

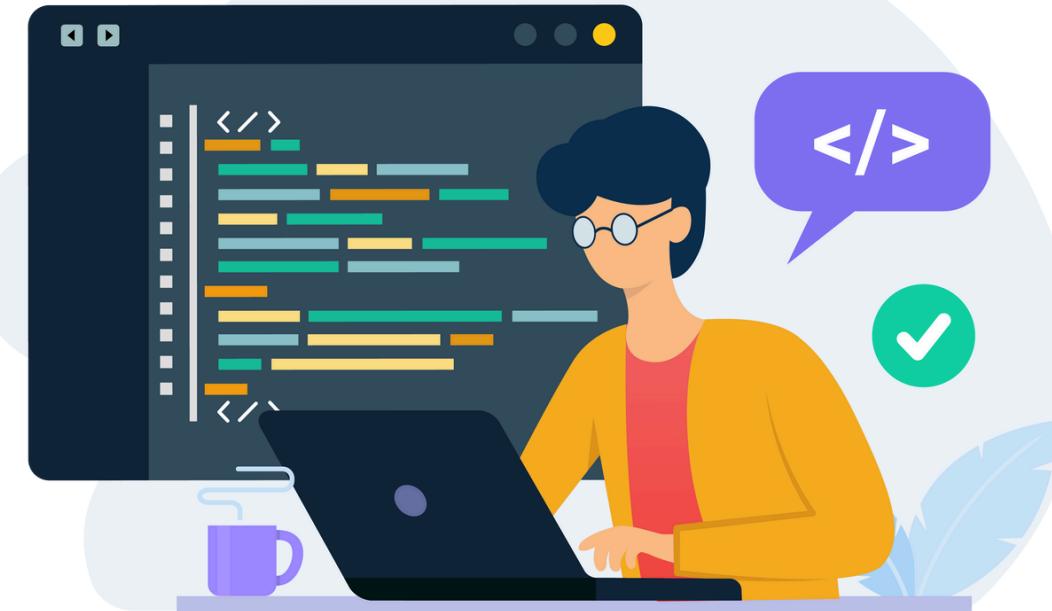
PENSANDO COMO UN PROGRAMADOR: INTRODUCCIÓN A LA LÓGICA DE PROGRAMACIÓN UTILIZANDO PYTHON

MASTER CLASS



De la experiencia a la abstracción: Filosofía aplicada a la lógica y programación

PROF: KEVIN TUPAC AGÜERO



INTRODUCCION

¿ALGUNA VEZ TE HAS PREGUNTADO CÓMO RESOLVEMOS PROBLEMAS EN LA VIDA COTIDIANA? DESDE DECIDIR QUÉ COMER HASTA RESOLVER UN CUBO RUBIK, USAMOS NUESTRA MENTE DE FORMAS SORPRENDENTES. PERO, ¿QUÉ PASA CUANDO QUEREMOS QUE UNA MÁQUINA NOS AYUDE? AQUÍ ES DONDE ENTRA LA MAGIA DE APRENDER A PENSAR COMO PROGRAMADORES.”

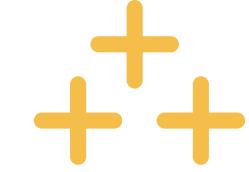


OBJETIVO

¿Qué valores? 

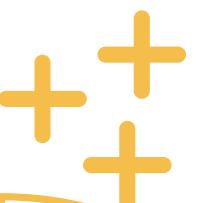
OBJETIVO GENERAL:

- BRINDAR A LOS ESTUDIANTES LOS FUNDAMENTOS DE LA LÓGICA DE PROGRAMACIÓN, UTILIZANDO PYTHON COMO HERRAMIENTA PARA RESOLVER PROBLEMAS DEL MUNDO REAL Y DESARROLLAR UN PENSAMIENTO ESTRUCTURADO.

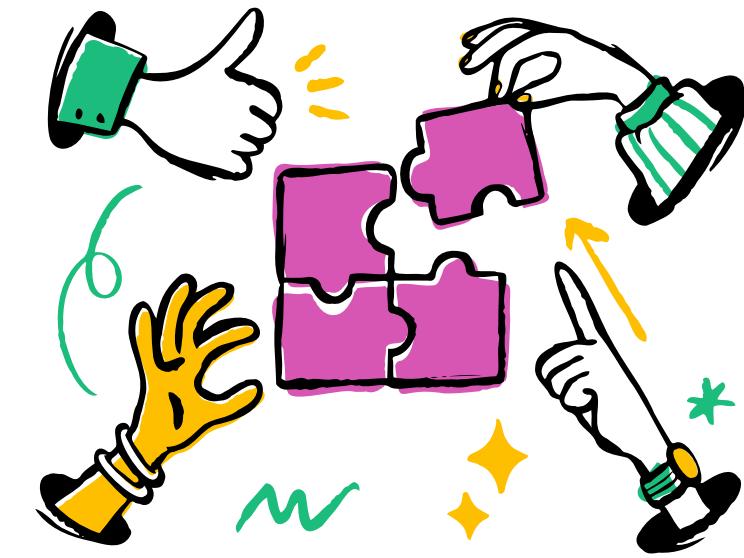
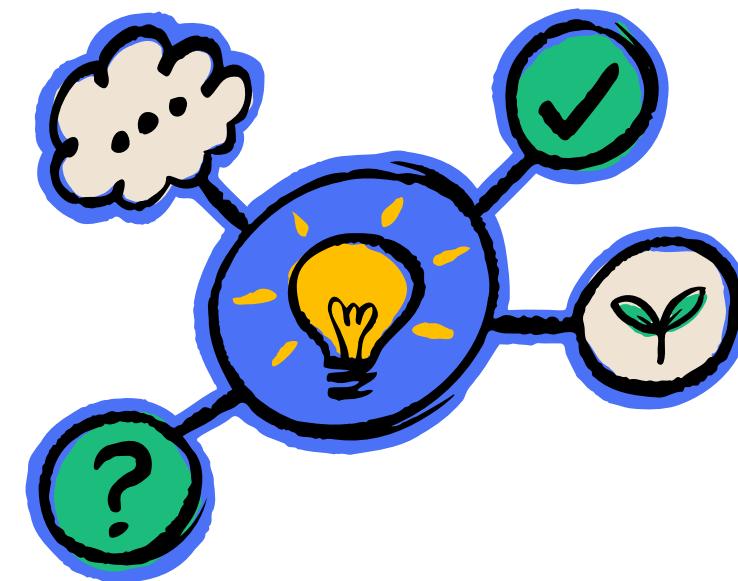


OBJETIVOS ESPECÍFICOS:

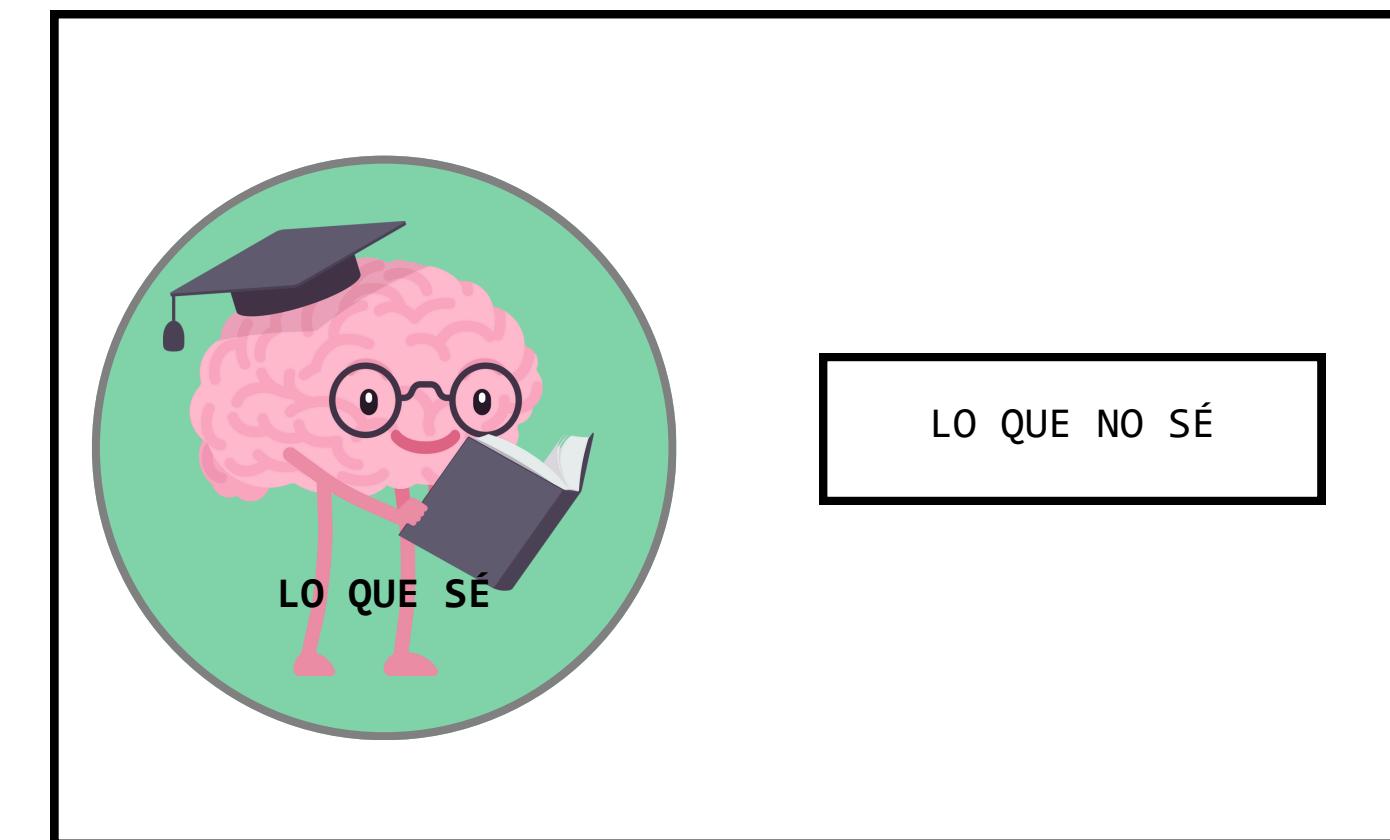
1. COMPRENDER CÓMO LOS CONCEPTOS BÁSICOS DE LA PROGRAMACIÓN, COMO CONDICIONES, BUCLES Y FUNCIONES, PUEDEN USARSE PARA MODELAR Y RESOLVER PROBLEMAS.
2. FOMENTAR HABILIDADES DE ANÁLISIS Y RESOLUCIÓN DE PROBLEMAS.
3. CONSTRUIR UNA BASE SÓLIDA EN PYTHON QUE PERMITA A LOS ESTUDIANTES CONTINUAR APRENDIENDO CONCEPTOS MÁS AVANZADOS DE PROGRAMACIÓN.
4. AYUDAR A LOS ESTUDIANTES A COMPRENDER LA IMPORTANCIA DEL PENSAMIENTO LÓGICO Y ESTRUCTURADO EN LA RESOLUCIÓN DE PROBLEMAS TANTO DENTRO COMO FUERA DE LA PROGRAMACIÓN.



EXPERIENCIA



EXPERIENCIA



RAZONAMIENTO

RAZONAMIENTO INDUCTIVO

De observaciones específicas a conclusiones generales.
El razonamiento inductivo en programación implica generalizar patrones observados en datos, comportamientos o problemas específicos para crear soluciones más generales.

Ejemplo:

Entrenar un modelo de ML con datos específicos.
Usas datos históricos de usuarios (películas vistas, calificaciones) para entrenar un modelo que prediga qué películas podrían gustarles.

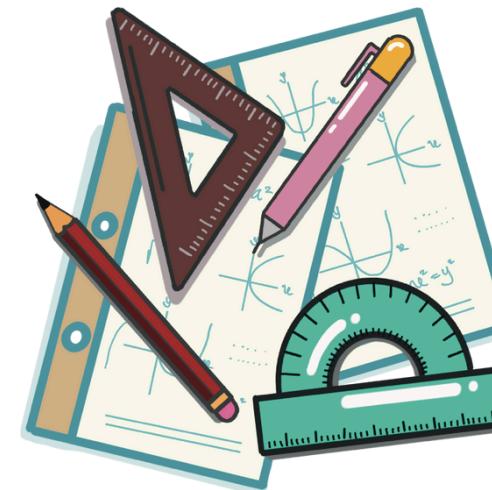


RAZONAMIENTO DEDUCTIVO

De premisas generales a conclusiones específicas.
El razonamiento deductivo implica aplicar reglas generales conocidas a casos específicos para garantizar resultados correctos.

Ejemplo:

Usar una fórmula matemática para resolver un problema.
Aplicas un algoritmo ya probado para calcular similitudes entre usuarios basándose en reglas matemáticas (por ejemplo, la distancia coseno).



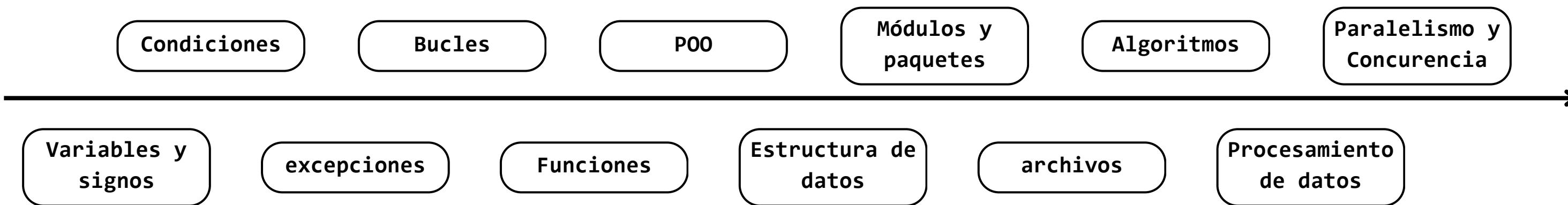
LÓGICA DE PROGRAMACIÓN

La lógica de programación consiste en la organización y planificación coherente de las instrucciones necesarias para ejecutar con éxito un programa.



ESTRUCTURAS BÁSICAS

La lógica de programación organiza las instrucciones utilizando estructuras como secuencias, condicionales, bucles y funciones. Estas herramientas se combinan estratégicamente para resolver problemas del día a día o situaciones abstractas, permitiendo abordar desafíos de manera estructurada y eficiente.



