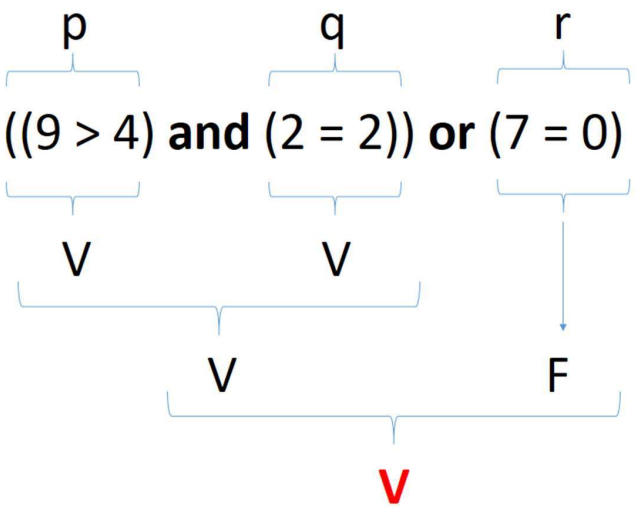
Una condicional compuesta está formada por dos o más condicionales simples, que se encuentran relacionadas por un operador lógico y que, para poder obtener un valor lógico como respuesta, se debe considerar lo indicado en las tablas de verdad para cada operador lógico (and / or).

Observar:



El resultado de evaluar esas dos condicionales simples es **FALSO.**

Observemos ahora este caso, siempre considerar las tablas de verdad para evaluar:



Cuando se tienen varias condicionales simples, debe tener en cuenta los paréntesis que agrupan a estas, caso contrario se opera de izquierda a derecha.

Si llevamos a Python ambos ejemplos, la evaluación sería directa:

```
print((4>6)and(8==5))
print(((9>4)and(2==2))or(7==0))
```

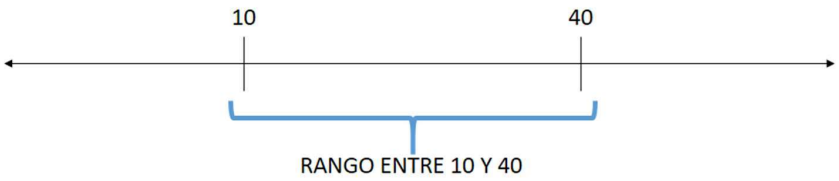
Al ejecutar, el resultado es el siguiente:

```
False
True
```

c. Definir estructuras de rangos y fuera de rango.

Cuando se habla de valores que están en un rango o que están fuera de rango, se generan condicionales compuesta que debemos saber diseñar y operar, de esa forma tenemos:

- **Rango**: está definido por dos valores que se ubican en una recta y que limitan a los números contenidos entre ellos, en el ejemplo vemos al número 10 y 40 que son los limitantes de un conjunto de valores.

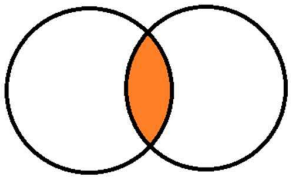


La condicional compuesta que acompaña al gráfico es la siguiente: siendo **Num** un valor cualquiera que se analiza.

(Num>10) and (Num<40)

Se puede considerar colocar el símbolo = dentro de las condiciones, pero esto depende del ejercicio, es decir si se toma el punto extremo; si ocurre ello se indica que el intervalo está cerrado en un punto y por lo tanto se debe colocar >= o <=.

Para interpretar el operador and, se puede indicar el siguiente gráfico que muestra un conjunto de valores definido entre un universo de datos:



- **Fuera de Rango**: está definido por dos valores que se ubican en una recta, pero que en esta oportunidad indican el límite superior e inferior de un conjunto de valores, finalmente forman dos grupos distintos.

