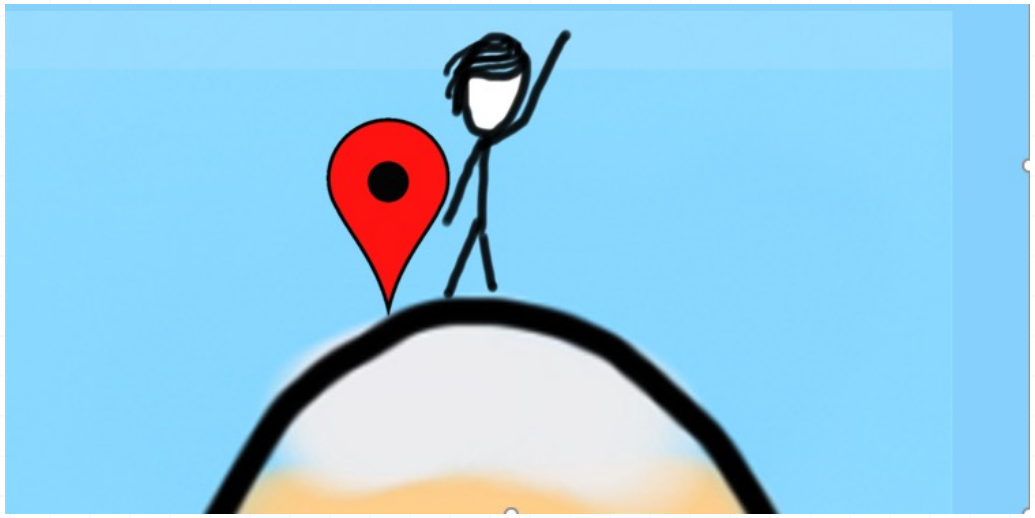
# CS Bridge

## Flujo de control

# Agenda



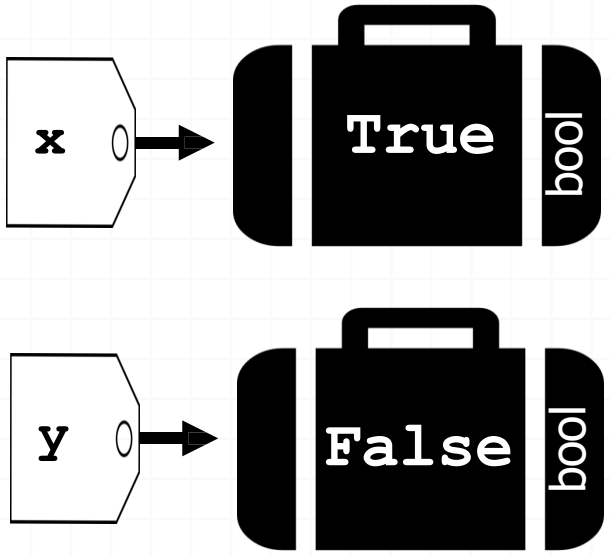
- Variables booleanas
- If/elif/else
- Ciclos While y For

# Variables booleanas

- Solo toman valores **`True`** o **`False`**

```
x = True
y = False
```

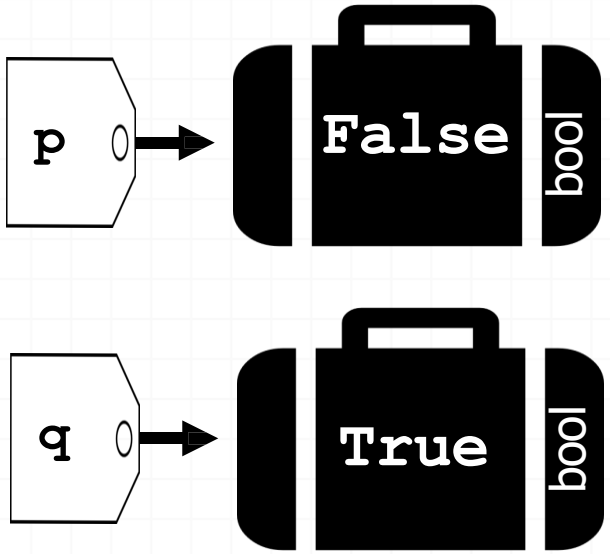
- Representan valores lógicos
- El tipo de estas variables es llamado **bool** en Python