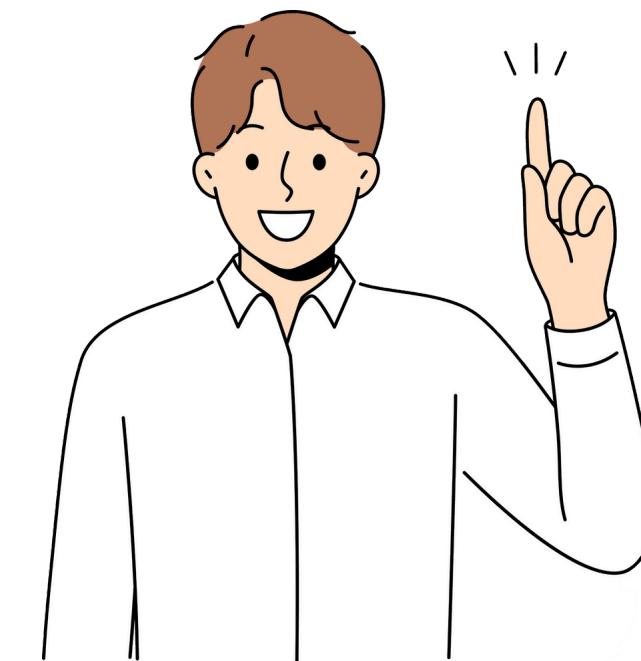
ESTRUCTURAS BÁSICAS

La lógica de programación organiza las instrucciones utilizando estructuras como secuencias, condicionales, bucles y funciones. Estas herramientas se combinan estratégicamente para resolver problemas del día a día o situaciones abstractas, permitiendo abordar desafíos de manera estructurada y eficiente.

¿Se podrían crear más estructuras, aparte de las que ya conocemos?

Si sé que es una estructura, es algo que hacemos día a día, que está ahí, como una condición o un ciclo repetitivo.

Hay algo que está en nuestras narices que es tan obvio, pero no le damos atención hasta que una persona lo nota y decimos: "como no se me ocurrió."



PARADIGMAS DE PROGRAMACIÓN

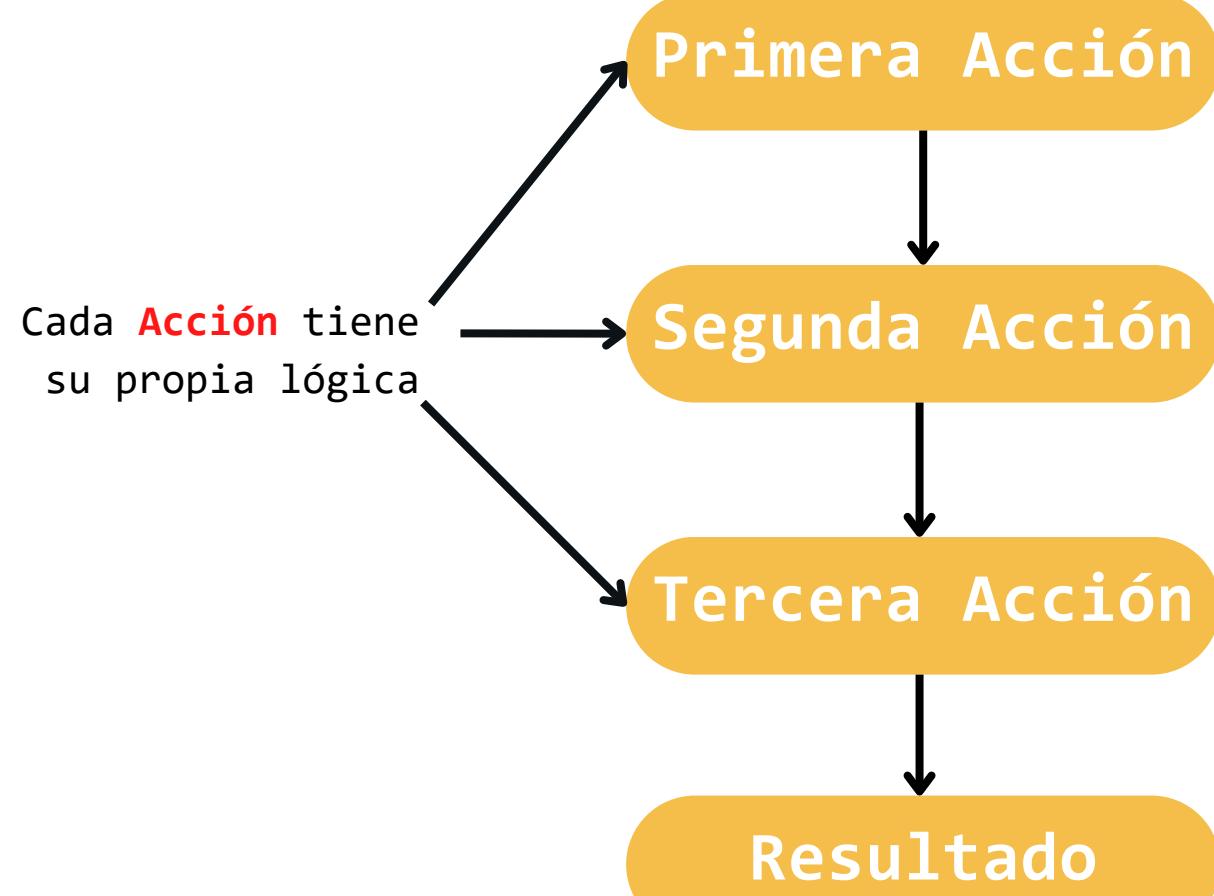
Un paradigma de programación es un estilo o enfoque para resolver problemas mediante programación, definiendo cómo se estructura, organiza y ejecuta el código. Los paradigmas ofrecen diferentes formas de pensar y escribir programas según el problema que se quiera resolver.



Nota: En esta clase veremos el Paradigma Estructurado.

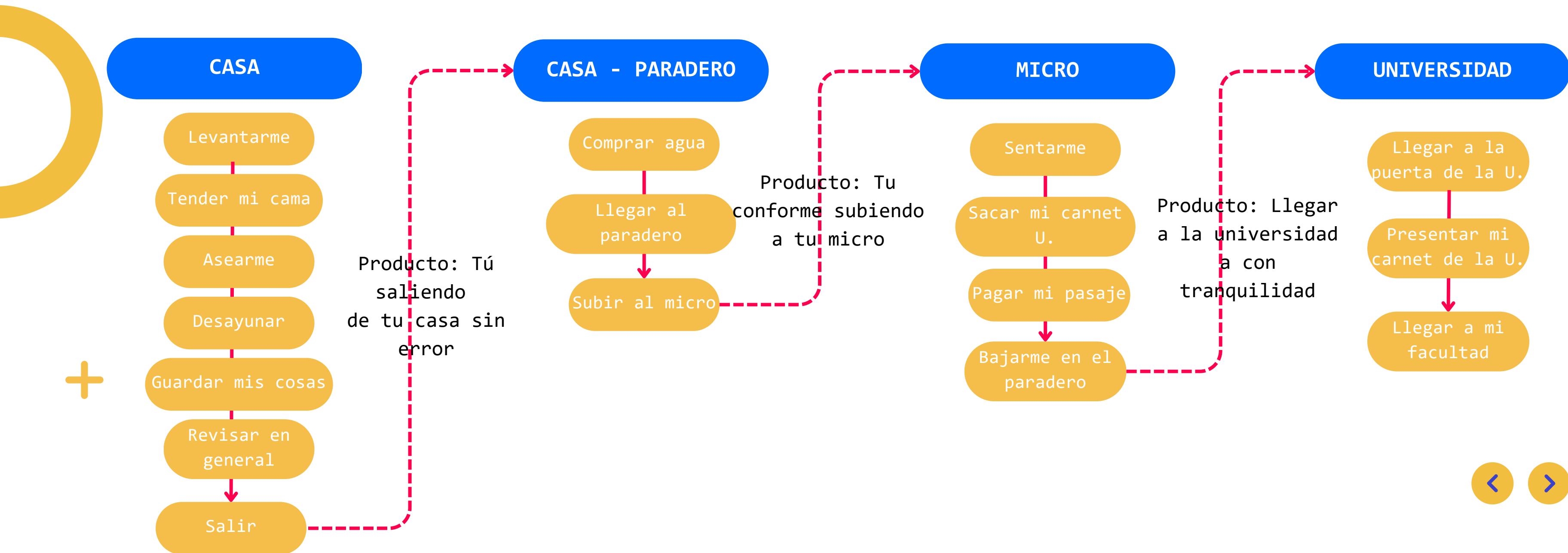


LÓGICA DE PROGRAMACIÓN



LÓGICA DE PROGRAMACIÓN

¿Cuál es tu rutina desde que te despiertas hasta que sales de casa y llegas a tu facultad?



PENSAR LOGICAMENTE

Kevin

Hola, ayúdame sumando 2 números.

Yo te daré 2 números: 5 y 8

Es correcto, muchas gracias.

Estudiante

Está bien, dame 2 números, ¿O utilizo mis números?

Gracias, los sumare

El resultado es: 13

Entonces si tengo 5 y 8 y los sumo seria 13



Pero ahora como podría hacer un programa para que Kevin pueda sumar 2 números.

ENTENDIMIENTO DEL PROBLEMA

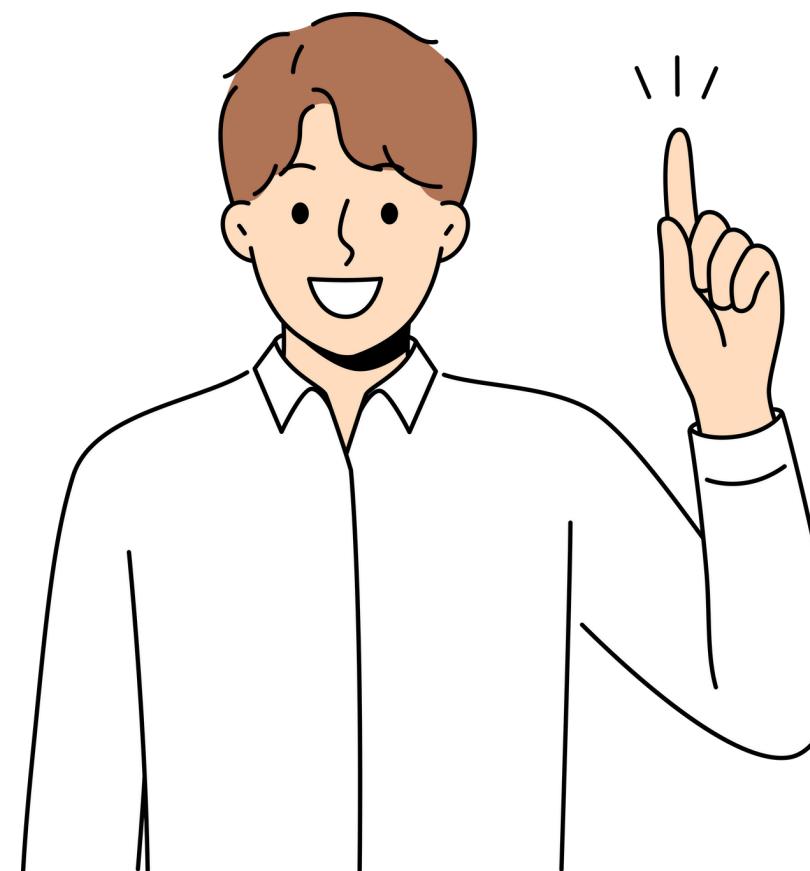


Aunque no sepas programar,
puedes aplicar lógica
básica:

¿Hay condiciones que debes
cumplir? (Por ejemplo, "un
número debe ser divisible
por 2").

¿Necesitas repetir algo
varias veces? (Por
ejemplo, "revisar cada
número en la lista").

¿Debes guardar un
resultado acumulado? (Por
ejemplo, "sumar los pares
encontrados").



Como podría hacer un
programa que sume 2
números.

¿Qué tengo que hacer
primero?

Primero el programa
tendría que pedirle los 2
números.