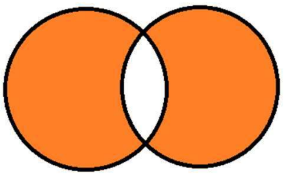
La condicional compuesta que acompaña al gráfico es la siguiente: siendo **Num** un valor cualquiera que se analiza.

(Num<10) or (Num>40)

Del mismo modo, se puede considerar colocar el símbolo = dentro de las condiciones, pero esto depende del ejercicio, es decir si se toma el

punto extremo menor o mayor; si ocurre ello se indica que el intervalo está cerrado en un punto y por lo tanto se debe colocar <= o >=.

Para interpretar el operador or, se puede indicar el siguiente gráfico que muestra un conjunto de valores definido entre un universo de datos:



d. Desarrollo de ejercicios tipo

Los ejercicios que veremos a continuación contienen condicionales compuestas, que se trabajan teniendo en cuenta la estructura de los ejercicios propuestos.

1. Problema Prg1

Se ingresa un número, mediante un algoritmo indique si representa una nota o no, use un mensaje.

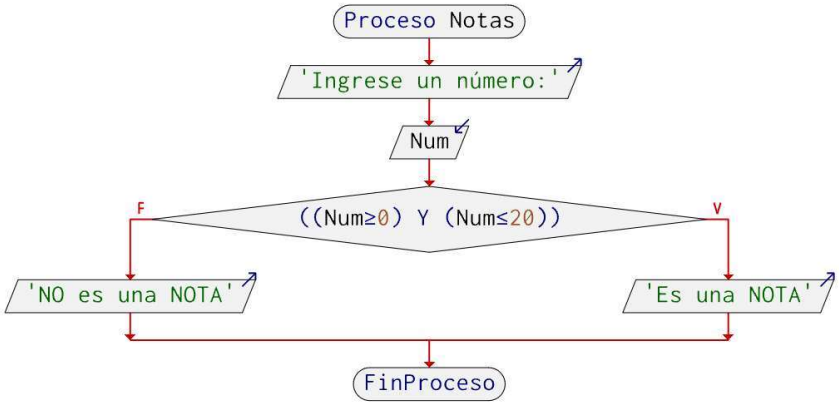
Solución:

a. Análisis.
   ¿Qué te piden que realices?
   Evaluar un número, eh indicar si fue una nota
   ¿Qué datos necesito conocer?
   El número.

b. Planteamiento Lógico.
   En este caso, debemos evaluar un número e indicar si fue una nota, para ello debemos recordar que las notas tienen un rango definido y es que están entre 0 y 20.

c. Diseño.

   Diagrama de Flujo:



d. Lenguaje de Programación.

```
print("Ingrese un número:")
Num=int(input())
if((Num>=0)and(Num<=20)):
    print("Es una NOTA")
else:
    print("NO es una NOTA")
```

Se coloca los símbolos = porque se debe incluir los valores extremos

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un número:
60
NO es una NOTA
```

Observación: Si no se ingresa un número que está en el rango indicado, simplemente se muestra el mensaje que no es una nota, porque no cumple la condición compuesta.

2. Problema Prg2

Diseñar un algoritmo que permita ingresar un número que representa una edad, luego indicar si este representa a la edad de un adulto, si se sabe que se considera adulto aquella persona que tiene entre 25 y 66 años, incluyendo solo el extremo menor.

Solución:

a. Análisis.

¿Qué te piden que realices?

Evaluar un número, eh indicar si es un adulto

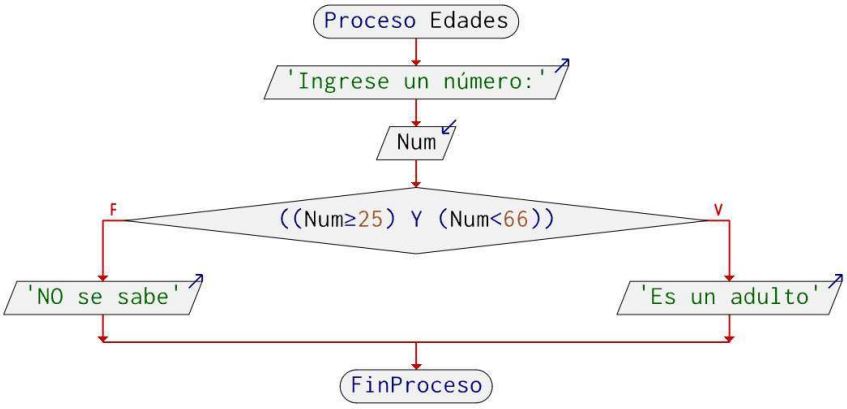
¿Qué datos necesito conocer?

El número.

b. Planteamiento Lógico.
Al igual que en el problema anterior, debe tener en cuenta el rango de valores, y otro dato a tener en cuenta es el extremo que no se considera, el de 66 años (no debe colocar el símbolo igual).

c. Diseño.

Diagrama de Flujo:



d. Lenguaje de Programación.