# Variables booleanas

- Una variable puede tomar un valor booleano cuando hacemos comparaciones
  - Esto se llama “expresión booleana”

```
p = 5.0 < 4.0
q = 2 > 1
```



# If/Else

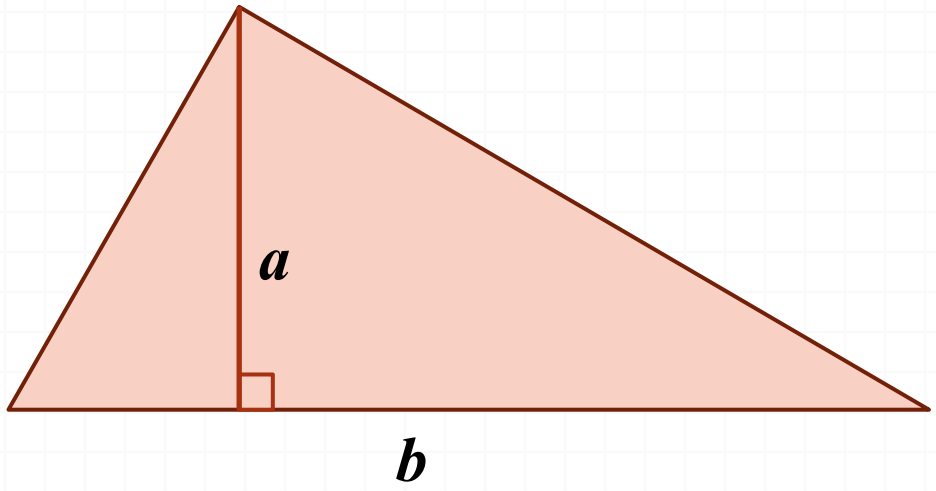
Condicionales

# Programa de ejemplo 1

Area of a triangle

# Área de un triángulo

- ¿Cuál es el área del triángulo?




$$Area = \frac{ba}{2}$$

# Área de un triángulo

```
b= float(input('Introduzca la longitud de la base:  '))
a= float(input('Introduzca el alto:  '))
print('')
area= b*a/2
print('El área del triángulo con b=',b,'y a=',a,'es',area)
```

**Result:**

```
Introduzca la longitud de la base:  8

Introduzca el alto:  4.5

El área del triángulo con b= 8.0 y a= 4.5 es 18.0
```

# Área de un triángulo

¿Qué pasa si el usuario digita un valor negativo?

**Ejemplo:**

```
Introduzca la longitud de la base:  4

Introduzca el alto:  -5

El área del triángulo con b= 4.0 y a= -5.0 es -10.0
```

**Ejemplo:**

```
Introduzca la longitud de la base:  -4

Introduzca el alto:  5

El área del triángulo con b= -4.0 y a= 5.0 es -10.0
```

# Valores inválidos

- No podemos evitar que el usuario digite valores negativos (inválidos)
- Pero podemos detectarlos y escoger no hacer nada con ellos
- Esto requiere escribir un programa con *condicionales*
- En lenguajes de programación esto se logra con **IF**
- Un IF incluye una expresión booleana (devuelve **TRUE** o **FALSE)**

# Operadores lógicos

- Operador **NOT** se usa sobre una proposición lógica
- Operadores **AND** y **OR** se usan sobre dos proposiciones lógicas
- Todos tres devuelven **TRUE** o **FALSE**

| p | q | not p | p and q | p or q |
|---|---|---|---|---|
| **False** | **False** | True | False | False |
| **False** | **True** | True | False | True |
| **True** | **False** | False | False | True |
| **True** | **True** | False | True | True |

# EJERCICIO

Tenemos tres proposiciones P, Q y R:

- P: Hoy está haciendo calor
- Q: Hoy es martes
- R: Hoy es un día festivo

- Q: Hoy es martes.
- not P
- not Q:
- P and Q
- Q and not R
- P or R

# Operadores relacionales

- Comparan dos operadores **comparables** de cualquier tipo (enteros, flotantes, cadenas, etc.)
- Todos ellos devuelven **TRUE** o **FALSE.**

| | | |
|---|---|---|
| Es igual que | == | x == y es True si x es igual a y |
| Es diferente de | != | x != y es True si x es diferente de y |
| Es menor que | < | x < y es True si x es menor que y |
| Es mayor que | > | x > y es True si x es mayor que y |
| Es menor o igual que | <= | x <= y es True si x es menor o igual que y |
| Es mayor o igual que | >= | x >= y es True si x es mayor o igual que y |

N2

# EJERCICIOS

Para:

→ a = 45

→ b = 30

→ c = 10

Escribe cuál es el resultado de las siguientes instrucciones:

✓ a+c >= b

✓ b-c < a-b

# = vs ==

En python:

==

Es un operador de comparación

=

Es usado para asignar un valor a una variable

# Ejemplo

```
if 1 < 2 :
   print("1 es menor que 2")
```

```
num = int(input(“Digite un número: "))
if num == 0:
   print(“El número digitado es 0")
else :
   print(“El número digitado no es 0.")
```

# Opuesto de una expresión lógica

- Supongamos que tenemos una expresión lógica de la forma:

$$p \otimes q$$

  donde $\otimes$ representa ya sea un **and** u **or**.

- El opuesto de esta expresión es:

$$\text{not } (p \otimes q)$$

- Lo cual equivale a:

$$(\text{not } p)\ (\text{not } \otimes)\ (\text{not } q)$$

# Opuesto de una expresión lógica

- ¿Cuáles son los opuestos de estas expresiones?
  - a>2
  - a==0 or b<5
  - c>4 and es_par
  - f==1 or f==2 or f==3

a<=2

a!=0 and b>=5

# Opuesto de una expresión lógica

- ¿Cuáles son los opuestos de estas expresiones?

| | |
|---|---|
| a>2 | a<=2 |
| a==0 or b<5 | a!=0 and b>=5 |
| c>4 and es_par | c<=4 or not es_par |
| f==1 or f==2 or f==3 | f!=1 and f!=2 and f!=3 |

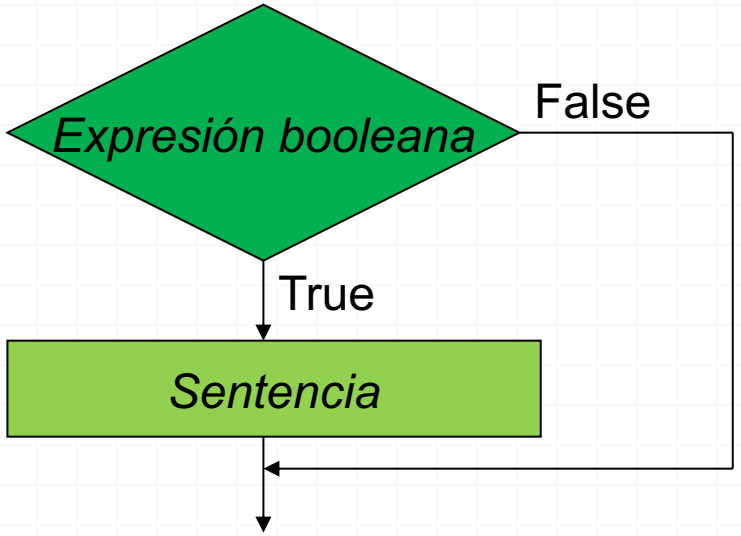
# Precedencia de operadores

- Paréntesis ( ( ) )
- Exponenciación (**)
- Unarios (+) y (-)
- Multiplicación (*), división (/), división entera (//), módulo o resto (%)
- Suma (+), resta (-)
- Operadores relacionales (<, <=, >, >=, ==, !=)
- NOT (not)
- AND (and)
- OR (or)

# Condicionales

- La mayor fortaleza de los programas es hacer cálculos rápidamente
- Su segunda fortaleza es tomar decisiones usando condicionales
- La parte principal de un condicional es una expresión booleana que produce TRUE o FALSE
- Exploremos las diferentes formas de construir condicionales

# Sentencia IF (1)



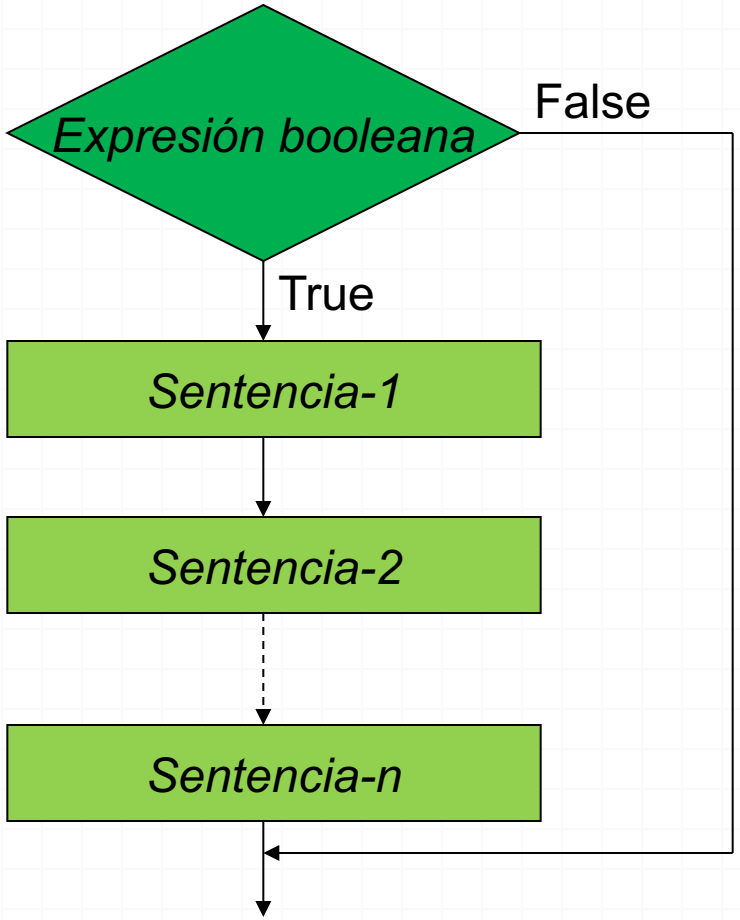

```
if expresión booleana:
    sentencia
```

# Ejemplo de una sentencia IF (1)

```
nota= int(input('Digite la nota de su examen: '))

if nota>=90:
    print('Bien hecho')
```

# Sentencia IF (2)

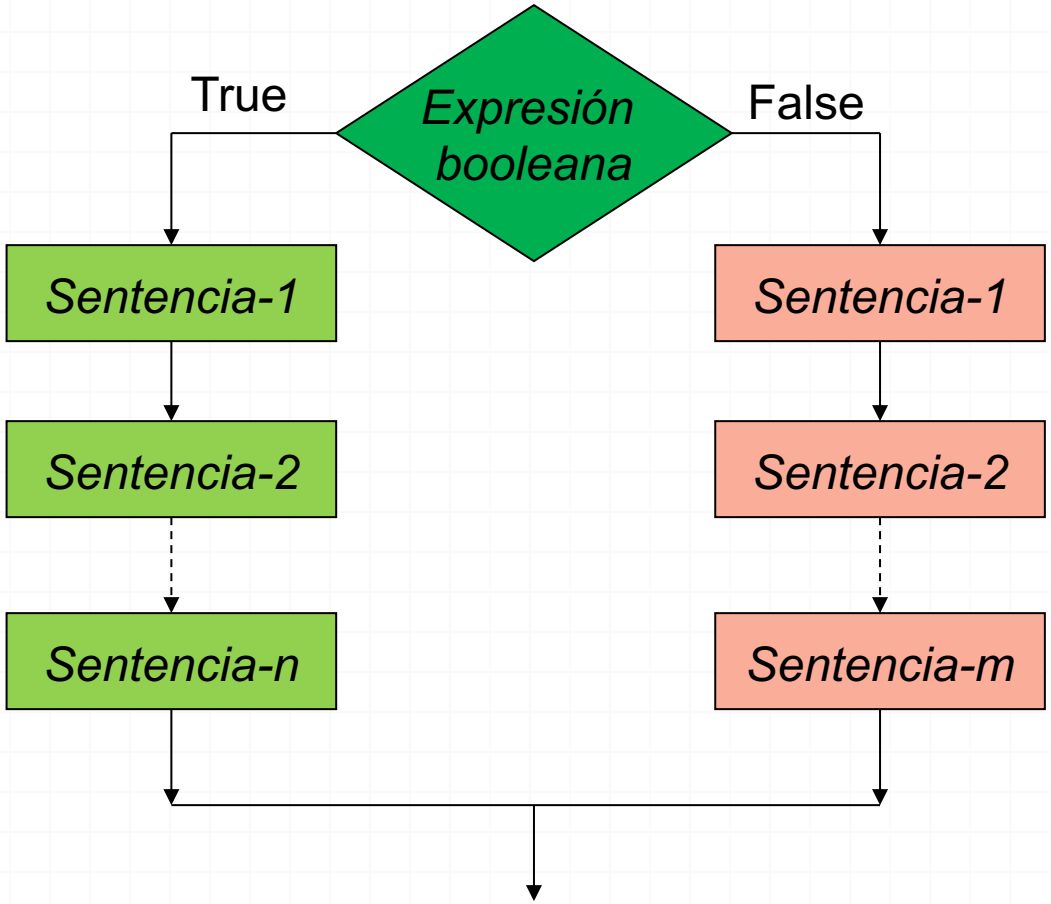



```
if expresión booleana:
    sentencia-1
    sentencia-2
    ...
    sentencia-n
```

# Ejemplo de una sentencia IF (2)

```
nota= int(input(‘Digite la nota de su examen: '))
if nota>=90:
    print(‘Bien hecho')
    print(‘Eres un estudiante excelente')
```

# Sentencia IF (3)

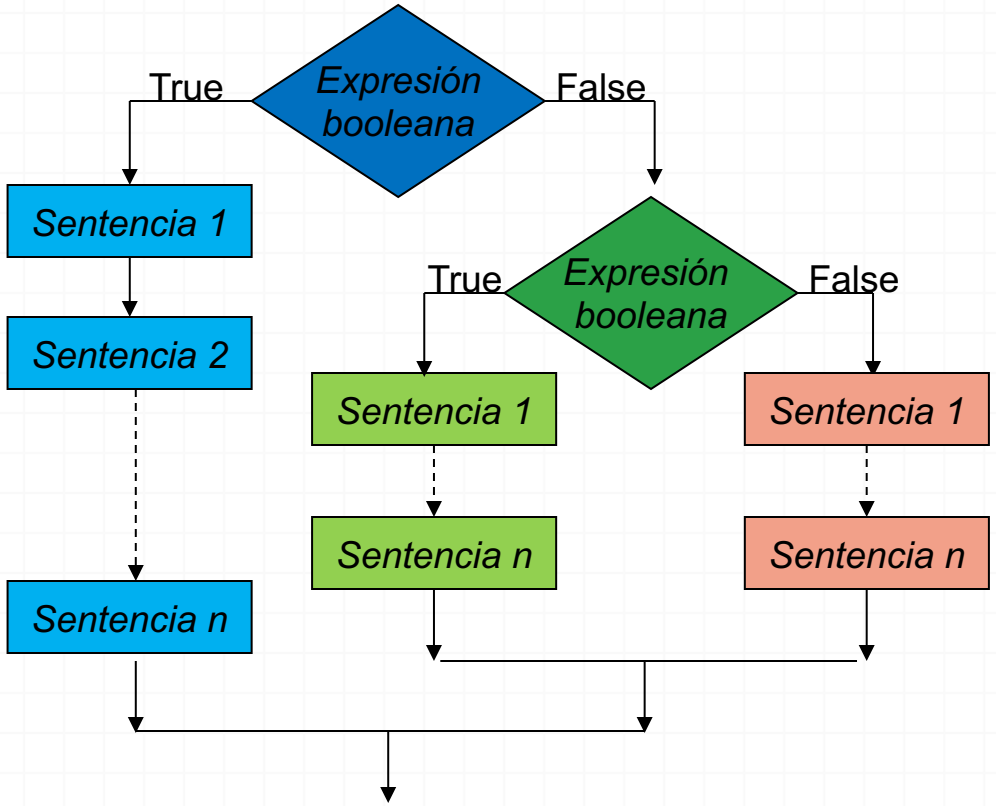



```
if  expresión booleana:
        sentencia-1
        ...
        sentencia-n
else:
        sentencia-1
        ...
        sentencia-m
```

# Ejemplo de sentencia IF (3)

```
nota= int(input('Digite la nota de su examen: '))
if nota>=50:
    print('Pasaste.')
else:
   print('Fallaste.')
   print('Esfuérzate más la próxima vez.')
```

# Sentencia IF (4)




```
if expresión booleana:
    sentencia-1
    sentencia-2
    ...
    sentencia-n
elif expresión booleana:
    sentencia-1
    ...
    sentencia-n
else:
    sentencia-1
    ...
    sentencia-n
```

# IF statement (4) example

```
edad= int(input('Digita tu edad: '))
if edad<13:
    print('Eres un niño.')
elif edad>=18:
    print('Eres un adulto.')
else:
    print('Eres un adolescente.')
```

# Formato general del IF

```
if <expresión booleana>:
    < sentencias >
elif <expresión booleana>:
    < sentencias >
elif <expresión booleana>:
    < sentencias >
elif <expresión booleana>:
    < sentencias >
…
else:
    < sentencias >
```

Se puede colocar cualquier cantidad de ELIF en un bloque IF

Si hay una parte ELSE, debe ser siempre la última rama

Siempre hay una expresión booleana en una línea ELIF pero nunca en el ELSE

**ELSE** se interpreta como si todas las expresiones booleanas previas fallaron entonces haga esto

# Programa de ejemplo 1

Área del triángulo mejorada

# Área de un triángulo

Estábamos discutiendo este problema…

**Ejemplo:**

```
Introduzca la longitud de la base:  4

Introduzca el alto:  -5

El área del triángulo con b= 4.0 y a= -5.0 es -10.0
```

**Ejemplo:**

```
Introduzca la longitud de la base:  -4

Introduzca el alto:  5

El área del triángulo con b= -4.0 y a= 5.0 es -10.0
```

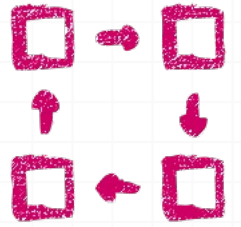
Arreglémoslo con un IF

# Programa de ejemplo 2 (While)

Imprime “Python es genial” 100 veces

# ¿CUÁNDO USAMOS INSTRUCCIONES ITERATIVAS?

- ✓ Puede ser un número fijo de veces o mientras se cumpla una condición
- ✓ Cuando necesitamos repetir varias veces un conjunto de instrucciones



# INSTRUCCIÓN WHILE

Mientras se cumple la condición lógica (booleana)

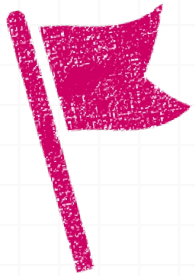
```
while condición:
    acción
    acción
    . . .
    acción
```

Se ejecutan estas acciones. No hay límite en la cantidad de instrucciones que componen el cuerpo de un while.

# Programa de ejemplo 3 (While)

Suma los números que digite el usuario hasta que introduzca un -1

# INSTRUCCIÓN WHILE CON CENTINELA



El centinela es una variable con la que vamos a controlar la continuación y salida del ciclo. Por eso es natural que sea una variable booleana, pero podría ser de otro tipo. En este

# Programa de ejemplo 2 (For range)

Imprime “Python es genial” 100 veces

# CUÁNDO SE PUEDE USAR FOR-IN EN VEZ DE WHILE

Podemos expresar de forma compacta este tipo de ciclos con un for-in siguiendo este otro patrón:

```
for i in range(valor_inicial, valor_final + 1):
    acciones
```

La función range devuelve una secuencia de valores entre un valor inicial y un valor final (sin incluirlo dentro de la secuencia)

# ¿Cuándo usar While o For?

- Cuando queramos repetir algo un número fijo de veces
  - For range
  - While
- Mientras se cumpla una condición
  - While (centinela)

¡Piensen en ejemplos!

# Forma corta de asignación

```
num1 = 5
num2 = 2
num3 = 1.9
```

| | | | | |
|---|---|---|---|---|
| **num1 = num1 + 1** | | **igual a** | | **num1 += 1** |
| **num2 = num2 - 4** | | **igual a** | | **num2 -= 4** |
| **num3 = num3 * 2** | | **igual a** | | **num3 *= 2** |
| **num1 = num1 / 2** | | **igual a** | | **num1 /= 2** |

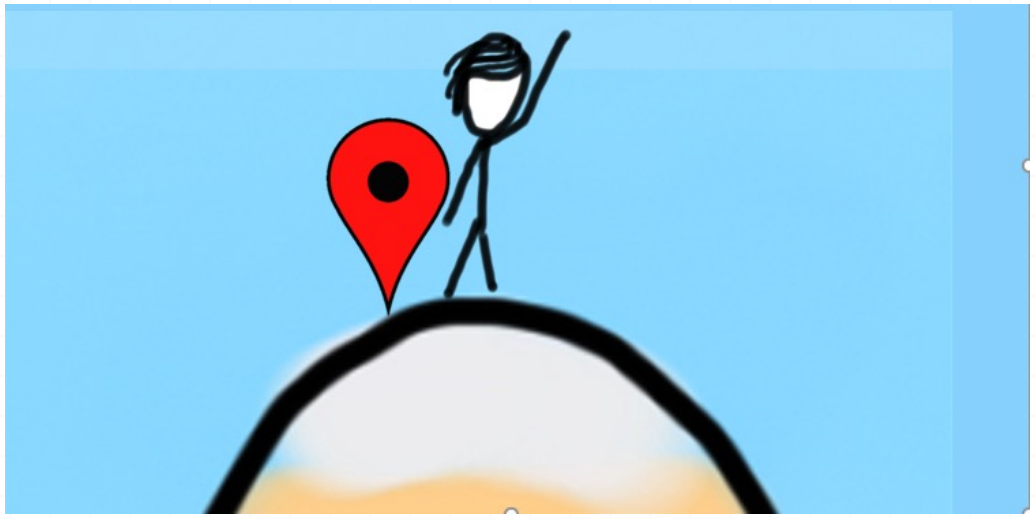
- Generalizando:

  ***variable = variable operador (expresión)***

  Igual a:

  ***variable operador= expresión***

# ¿Qué revisamos en este rato?



- **Variables booleanas**
- **If/elif/else**
- **Ciclos While y For**

# Preguntas