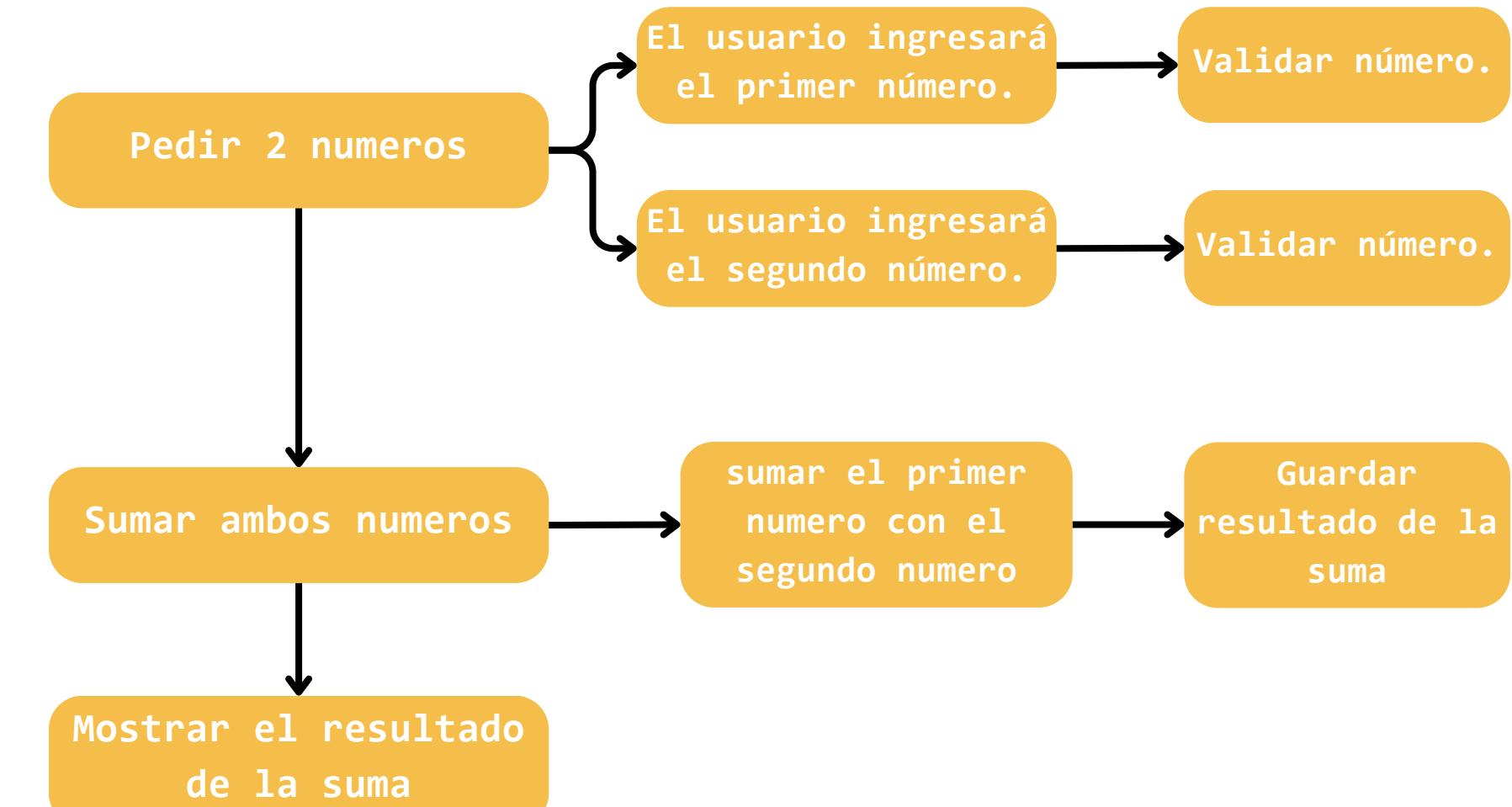
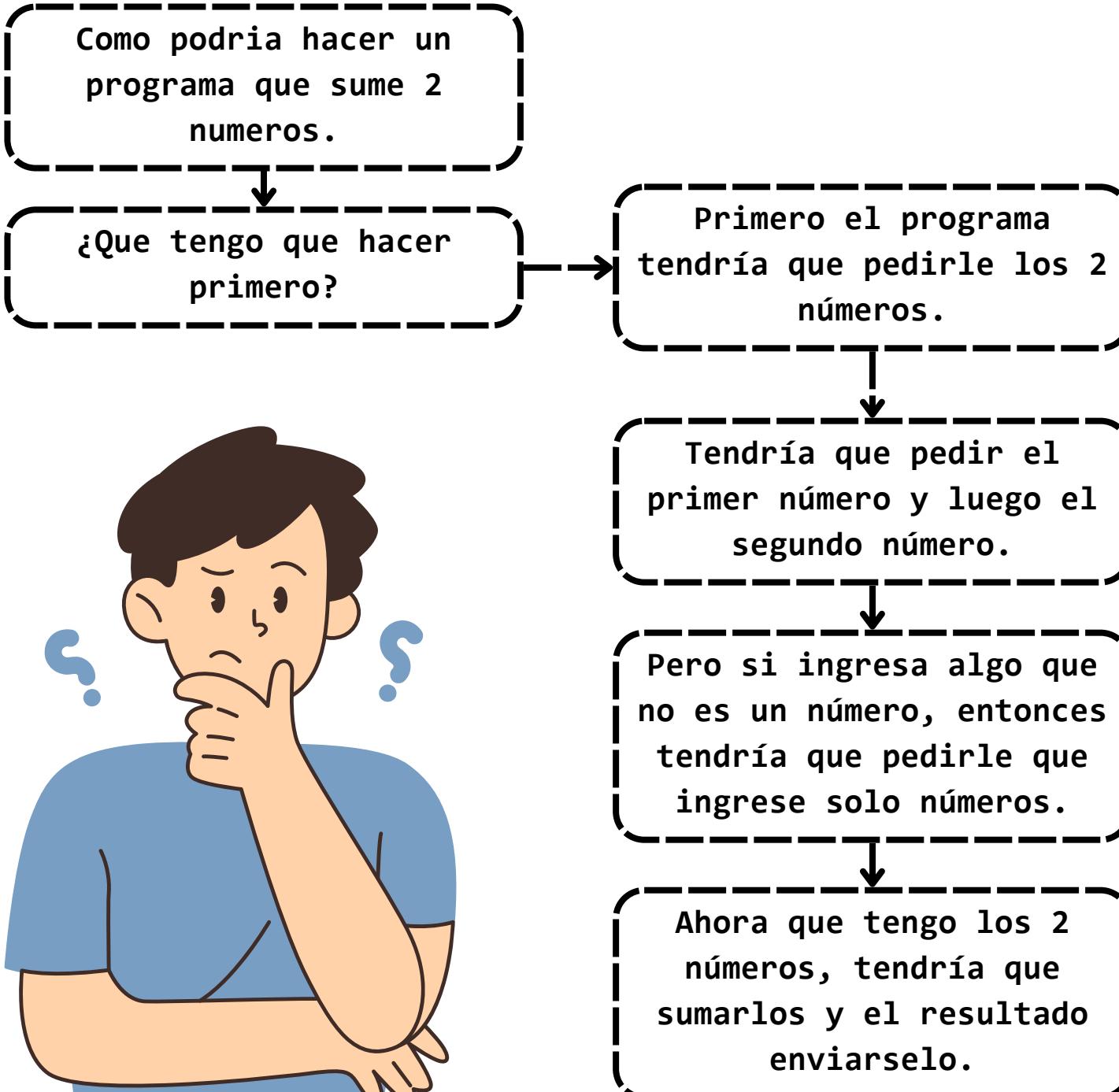
Tendría que pedir el
primer número y luego el
segundo número.

Pero si ingresa algo que
no es un número, entonces
tendría que pedirle que
ingrese solo números.

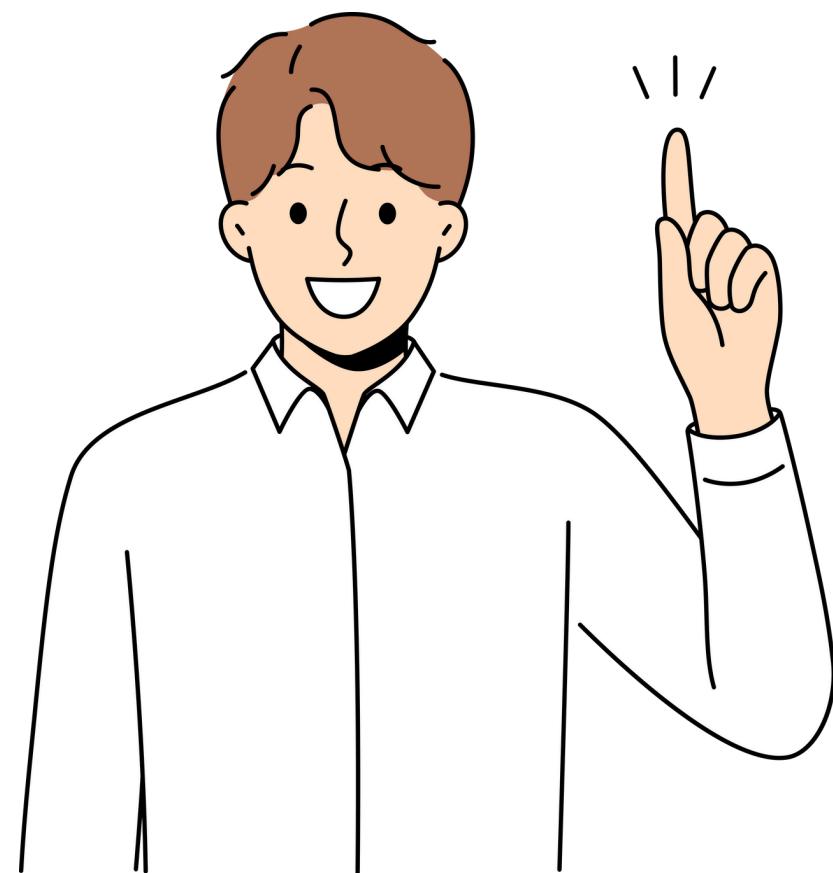
Ahora que tengo los 2
números, tendría que
sumarlos y el resultado
enviarselo.



PENSAR LOGICAMENTE



PASAR A CODIGO



Voy a utilizar el programa
de mi alumno

Programa

Kevin

Ingresé el primer número

12

Ingresé el segundo número

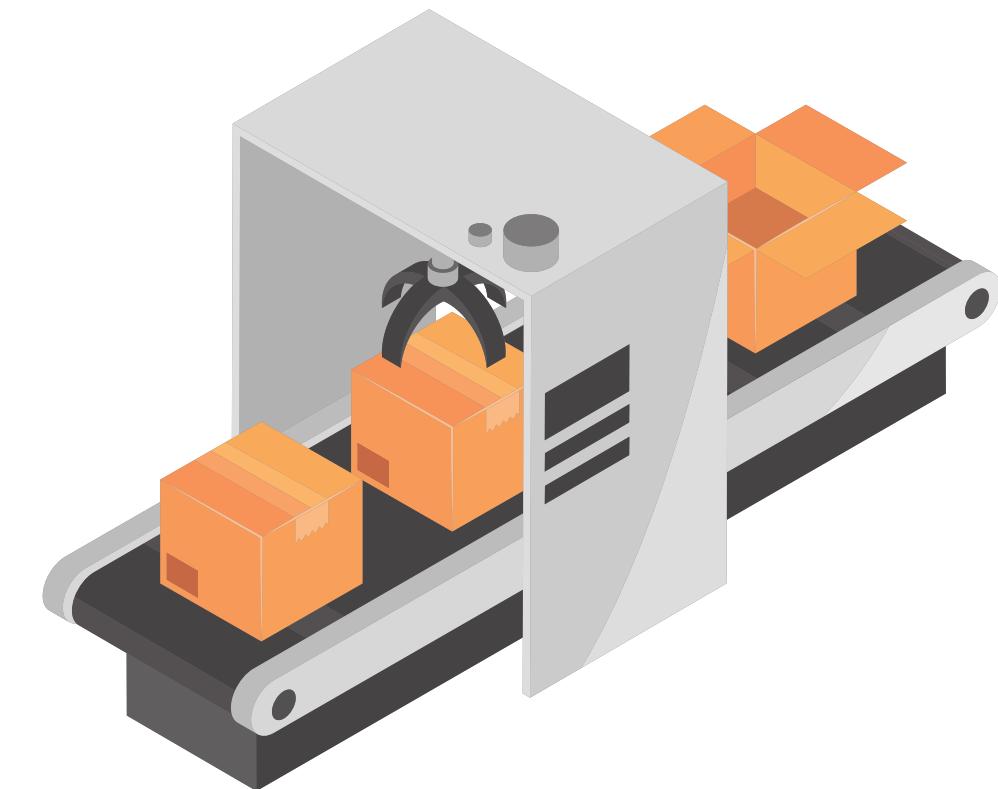
18

El resultado es: 30



PROBLEMAS

Problema 1: Estás trabajando en un sistema que analiza los hábitos de compra en una tienda en línea. Se te proporciona una lista de productos comprados en un día, y necesitas identificar cuál es el producto más comprado para destacarlo en promociones futuras.



PROBLEMAS

Problema 1: Estás trabajando en un sistema que analiza los hábitos de compra en una tienda en línea. **Se te proporciona una lista de productos comprados en un día**, y necesitas identificar cuál es **el producto más comprado** para destacarlo en promociones futuras.



El reto es encontrar el número (o producto) que aparece más veces en la lista.

Sin embargo, al analizar el problema, te das cuenta de que esto puede dividirse en dos pasos:

1. Contar las veces que aparece cada producto
2. Encontrar el que tiene la mayor cantidad.

Pero ahora que tienes esos 2 puntos, dentro de esos puntos hay más paso:

Del punto 1: Contar las veces que aparece cada producto

- ¿Cómo hago para contar producto por producto?
- ¿Cómo están almacenados?

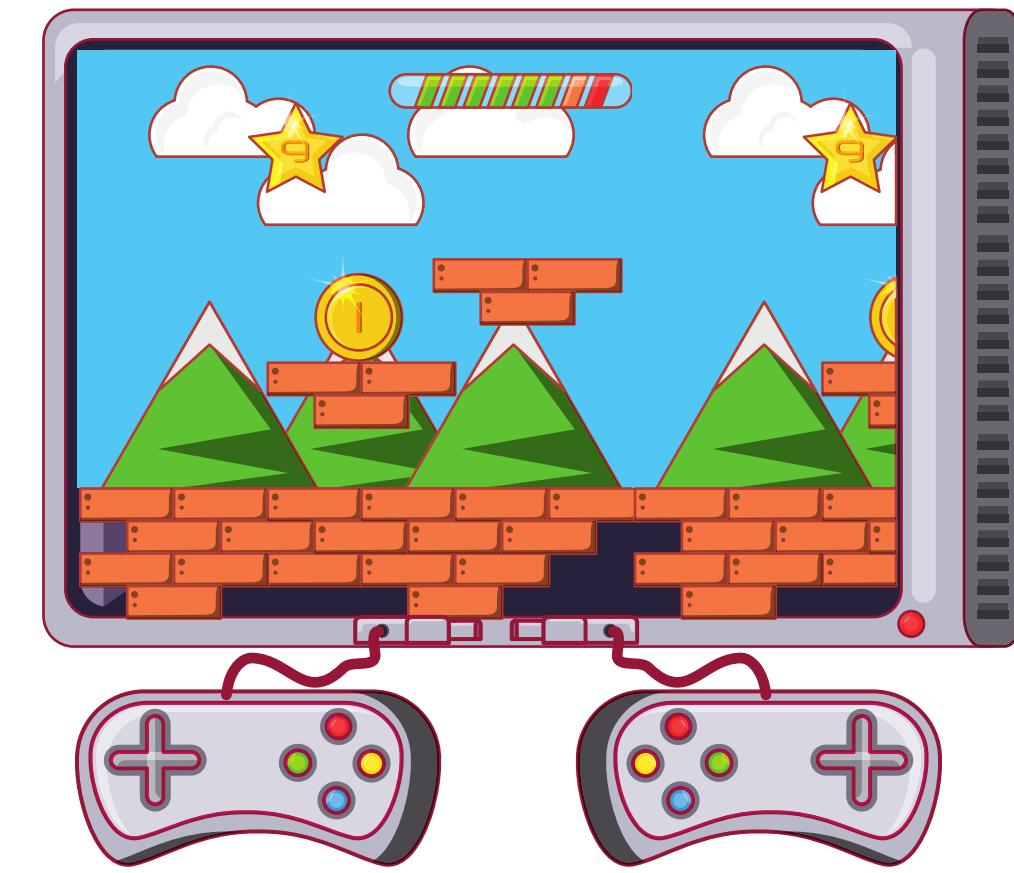


Del punto 2:

- ¿Luego de contar tendré que comparar para saber quién tiene más productos vendidos?
- ¿Mostrar el producto más vendido?

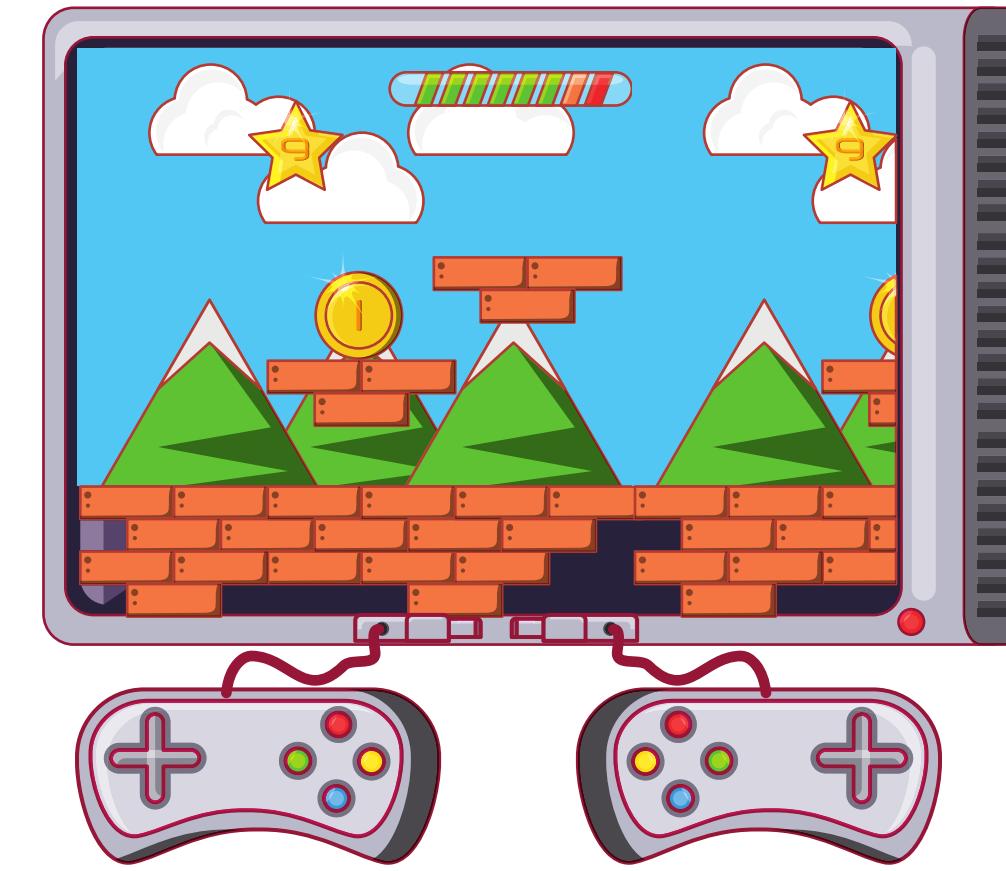
PROBLEMAS

Problema 2: Imagina que estás creando un sistema para calcular las puntuaciones únicas en un torneo de videojuegos. Cada jugador obtiene varios puntajes, pero el sistema debe ignorar aquellos que se repiten, sumando únicamente los valores únicos para calcular la puntuación final.



PROBLEMAS

Problema 2: Imagina que estás creando un sistema para calcular **las puntuaciones únicas** en un torneo de videojuegos. Cada jugador obtiene varios **puntajes**, pero el sistema debe ignorar aquellos que se repiten, sumando únicamente los **valores únicos** para calcular la **puntuación final**.





Pregunta: ¿Qué estructura de datos es más adecuada para almacenar los puntajes de cada jugador? ¿Una lista, un conjunto, un diccionario?

Lógica: Dado que necesitamos manejar múltiples puntajes y eliminar duplicados, un conjunto (set) podría ser útil porque automáticamente ignora los valores repetidos.

Pregunta: ¿Cómo podemos asegurarnos de que solo se consideren los puntajes únicos para cada jugador?

Lógica: Si usamos un conjunto, los duplicados se eliminarán automáticamente. Si usamos una lista, podríamos convertirla a un conjunto y luego de vuelta a una lista para eliminar duplicados.

Pregunta: ¿Cuál es la mejor manera de sumar los valores únicos para obtener la puntuación final?

Lógica: Una vez que tengamos los puntajes únicos (ya sea en una lista o un conjunto), podemos iterar sobre ellos y sumarlos para obtener la puntuación final.

Pregunta: ¿Qué pasa si un jugador no tiene puntajes, o si los puntajes no son números válidos?

Lógica: Debemos incluir validaciones para asegurarnos de que los puntajes sean números y que cada jugador tenga al menos un puntaje válido.

Pregunta: ¿Qué pasa si dos o más jugadores tienen la misma puntuación final?

Lógica: Podemos decidir si mostrar a todos los jugadores con la puntuación más alta, o implementar un criterio de desempate (por ejemplo, el jugador que alcanzó la puntuación primero).



PROBLEMAS

Problema 3: Estás creando un programa que cuenta cuántas vocales (a, e, i, o, u) hay en una palabra o frase ingresada por el usuario. El programa debe ser insensible a mayúsculas y minúsculas, es decir, debe contar tanto las vocales en mayúsculas como en minúsculas.



PROBLEMAS

Problema 3: Estás creando un programa que **cuenta cuántas vocales (a, e, i, o, u) hay en una palabra o frase** ingresada **por el usuario**. El programa **debe ser insensible a mayúsculas y minúsculas**, es decir, debe **contar tanto las vocales** en mayúsculas como en minúsculas.



Pregunta: ¿Cómo podemos verificar si un carácter es una vocal, sin importar si está en mayúscula o minúscula?

Lógica: Podemos convertir el texto a minúsculas (o mayúsculas) y luego comparar cada carácter con las vocales.

Pregunta: ¿Cuál es la mejor manera de iterar sobre cada carácter del texto para contar las vocales?

Lógica: Podemos usar un bucle for para recorrer el texto carácter por carácter.

Pregunta: ¿Qué estructura de datos es más adecuada para almacenar el conteo de vocales?

Lógica: Podemos usar un contador (una variable numérica) que se incremente cada vez que encontramos una vocal.



Pregunta: ¿Qué formato debe tener la salida del programa para que el usuario entienda el resultado?

Lógica: Podemos imprimir un mensaje claro que indique el número total de vocales encontradas.



LA NATURALEZA CONTEXTUAL DE LAS VARIABLES

Las variables no solo almacenan datos, sino que su tipo depende del contexto y propósito con el que se utiliza esa información. Esto significa que no basta con identificar el valor de una variable; también debemos considerar cómo será interpretado o usado en un programa.

Ejemplo

Número: **943617892**

1. Si se usa como un contador o para realizar operaciones matemáticas:

- **Tipo:** Entero (int).
- **Uso:** Sumar, restar o realizar cálculos.
- **Ejemplo:** Incrementar un contador en 1.

2. Si representa un número de teléfono:

- **Tipo:** Cadena (string).
- **Uso:** Almacenar como texto, ya que no se realizarán operaciones matemáticas con él.
- **Ejemplo:** Formatear como "+51 943-617-892".

3. Si se trata de una clave única en una base de datos:

- **Tipo:** Cadena o, en algunos casos, identificador (UUID).
- **Uso:** Referenciar algo específico, no operar matemáticamente.

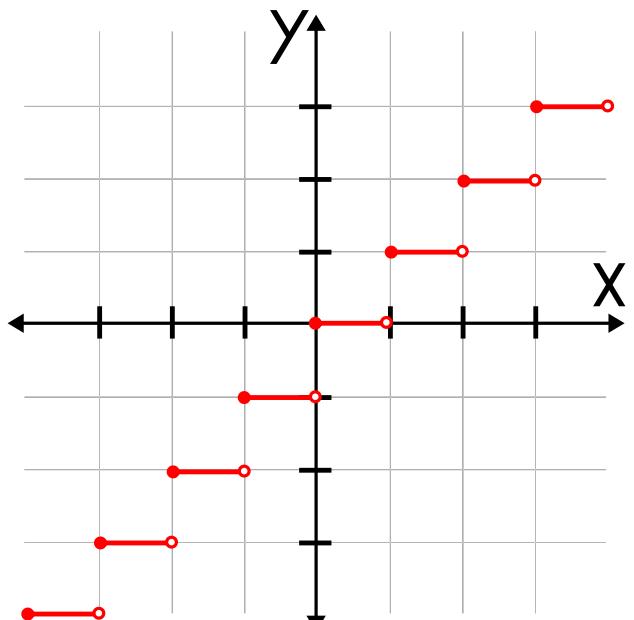


TIPOS DE VARIABLES.

Enteros (int)

Se utilizan para almacenar números enteros, sin decimales.

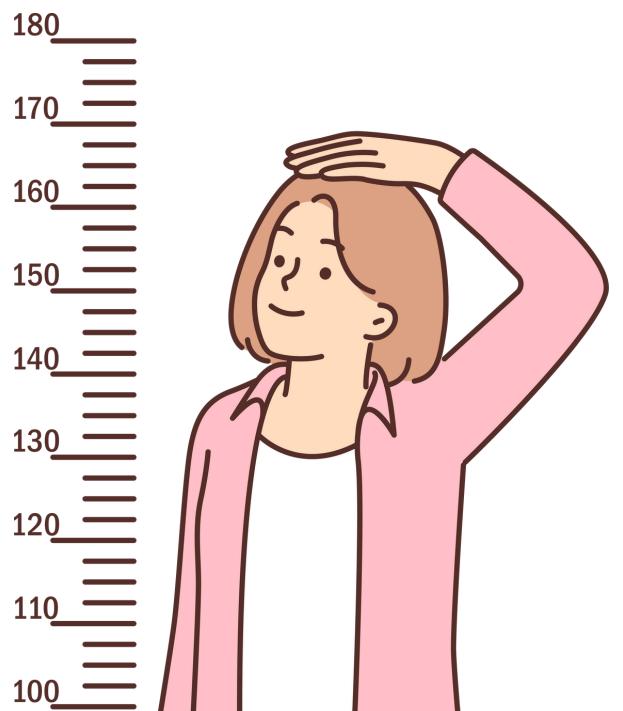
- Ejemplos:
 - Cantidad: 10 (número de manzanas).
 - Edad: 25 (años de una persona).
 - Años: 2024 (año actual).
 - Días: 365 (días en un año).



Reales (float o double)

Se usan para valores con decimales, como mediciones o cálculos precisos.

- Ejemplos:
 - Altura: 1.75 metros.
 - Peso: 68.5 kilogramos.
 - Promedios: 89.3 (promedio de calificaciones).
 - Medidas: 3.1416 (valor de π).



Caracter (char)

Almacena un solo símbolo, letra o número en formato ASCII.

- Ejemplos:
 - Símbolos: 'A', '5', '#'.

| Caracteres ASCII imprimibles | | | | ASCII extendido (Página de código 437) | | | |
|------------------------------|---------|----|---|--|---|-----|-------|
| 32 | espacio | 64 | @ | 128 | ç | 160 | á |
| 33 | ! | 65 | A | 129 | ú | 161 | í |
| 34 | " | 66 | B | 130 | é | 162 | ó |
| 35 | # | 67 | C | 131 | â | 163 | û |
| 36 | \$ | 68 | D | 132 | ã | 164 | ñ |
| 37 | % | 69 | E | 133 | à | 165 | N |
| 38 | & | 70 | F | 134 | à | 166 | ª |
| 39 | ' | 71 | G | 135 | ç | 167 | º |
| 40 | (| 72 | H | 136 | è | 168 | ξ |
| 41 |) | 73 | I | 137 | ë | 169 | ® |
| 42 | * | 74 | J | 138 | è | 170 | ™ |
| 43 | + | 75 | K | 139 | í | 171 | ½ |
| 44 | , | 76 | L | 140 | í | 172 | ¼ |
| 45 | - | 77 | M | 141 | í | 173 | i |
| 46 | . | 78 | N | 142 | À | 174 | « |
| 47 | / | 79 | O | 143 | ò | 175 | » |
| 48 | 0 | 80 | P | 144 | É | 176 | ò |
| 49 | 1 | 81 | Q | 145 | æ | 177 | ò |
| 50 | 2 | 82 | R | 146 | Æ | 178 | ò |
| 51 | 3 | 83 | S | 147 | ò | 179 | — |
| 52 | 4 | 84 | T | 148 | ö | 180 | — |
| 53 | 5 | 85 | U | 149 | ò | 181 | Á |
| 54 | 6 | 86 | V | 150 | ú | 182 | À |
| 55 | 7 | 87 | W | 151 | ù | 183 | À |
| 56 | 8 | 88 | X | 152 | ÿ | 184 | © |
| 57 | 9 | 89 | Y | 153 | ö | 185 | í |
| 58 | : | 90 | Z | 154 | Ü | 186 | — |
| 59 | : | 91 | [| 155 | ø | 187 | — |
| 60 | < | 92 | \ | 124 |] | 156 | £ |
| 61 | = | 93 | I | 125 | } | 157 | Ø |
| 62 | > | 94 | ^ | 126 | ~ | 158 | × |
| 63 | ? | 95 | _ | 190 | ¥ | 222 | — |
| | | | | 159 | f | 191 | nnbsp |



TIPOS DE VARIABLES.

Cadenas (string)

Se usan para almacenar texto, como palabras o frases.

- Ejemplos:
 - Palabras: "Hola".
 - Oraciones: "El cielo es azul".
 - Texto sin sentido: "xkjsdf98wq".



Booleano (bool)

Solo tiene dos valores posibles, usado para tomar decisiones lógicas.

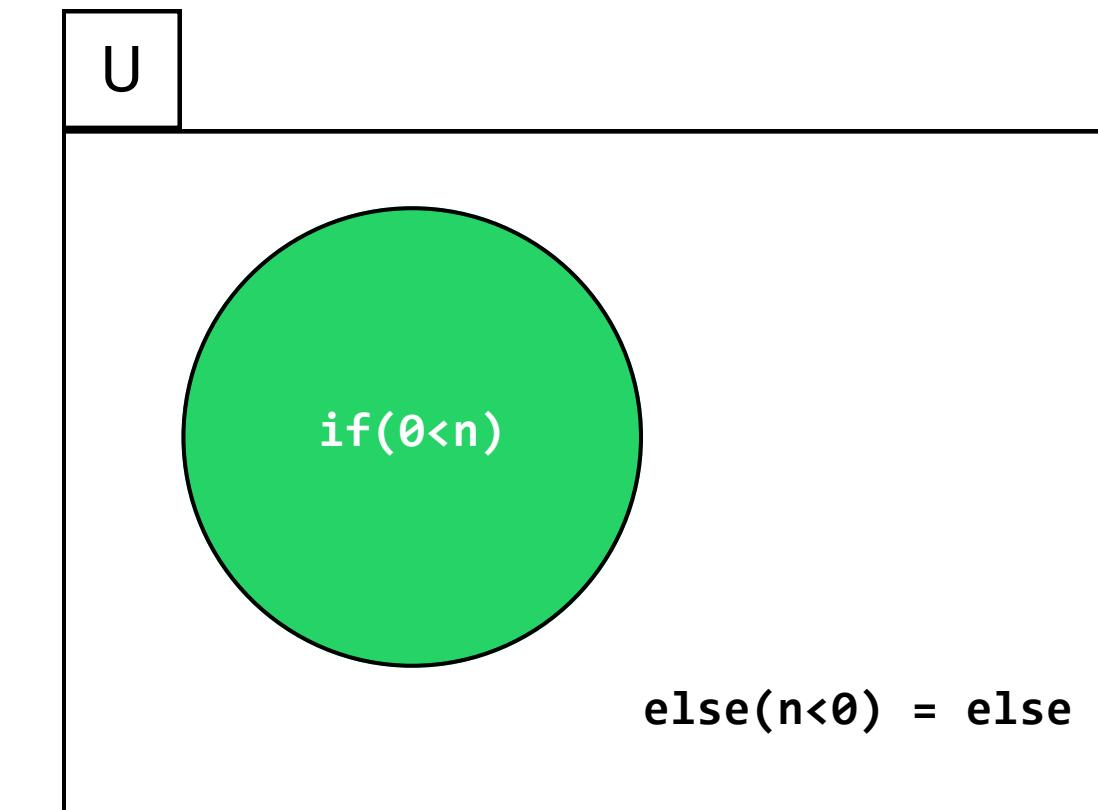
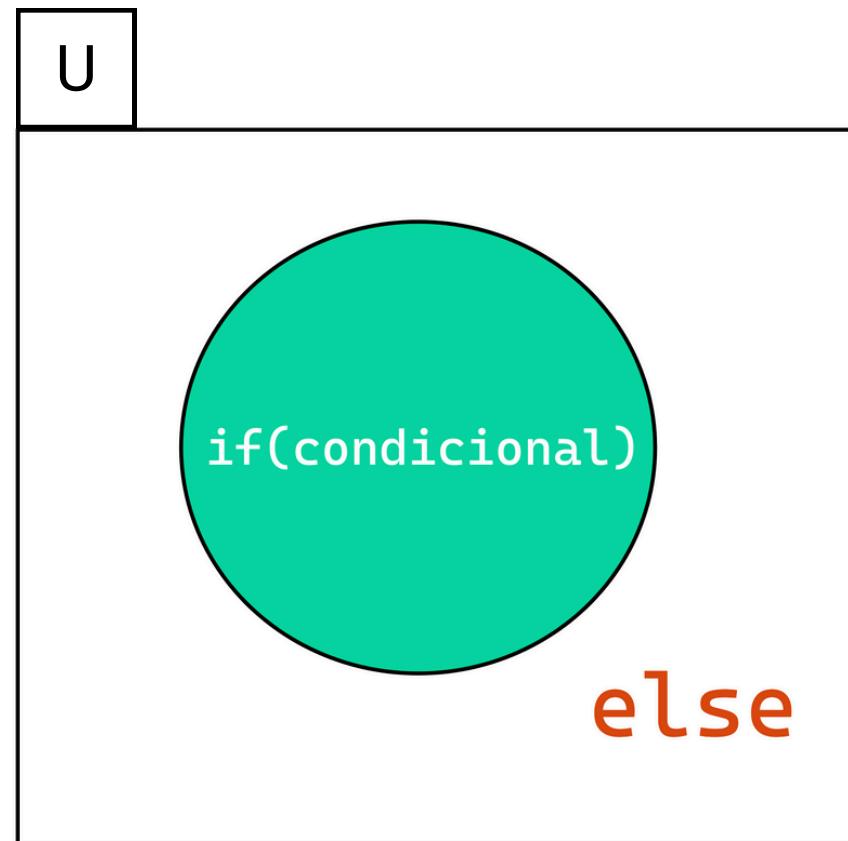
- Ejemplos:
 - Verdadero: true (es mayor de edad).
 - Falso: false (el usuario no ingresó la contraseña correcta).

TRUE

FALSE

CONDICIONALES

Los condicionales son estructuras de control que permiten que un programa siga diferentes caminos dependiendo de si una condición es verdadera o falsa. Esto hace que los programas sean más dinámicos, adaptándose a diferentes escenarios durante la ejecución.

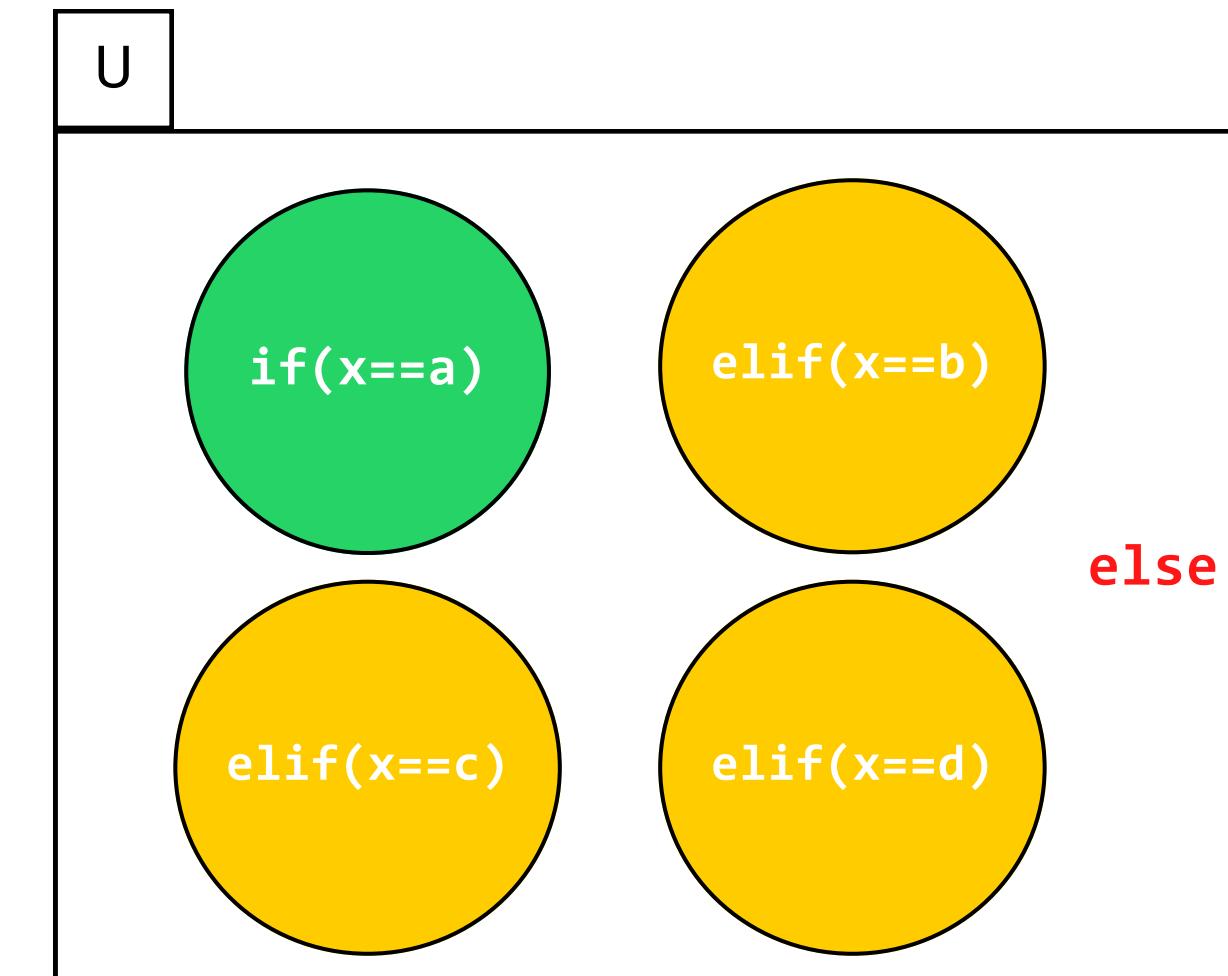
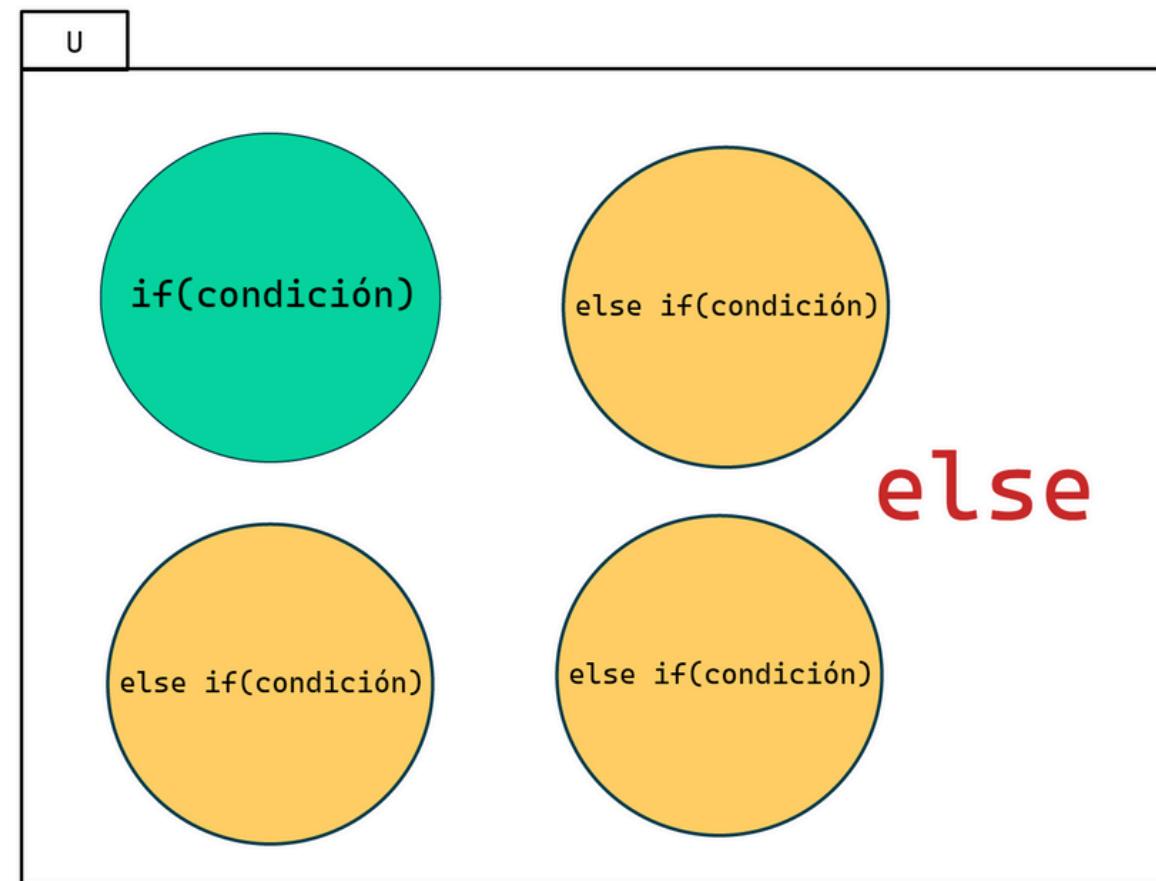


Nota: Esto es el caso para una variable



CONDICIONALES

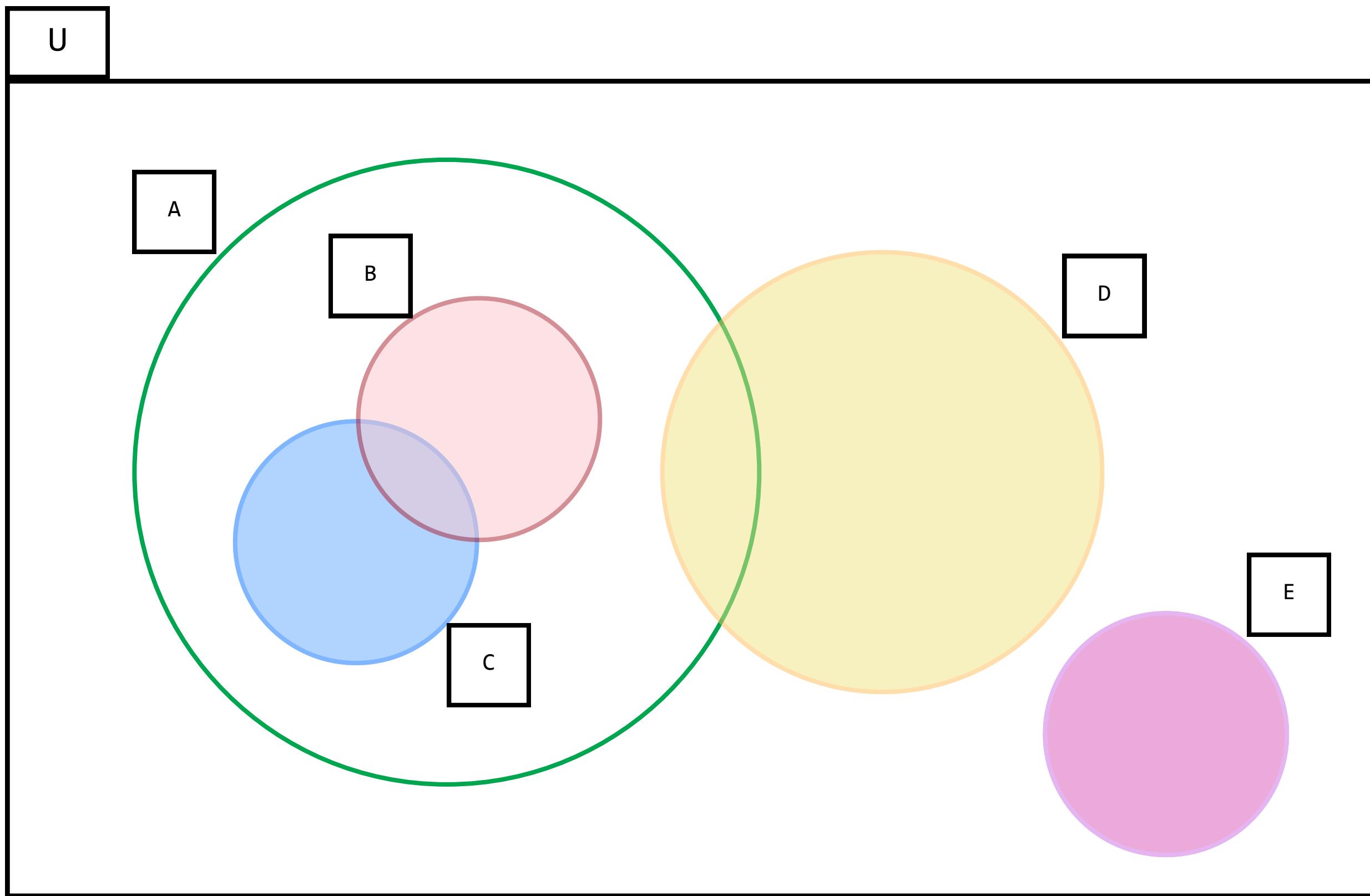
Los condicionales son estructuras de control que permiten que un programa siga diferentes caminos dependiendo de si una condición es verdadera o falsa. Esto hace que los programas sean más dinámicos, adaptándose a diferentes escenarios durante la ejecución.