```
print("Ingrese un número:")
Num=int(input())
if((Num>=25)and(Num<66)):
    print("Es un adulto")
else:
    print("NO se sabe")
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un número:
45
Es un adulto
```

3. Problema Prg3

Diseñar un algoritmo que permita ingresar un número positivo, luego indicar si este no es un número de 3 cifras; use un mensaje.

Solución:

a. Análisis.

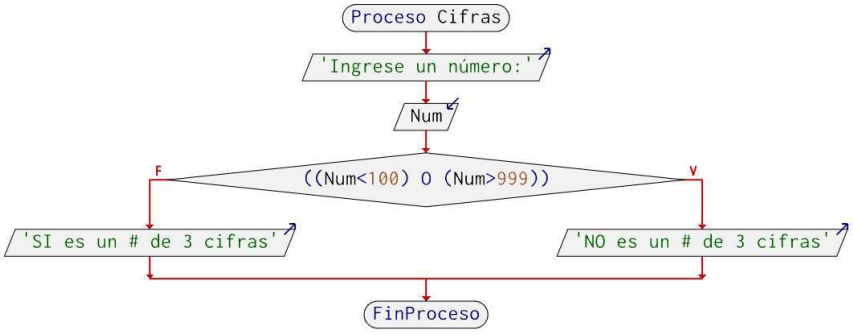
¿Qué te piden que realices?
Indicar si el número ingresado no es de 3 cifras
¿Qué datos necesito conocer?
El número

b. Planteamiento Lógico.

Para este ejercicio se debe tener en cuenta, todo lo que no se encuentre en el rango de un número de 3 cifras, es decir números menores de 100 y mayores de 999.

c. Diseño.

Diagrama de Flujo:



d. Lenguaje de Programación.

```
print("Ingrese un número:")
Num=int(input())
if((Num<100)or(Num>999)):
    print("NO es un # de 3 cifras")
else:
    print("SI es un # de 3 cifras")
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese un número:
3456
NO es un # de 3 cifras
```

Observemos esta otra variante, que considera a los números positivos y negativos:

```
print("Ingrese un número:")
Num=int(input())
if not((Num>=100)and(Num<=999))and not((Num>=-999)and(Num<=-100)):
    print("NO es un # de 3 cifras")
else:
    print("SI es un # de 3 cifras")
```

Esta solución resulta interesante, ya que se está usando operadores not, para negar rangos e intersectar los valores con el and para la respuesta final.

4. Problema Prg4

Ingresar una letra, luego indicar mediante un algoritmo si se trató de una vocal.

Solución:

a. Análisis.
¿Qué te piden que realices?
Indicar si el valor ingresado es una vocal.
¿Qué datos necesito conocer?
La letra.

b. Planteamiento Lógico.
Se tiene que realizar 5 condiciones simples y unirlas con operadores or; es la forma más lógica de solucionarlo; mientras que no trabajemos aun con listas en Python.

c. Diseño.
Diagrama de Flujo: