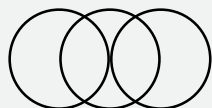
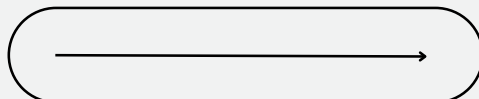


# TEAM



# TRAINING BOOK

## 1 \* EL FAMOSO "FIZZ BUZZ"

1

\* Reto #1

\* Dificultad: FÁCIL

\* Enunciado: Escribe un programa que muestre por consola (con un print) los números de 1 a 100 (ambos incluidos y con un salto de línea entre cada impresión), sustituyendo los siguientes:

\* - Múltiplos de 3 por la palabra "fizz".

\* - Múltiplos de 5 por la palabra "buzz".

\* - Múltiplos de 3 y de 5 a la vez por la palabra "fizzbuzz".

## 2 \* ¿ES UN ANAGRAMA?

2

\* Reto #2

\* Dificultad: MEDIA

\* Enunciado: Escribe una función que reciba dos palabras (String) y retorne verdadero o falso (Boolean) según sean o no anagramas.

\* Un Anagrama consiste en formar una palabra reordenando TODAS las letras de otra palabra inicial.

\* NO hace falta comprobar que ambas palabras existan.

\* Dos palabras exactamente iguales no son anagrama.

\* Reto #3

\* Dificultad: FÁCIL

\* Enunciado: Crea UNA ÚNICA FUNCIÓN (importante que sólo sea una) que sea capaz de calcular y retornar el área de un polígono.

\* - La función recibirá por parámetro sólo UN polígono a la vez.

\* - Los polígonos soportados serán Triángulo, Cuadrado y Rectángulo.

\* - Imprime el cálculo del área de un polígono de cada tipo.

\* Reto #4

\* Dificultad: MEDIA

\* Enunciado: Crea una función que evalúe si un/a atleta ha superado correctamente una carrera de obstáculos.

\* - La función recibirá dos parámetros:

\* - Un array que sólo puede contener String con las palabras "run" o "jump"

\* - Un String que represente la pista y sólo puede contener "\_" (suelo) o "|" (valla)

\* - La función imprimirá cómo ha finalizado la carrera:

\* - Si el/a atleta hace "run" en "\_" (suelo) y "jump" en "|" (valla) será correcto y no variará el símbolo de esa parte de la pista.

\* - Si hace "jump" en "\_" (suelo), se variará la pista por "x".

\* - Si hace "run" en "|" (valla), se variará la pista por "/".

\* - La función retornará un Boolean que indique si ha superado la carrera.

\* Para ello tiene que realizar la opción correcta en cada tramo de la pista.

\* Reto #5

\* Dificultad: FÁCIL

\* Enunciado: Crea una función que reciba dos array, un booleano y retorne un array.

\* - Si el booleano es verdadero buscará y retornará los elementos comunes de los dos array.

\* - Si el booleano es falso buscará y retornará los elementos no comunes de los dos array.

\* - No se pueden utilizar operaciones del lenguaje que lo resuelvan directamente.

\* Reto #6

\* Dificultad: MEDIA

\* Enunciado: Crea un programa que calcule quien gana más partidas al piedra, papel, tijera.

\* - El resultado puede ser: "Player 1", "Player 2", "Tie" (empate)

\* - La función recibe un listado que contiene pares, representando cada jugada.

\* - El par puede contener combinaciones de "R" (piedra), "P" (papel) o "S" (tijera).

\* - Ejemplo. Entrada: [("R","S"), ("S","R"), ("P","S")]. Resultado: "Player 2".

\* Reto #7

\* Dificultad: MEDIA

- \* Enunciado: ¡La Tierra Media está en guerra! En ella lucharán razas leales a Sauron
- \* contra otras bondadosas que no quieren que el mal reine sobre sus tierras.
- \* Cada raza tiene asociado un "valor" entre 1 y 5:
- \* - Razas bondadosas: Pelosos (1), Sureños buenos (2), Enanos (3), Númenóreanos (4), Elfos (5)
- \* - Razas malvadas: Sureños malos (2), Orcos (2), Goblins (2), Huargos (3), Trolls (5)
- \* Crea un programa que calcule el resultado de la batalla entre los 2 tipos de ejércitos:
- \* - El resultado puede ser que gane el bien, el mal, o exista un empate. Dependiendo de la
- \* suma del valor del ejército y el número de integrantes.
- \* - Cada ejército puede estar compuesto por un número de integrantes variable de cada raza.
- \* - Tienes total libertad para modelar los datos del ejercicio.
- \* Ej: 1 Peloso pierde contra 1 Orco. 2 Pelosos empatan