Nota: Esto es el caso para una variable



CONJUNTOS



Nota: Esto es el caso para más variables

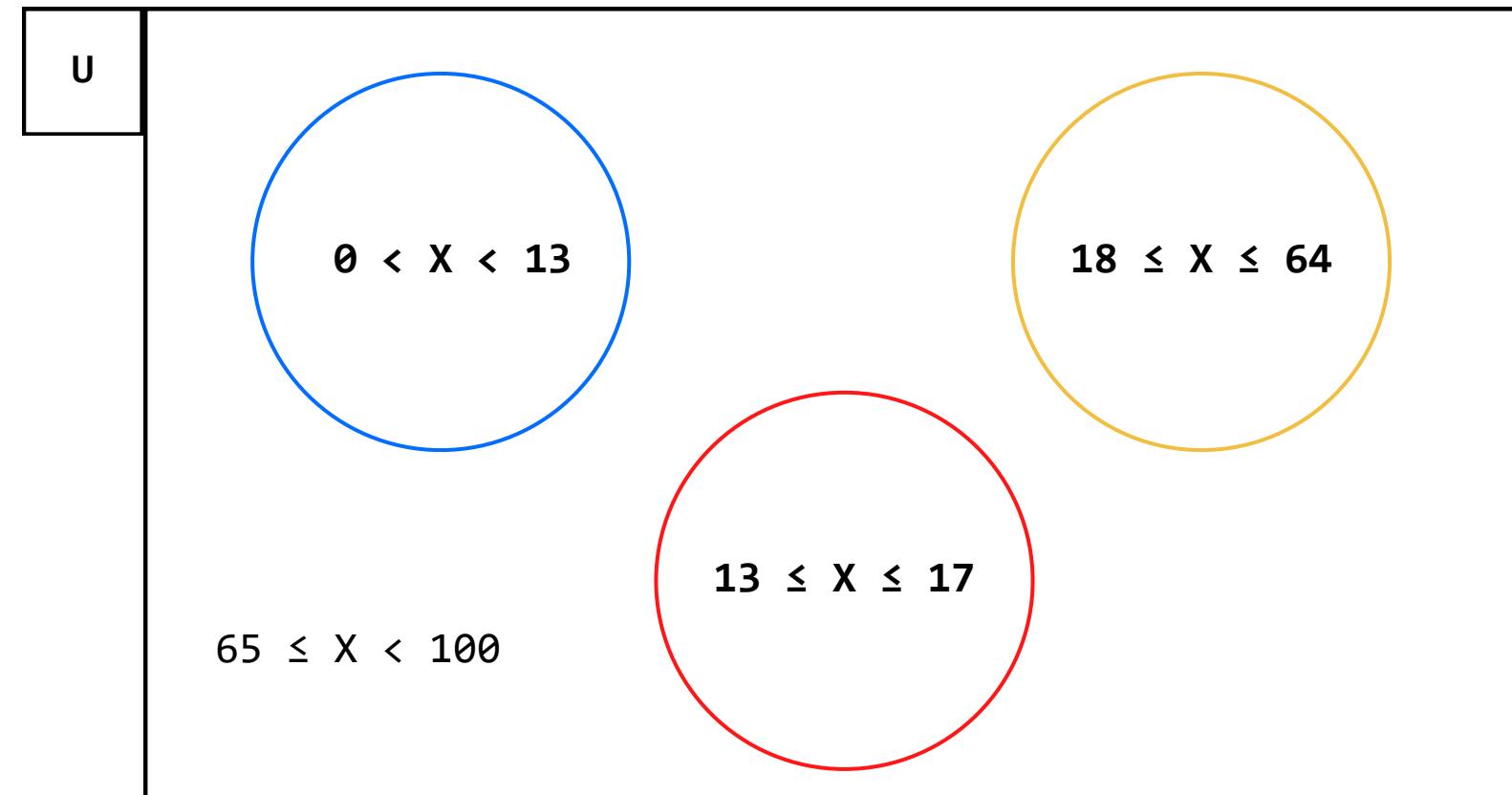


CONDICIONALES

Problema 1: Clasificación de edades

Escribe un programa que solicite al usuario su edad y determine en qué categoría se encuentra:

- Menor de 13 años: "Niño"
- Entre 13 y 17 años: "Adolescente"
- Entre 18 y 64 años: "Adulto"
- 65 años o más: "Adulto mayor"

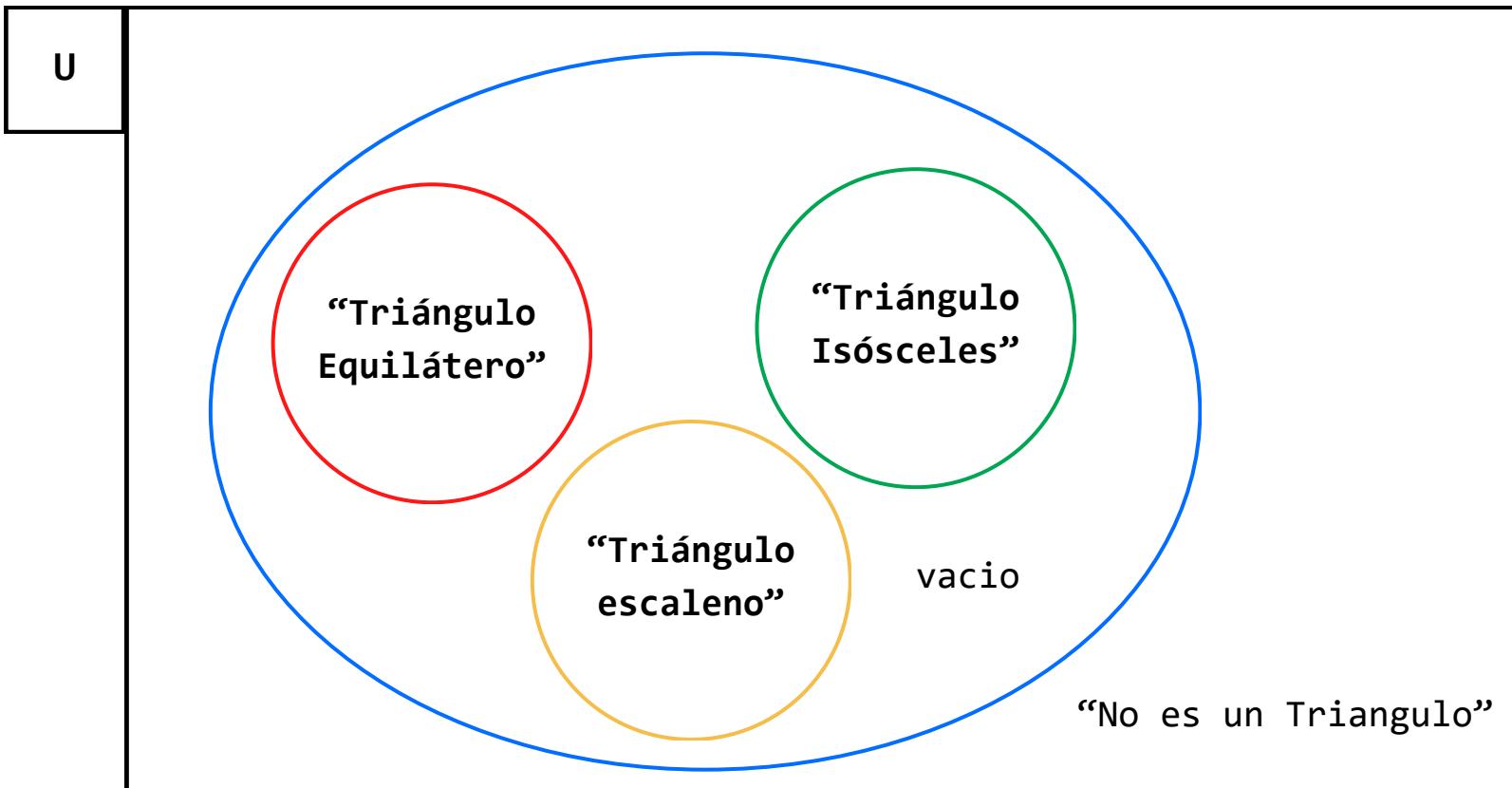
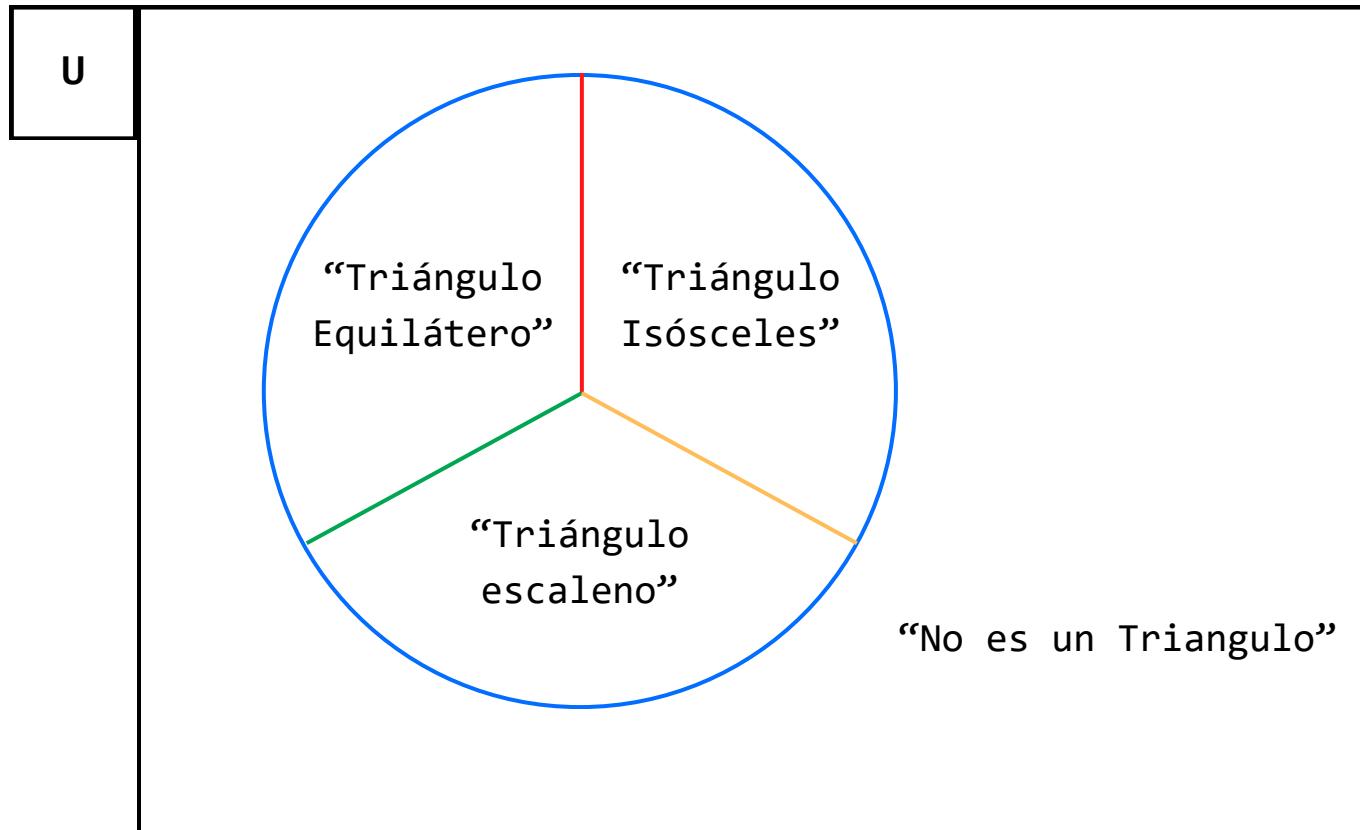


