

NIVEL 2

LEYES DE “DE MORGAN”



LEYES DE DE MORGAN

- ✓ Son reglas de transformación que cuando tenemos expresiones booleanas muy complicadas, resultan muy útiles

$$\begin{aligned}\text{not } (x \text{ and } y) &== (\text{not } x) \text{ or } (\text{not } y) \\ \text{not } (x \text{ or } y) &== (\text{not } x) \text{ and } (\text{not } y)\end{aligned}$$

EJEMPLO

Supongamos que en nuestro juego solo podemos matar al dragón si nuestro sable de luz mágica tiene 90% o más de batería y tenemos 100 o más unidades de energía en nuestro escudo protector. Encontramos la siguiente función Python en el juego:

```
EjemploLeyesMorgan.py x
1 def mision_rescate (carga_sable: int, energia_escudo: int)->None:
2     if not ((carga_sable >= 90) and (energia_escudo >= 100)):
3         print("Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!")
4     else:
5         print("Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!")
6
```

Resultado de la ejecución

```
In [30]: mision_rescate(80, 120)
Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!

In [31]: mision_rescate(200, 50)
Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!

In [32]: mision_rescate(200, 101)
Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!
```

ANALICEMOS LA EXPRESIÓN



EjemploLeyesMorgan.py

```
1 def mision_rescate (carga_sable: int, energia_escudo: int)->None:
2     if not ((carga_sable >= 90) and (energia_escudo >= 100)):
3         print("Tu ataque no tiene efecto, el dragón te va a fre
4     else:
5         print("Has logrado arrugar al dragón. ¡Puedes rescatar
6
```

La negación
not
engloba
toda la
expresión

El operador
lógico es un
and

Los
operadores
relacionales
son **>=**

APLICANDO LAS LEYES DE DE MORGAN Y LOS OPUESTOS LÓGICOS

¿Es la expresión transformada más sencilla de entender?



```
def mision_rescate_2 (carga_sable: int, energia_escudo: int)->None:
    if (carga_sable < 90) or (energia_escudo < 100):
        print("Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!")
    else:
        print("Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!")
```

Resultado de
la ejecución

```
In [35]: mision_rescate_2(80, 200)
Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!

In [36]: mision_rescate_2(100, 200)
Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!
```

ANALICEMOS LA TRANSFORMACIÓN



Usando la ley de Morgan $\text{not } (x \text{ and } y) == (\text{not } x) \text{ or } (\text{not } y)$ y los opuestos lógicos

EjemploLeyesMorgan.py

```
1 def mision_rescate (carga_sable: int, energia_escudo: int)->Non
2     if not ((carga_sable >= 90) and (energia_escudo >= 100)):
3         print("Tu ataque no tiene efecto, el dragón te va a fre
4     else:
5         print("Has logrado arrugar al dragón. ¡Puedes rescatar
6
def mision_rescate_2 (carga_sable: int, energia_escudo: int)->Non
1     if (carga_sable < 90) or (energia_escudo < 100):
2         print("Tu ataque no tiene efecto, el dragón te va a freir
3     else:
4         print("Has logrado arrugar al dragón. ¡Puedes rescatar a
```

OTRA OPCIÓN

También podemos invertir el orden del **if** y del **else**, si nos es más fácil de expresar o entender:



```
def mision_rescate_3 (carga_sable: int, energia_escudo: int)->None:
    if (carga_sable >= 90) and (energia_escudo >= 100):
        print("Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!")
    else:
        print("Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!")
```

Resultado de
la ejecución

```
Terminal 1/A ✕

In [40]: mision_rescate_3(80, 200)
Tu ataque no tiene efecto, el dragón te va a freir hasta dejarte crujiente!

In [41]: mision_rescate_3(100, 200)
Has logrado arrugar al dragón. ¡Puedes rescatar a la hermosa princesa!
```


MORALEJA

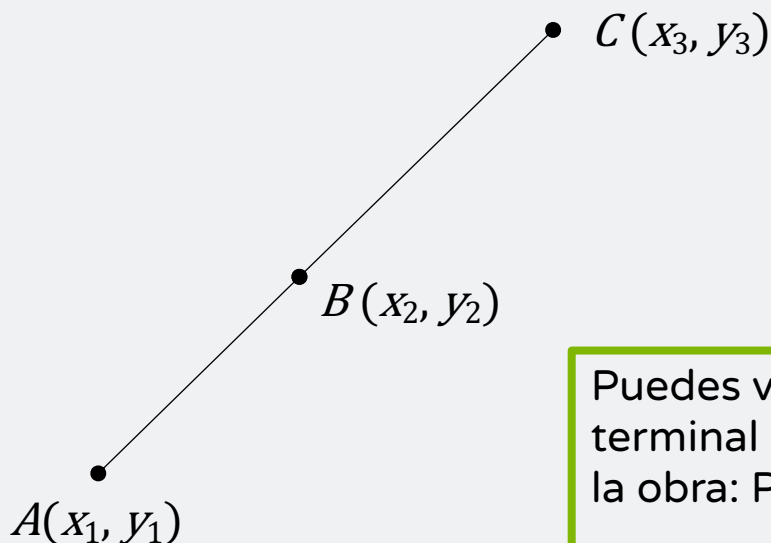


- ✓ La claridad de nuestro código para nosotros mismos y para otras personas siempre debe tener la más alta prioridad. Esto es: que sea fácil ver que el código hace lo que se esperaba
- ✓ A medida que se desarrollamos nuestras habilidades de programación, encontraremos que tenemos más de una forma de resolver cualquier problema

- ✓ Una vez que nuestro programa funcione, debemos tratar de pulirlo:
 - Documentarlo bien = escribir buenos comentarios.
 - Pensar si el código sería más claro con diferentes nombres de variables.
 - Preguntarse:
 - ¿Podemos haberlo hecho más elegante?
 - ¿Debemos más bien utilizar una función?
 - ¿Podemos simplificar los condicionales?
 - ...

1. Escriba una función que reciba por parámetro tres puntos (parejas de coordenadas (x,y) de enteros) y retorne (True o False) si los tres están sobre una recta o no.

AYUDA: Los puntos $A(x_1, y_1)$, $B(x_2, y_2)$ y $C(x_3, y_3)$ están alineados siempre que los vectores \overline{AB} y \overline{BC} tengan la misma dirección. Esto ocurre cuando las pendientes de las rectas que forman son las mismas:



$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_3 - y_2}{x_3 - x_2}$$

Puedes verificar tus resultados usando la terminal presente en la actividad “Manos a la obra: Puntos colineales” en Brightspace

2. La persona que vende pasajes aéreos en la compañía de turismo «La Libertad» pierde mucho tiempo calculando el precio de los pasajes. Esto se debe a que el precio del pasaje depende de: una tarifa básica, la temporada, la compañía aérea, la edad del pasajero y si este es estudiante o no. Usted deben hacer un programa que ayude a calcular el precio de un pasaje Bogotá-Tokio, teniendo en cuenta que:
- ✓ La compañía «ALAS» incrementa el valor de sus pasajes en un 30% en alta temporada mientras que la compañía «VOLAR» lo incrementa en solo 20%
 - ✓ Ambas compañías descuentan el 50% si el pasajero es menor de edad; además, la compañía «VOLAR» tiene un recargo de \$100000 para los pasajeros mayores de 60 años para cubrir el seguro de vida
 - ✓ Los estudiantes que viajan por «ALAS» y que no son menores de edad, tienen un descuento del 10% en temporada baja
 - ✓ La tarifa básica Bogotá-Tokio reglamentaria es de 5 millones de pesos

AYUDA: Antes de escribir su programa, siga los siguientes pasos:

- Defina cuáles son los datos de entrada y de salida, con sus respectivos tipos
- Determine cuáles son las condiciones sobre los datos de entrada, que intervienen en el cálculo del precio del pasaje



Puedes verificar tus resultados usando la terminal presente en la actividad “Manos a la obra: Precio pasaje Bogotá-Tokyo” en Brightspace