CONDICIONALES

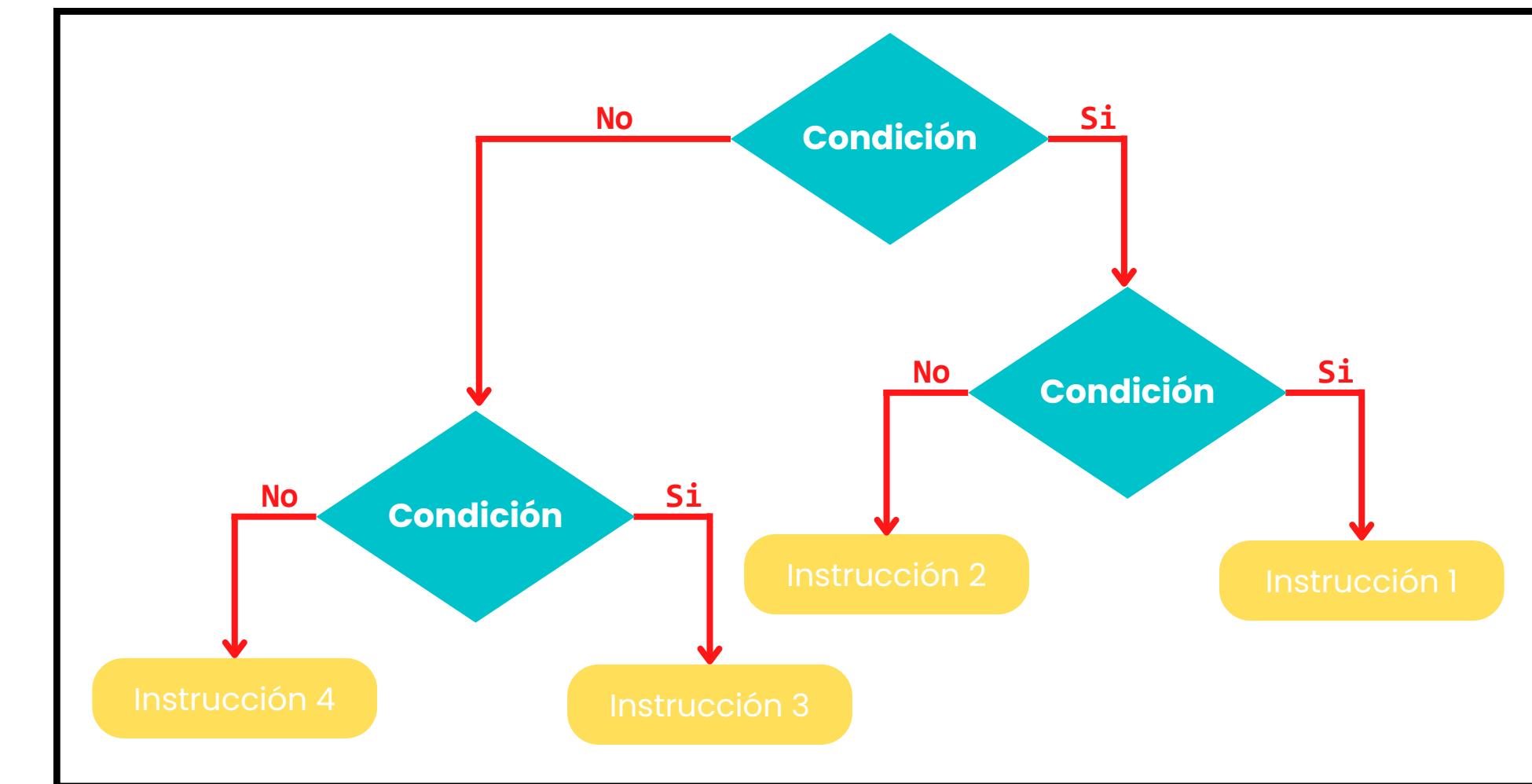
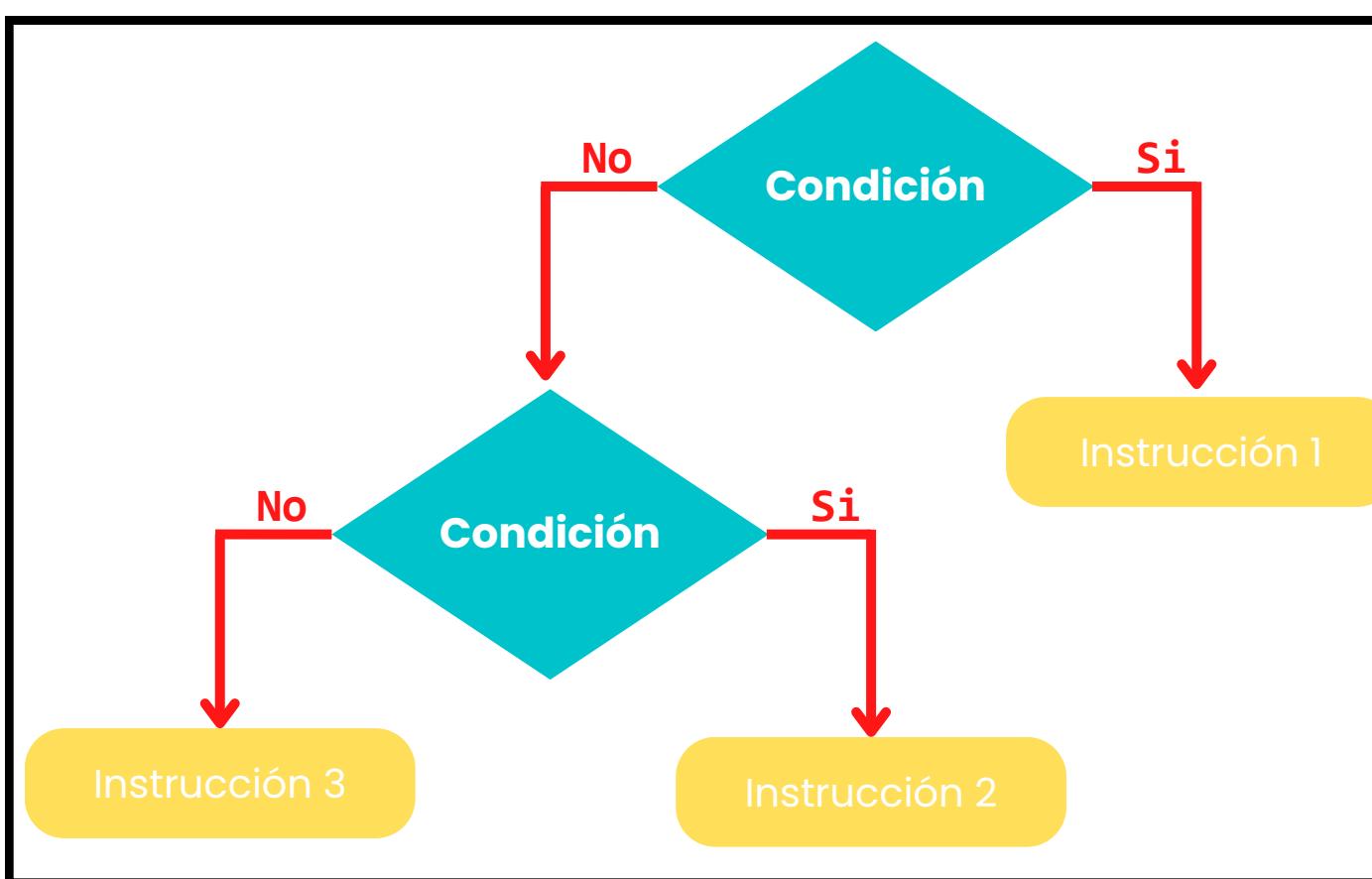
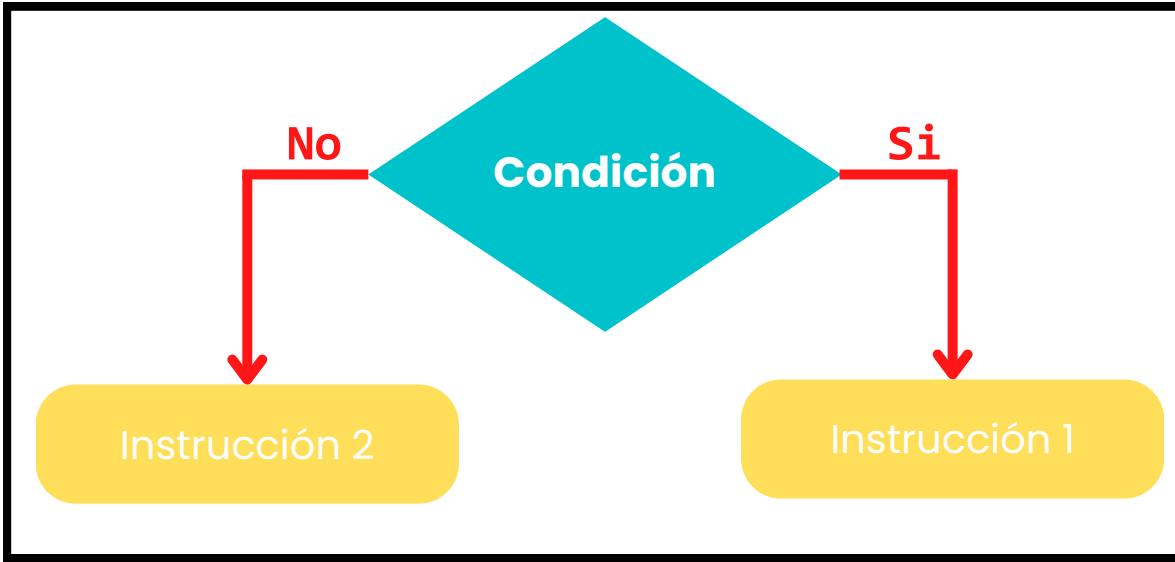
Problema 2: Clasificación de triángulos

Escribe un programa que solicite al usuario los tres lados de un triángulo y determine:

1. Si los lados no forman un triángulo válido, muestra: "No es un triángulo".
2. Si es un triángulo válido:
 - Si todos los lados son iguales, es un triángulo equilátero.
 - Si dos lados son iguales, es un triángulo isósceles.
 - Si todos los lados son diferentes, es un triángulo escaleno.



CONDICIONALES ANIDADAS



ARBOL DE DESICIONES

Un árbol de decisiones es una herramienta ampliamente utilizada en diversos campos como la inteligencia artificial, la estadística y la toma de decisiones empresariales. Su estructura jerárquica permite descomponer un problema en subproblemas más pequeños, representando decisiones y sus posibles consecuencias de manera clara y visual.

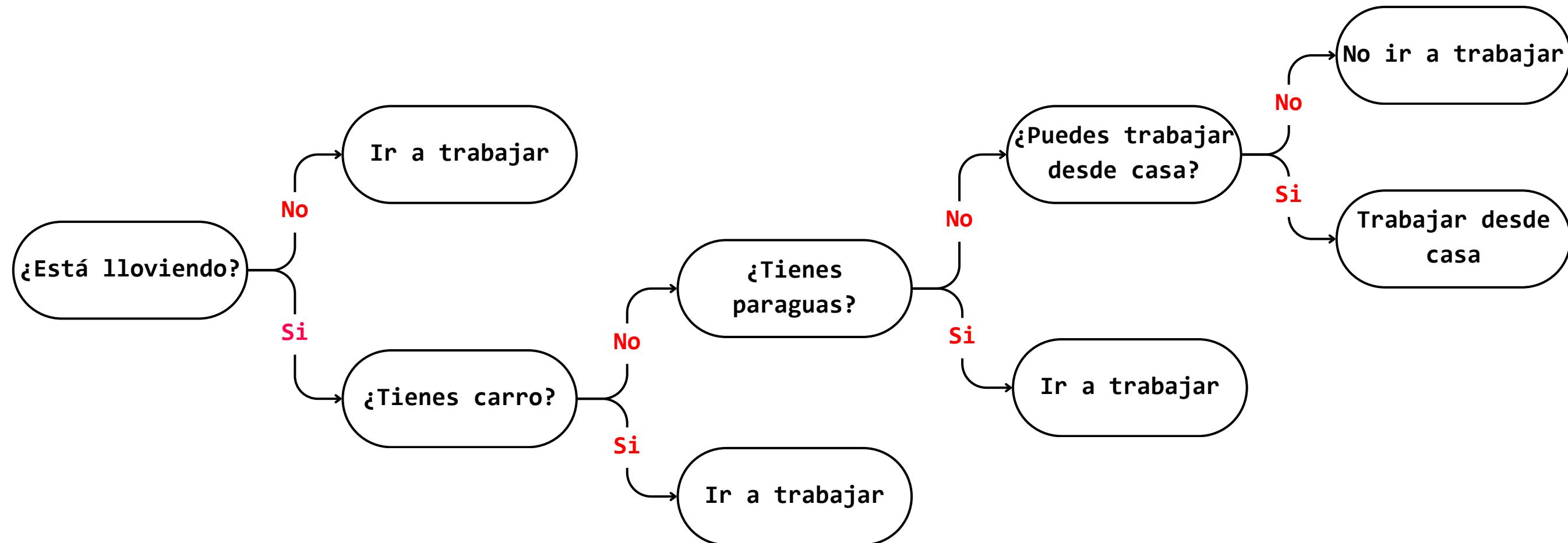


TABLA DE PREPOSICIONES

| p | q | and |
|---|---|-----|
| v | v | v |
| v | f | f |
| f | v | f |
| f | f | f |

Problema: Determinar si un número está dentro de un rango permitido y es par.

- Condiciones:

- P: El número está entre 10 y 50 (inclusive).
- Q: El número es divisible entre 2.

Código:

```
numero = 24
```

```
if(10 <= numero <= 50) and (numero % 2 == 0):
```

```
    print("El número es válido.")
```

```
else:
```

```
    print("El número no cumple las  
condiciones.")
```



TABLA DE PREPOSICIONES

| p | q | and |
|---|---|-----|
| V | V | V |
| V | F | F |
| F | V | F |
| F | F | F |

numero = 35

```
if (numero >= 10 and numero <= 100) and \
(numero % 2 != 0) and \
(numero % 5 == 0) and \
(numero % 7 != 0) and \
(numero != 50):
    print("El número cumple todas las condiciones.")
else:
    print("El número no cumple todas las condiciones.")
```



TABLA DE PREPOSICIONES

| p | q | or |
|---|---|----|
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

Problema: Comprobar si un número es divisible entre 3 o entre 5.

- Condiciones:

- P: El número es divisible entre 3.
- Q: El número es divisible entre 5.

Código:

```
numero = 10
```

```
if (numero % 3 == 0) or (numero % 5 == 0):  
    print("El número es divisible entre 3 o 5.")
```

else:

```
    print("El número no es divisible entre 3 ni entre 5.")
```



TABLA DE PREPOSICIONES

| p | q | or |
|---|---|----|
| V | V | V |
| V | F | V |
| F | V | V |
| F | F | F |

numero = 35

```
if (numero >= 10 or numero <= 100) and \
(numero % 2 != 0) or \
(numero % 5 == 0) or \
(numero % 7 != 0) or \
(numero != 50):
    print("El número cumple al menos una de las condiciones.")
else:
    print("El número no cumple ninguna de las condiciones.")
```



TABLA DE PREPOSICIONES

| | |
|---|----|
| P | ~P |
| V | F |
| F | V |

Problema: Determinar si un número no es par.

- Condiciones:

- P: El número es par.
- NOT P: El número no es par (es impar).

Código:

```
es_mayor_de_edad = True
```

```
if not es_mayor_de_edad:  
    print("La persona no es mayor de edad.")  
else:  
    print("La persona es mayor de edad.")
```



ITERADORES

Un iterable es cualquier objeto en Python (o en otros lenguajes de programación) que puede ser recorrido o iterado, es decir, un objeto sobre el cual se puede realizar un ciclo.

Código:

```
frutas = ["manzana", "banana", "cereza"]  
  
for fruta in frutas:  
    print(fruta)
```

Lo que realiza es lo siguiente

Tenemos la lista Frutas y dentro de ella hay elementos: manzana, banana, cereza. Para acceder a cada elemento ello utilizamos un bucle.

- 1 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> manzana

- 2 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> banana

- 3 frutas = ["manzana", "banana", "cereza"]
 ↑
 for fruta in frutas:
 print(fruta) --> cereza



ACUMULABLES

Un acumulador es una variable que se utiliza para almacenar y acumular valores durante la ejecución de un bucle o de una operación. Generalmente, se usa para sumar, multiplicar o concatenar valores de una colección de datos.

"En los acumuladores, el valor inicial depende de la operación que se deseé realizar. Para una suma, siempre se inicia en 0, ya que este es el elemento neutro de la suma (cualquier número sumado a 0 no cambia su valor)."

"En el caso de una multiplicación, el acumulador comienza en 1, porque este es el elemento neutro de la multiplicación (cualquier número multiplicado por 1 permanece igual)."



Ejemplo:

```
numeros = [1, 2, 3, 4, 5]
suma = 0
for numero in numeros:
    suma += numero
print("La suma es:", suma)
```

```
numeros = [1, 2, 3, 4, 5]
producto = 1
for numero in numeros:
    producto *= numero
print("La multiplicación es:", producto)
```



CONTADORES

Un contador es una variable que se utiliza para contar cuántas veces ocurre un evento específico, generalmente incrementando su valor en cada iteración de un bucle. Es común usarlo cuando se quiere contar cuántos elementos cumplen una determinada condición.

```
# Lista de números
numeros = [1, 2, 3, 4, 5, 6]

# Inicializamos el contador
contador = 0

# Iteramos sobre la lista
for numero in numeros:
    if numero % 2 == 0: # Comprobamos si el número es par
        contador += 1 # Incrementamos el contador si es par

# Imprimimos el resultado
print("Cantidad de números pares:", contador)
```



BUCLES

Un bucle while en Python se utiliza para ejecutar repetidamente un bloque de código mientras se cumpla una condición específica. Es ideal cuando no sabes cuántas veces necesitas iterar, pero sí sabes la condición que debe cumplirse.

Bucle while

```
contador = 0  
  
while contador < 5:  
    print("Contador:", contador)  
    contador += 1
```

Break: Rompe el bucle.

```
contador = 0  
  
while True: # Bucle infinito  
    print("Contador:", contador)  
    contador += 1  
  
    if contador == 5:  
        break
```

Salida:

```
Contador: 0  
Contador: 1  
Contador: 2  
Contador: 3  
Contador: 4
```

Continue: Itera de nuevo el bucle

```
contador = 0  
  
while contador < 5:  
    contador += 1  
  
    if contador == 3:  
        continue  
  
    print("Contador:", contador)
```

Salida:

```
Contador: 1  
Contador: 2  
Contador: 4  
Contador: 5
```



BUCLES

El bucle for en Python se utiliza para iterar sobre una secuencia (como listas, tuplas, cadenas, rangos, etc.) y ejecutar un bloque de código para cada elemento de esa secuencia. Es ideal cuando sabes de antemano cuántas iteraciones necesitas o cuando trabajas con colecciones de datos.

Bucle For

```
numeros = [1, 2, 3, 4, 5]  
  
for numero in numeros:  
    print("Número:", numero)
```

Enumerate: indice y valor

```
frutas = ["manzana", "plátano", "cereza"]  
  
for índice, fruta in enumerate(frutas):  
    print(f"Índice: {índice}, Fruta: {fruta}")
```

Salida:

Índice: 0, Fruta: manzana
Índice: 1, Fruta: plátano
Índice: 2, Fruta: cereza

Range: itera sobre el rango asignada

```
for i in range(5):  
    print("i:", i)    Salida:  
                    i: 0  
                    i: 1  
                    i: 2  
                    i: 3  
                    i: 4
```

Range: itera sobre el rango, pero con un razon de 2

```
for i in range(2, 10, 2):  
    print("i:", i)    Salida:  
                    i: 2  
                    i: 4  
                    i: 6  
                    i: 8
```



TEN INGENIO

Para la parte de porcentajes

1 es = al 100%

Si aumenta en 10% algo sería el número $*110\% = 1.1$

Si disminuye en 10% algo sería el
número $*90\% = 0.9$

Precio original

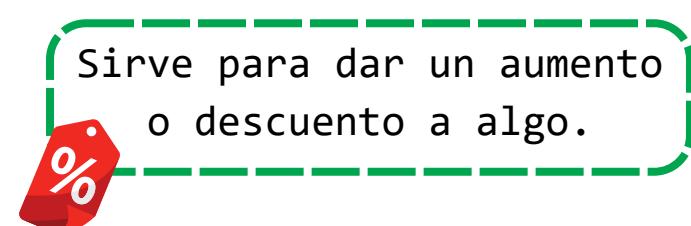
precio_original = 100

Aumento del 10% (multiplicando por 1.10)

precio_con_aumento = precio_original * 1.10

Aplicando descuento del 12% (multiplicando por 0.88)

precio_final = precio_con_aumento * 0.88



TEN INGENIO

Para el resto de la división

Recordar:

$$D(x) = d(x) * q(x) + r(x)$$

$d(x)$ es el dividendo (el número que se divide),

$q(x)$ es el cociente (el resultado de la división),

$r(x)$ es el resto (la parte que queda después de la división).

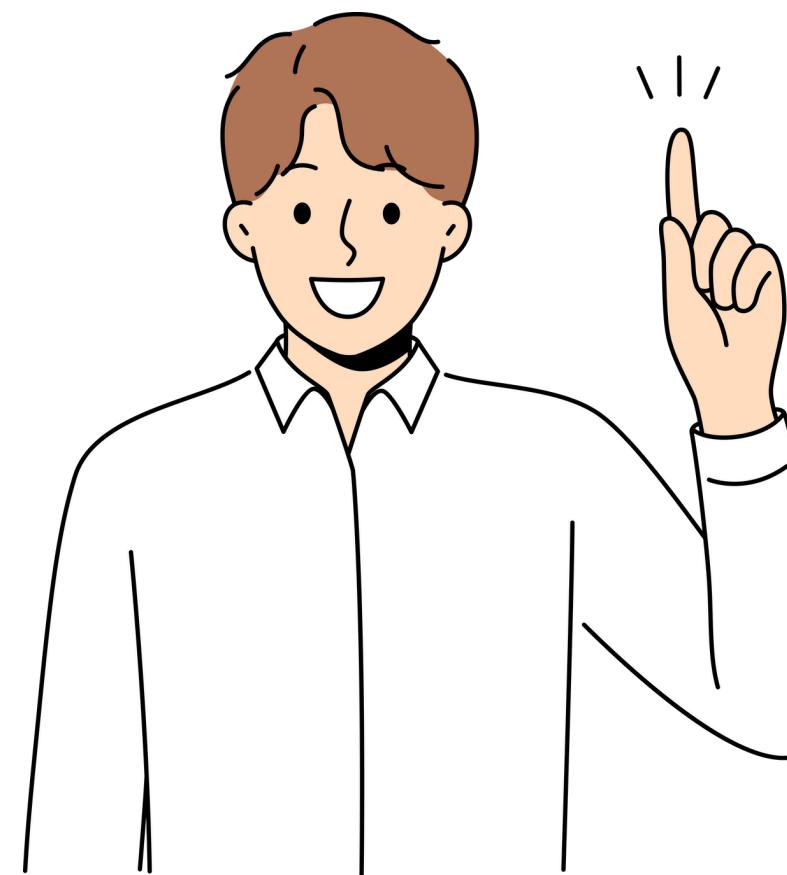
Si divido 2 números puede que me de un resto

$$4 = 2 \times 2 + 0$$

$$3 = 2 \times 1 + 1$$

$$30 = 7 \times 4 + 2$$

Para qué sirve esto, para muchas cosas, como hallar si un número cumple con una condición o no.

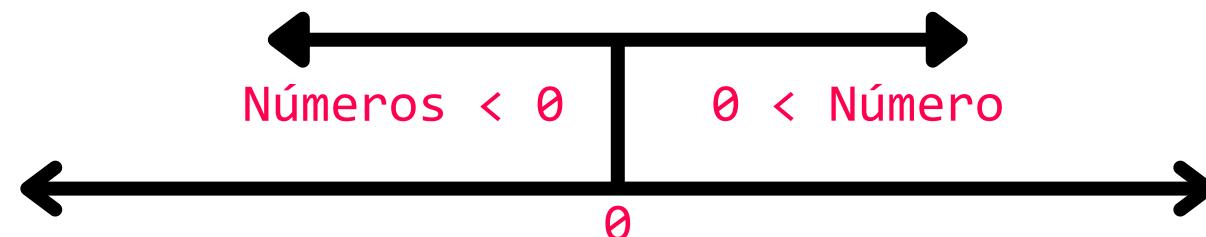


estos se pueden traducir en el cálculo de la división entera y el resto utilizando los operadores de división (//) y módulo (%)



PIENSA EN GRANDE

Positivos y Negativos

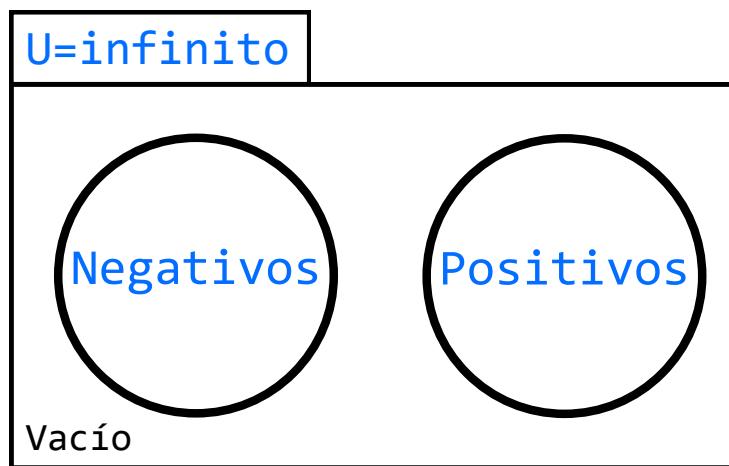


La naturaleza de un número positivo

0 < número

La naturaleza de un número negativo

número < 0



Pares e impares.

Recordar:

$$D(x)=d(x)*q(x)+r(x)$$

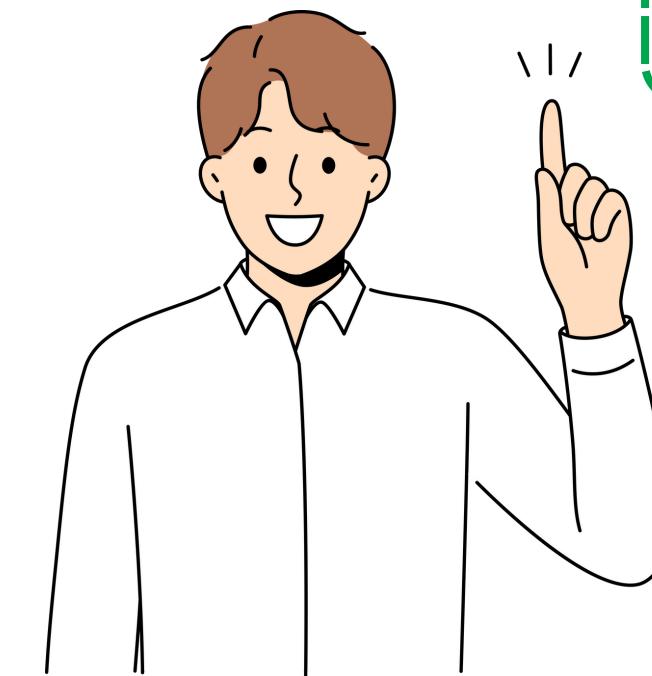
La naturaleza de un número par

número % 2 = 0

La naturaleza de un número impar

número % 2 = 1

El resto, $r(x)$, nos ayuda
a saber la propiedad de un
número.



PIENSA EN GRANDE

Números primos

Los números primos tienen una característica clave:

Solo tienen dos divisores: el 1 y el mismo número. Esto significa que no pueden ser divididos exactamente por ningún otro número.

Ejemplo de números primos:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

Código para encontrar numeros primos

```
contador = 0  
  
numero = int(input("Ingrese un número: "))  
  
for i in range(1, numero + 1):  
    if numero % i == 0:  
        contador += 1  
  
if contador == 2:  
    print("Este número es primo.")  
  
else:  
    print("No es primo.")
```



PIENSA EN GRANDE

Números perfectos

Los números perfectos tienen una característica única:

La suma de sus divisores propios (excluyendo el número en sí mismo) es igual al propio número.

Ejemplos:

6

Divisores propios: 1, 2, 3

Suma: $1+2+3=6$

Por lo tanto, 6 es un número perfecto.

28

Divisores propios: 1, 2, 4, 7, 14

Suma: $1+2+4+7+14=28$

Por lo tanto, 28 es un número perfecto.

Código para encontrar números perfectos

```
divisor = 0
numero = int(input("Ingrese un número: "))

for i in range(1, numero):
    if numero % i == 0:
        divisor += i

if divisor == numero:
    print("Este número es perfecto.")
else:
    print("No es perfecto.")
```



Dame un ejercicio 

EJERCICIOS



PENSAR LOGICAMENTE

Kevin

Hola, tengo una duda, quiero saber la cantidad de números primos dentro de un rango.

Los rangos son 1 y 10

Podrías darme el promedio de esos numeros.

Estudiante

Está bien, dame los 2 rangos.

La cantidad de números primos en ese rango son 4.

El promedio es 4.25

¿Cuántos números primos hay entre 1 y 10?

Sé que un número primo solo tiene 2 divisores, el 1 y el mismo número.

Entonces los números que cumplen con esa condición son 2, 3, 5 y 7, en total 4 números.

El promedio sería sumar 2 + 3 + 5 + 7 luego dividirlo entre 4.



Pero ahora como podría hacer un programa para que Kevin pueda realizar lo que me pidió.

PENSAR LOGICAMENTE

¿Cómo hago el programa?

Primero tengo que pedir los rangos, pero los rangos son 2 números y antes Kevin ya me había pedido sumar 2 números, volveré a utilizar esa lógica.

Entonces el usuario tendría que ingresar el rango inferior y el rango superior, tendré que validar que son números. Y que el rango inferior es menor al rango superior

Ahora dentro de ese rango tengo que hallar los números primos.

Sé que un número primo tiene exactamente 2 divisores: 1 y él mismo. Para determinar si un número es primo, lo dividiré entre los números menores a él y contaré sus divisores. Si solo tiene 2, entonces será primo.

Utilizaré un contador dentro de la condición de números primos para saber la cantidad de primos.

Pero también me piden el promedio, así que utilizaré un acumulador para guardar la suma de los números que cumplan con la condición.

Al tener, por una parte, la cantidad y, por otra parte, la suma, utilizaré esos 2 valores para hallar el promedio, el cual será la división de la suma entre la cantidad.

Ahora teniendo la cantidad y el promedio los mostraré al usuario.



PENSAR LOGICAMENTE



Programa

Kevin

Ingrese el rango inferior.

1

Ingrese el rango superior

100

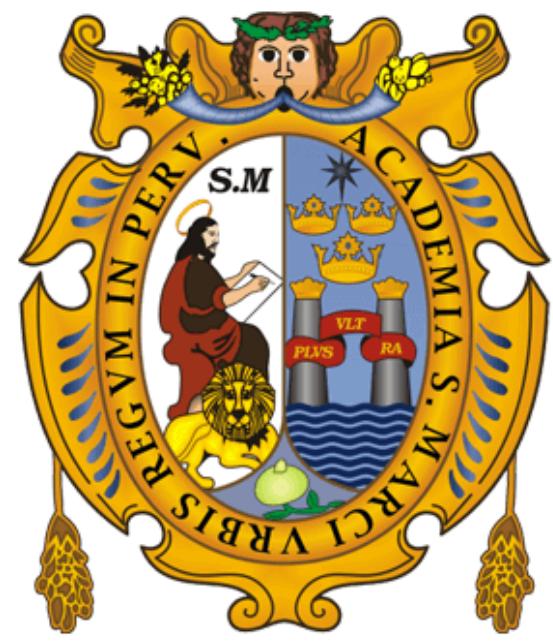
La cantidad es: 25
El promedio es: 42.4



Dame un ejercicio 

CONTACTOS





UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA



Kevin Tupac Agüero



Kev-1729



+51 971427792

