

## Desafío - Arreglos y archivos

### Instrucciones

A continuación se detallan variados desafíos a desarrollar. Para su correcta evaluación, los programas deben ser almacenados en un archivo comprimido `.zip` de la siguiente manera:

```
desafios.zip
├─ calculo_notas2.rb
├─ calculo_notas.rb
├─ grafico.rb
└─ proyecciones.rb
```

### Desafío 1

Crear el programa `proyecciones.rb` además tendremos un archivo con las proyecciones de venta de cada mes del próximo año. Este archivo debe llamarse `ventas_base.db` y estar en la misma carpeta de trabajo que el código. Para realizar el ejercicio puedes agregar como contenido el siguiente.

```
300070,50520,35000,32810,47999,62050,89100,21000,100010,121900,99549,210000
```

El desafío consiste en hacer 2 simulaciones.

- Las ventas totales dado que en la primera mitad del semestre se vende un 10% más.
- Las ventas totales dado que en la segunda mitad del semestre se vende un 20% más.

El resultado debe ser presentado en un archivo llamado `resultados.data`. Los datos deben contener máximo 2 decimales y estar cada uno en una sola línea.

Uso:

```
ruby proyecciones.rb
```

Contenido `resultados.data`

```
231231.32
879879.43
```

Tips:

- El output en pantalla no es necesario pero no afecta la evaluación, se debe crear el archivo resultados.data y debe cumplir con el formato establecido.
- Tener cuidado con los datos de entrada, son strings y deben ser tratados como float.
- Se recomienda crear un método que reciba cuatro argumentos:
  - El arreglo con los datos.
  - El aumento
  - Desde qué índice se debe leer el arreglo
  - Hasta qué índice se debe leer el arreglo
- Investigar respecto a `ruby format number`
- El archivo con el que se probará el ejercicio tendrá distintos datos.

## Desafío 2

---

Se tiene un archivo `notas.data` con las notas de un curso

```
Javiera,9,5,3,9
Francisca,8,3,5,5
Juan,9,5,5,9
Pedro,5,4,6,8
Cecilia,8,7,8,8
```

Se pide:

- Crear un archivo llamado `calculo_notas.rb`.
- Crear un método llamado `nota_mas_alta` dentro del archivo `calculo_notas.rb` que reciba un arreglo con el nombre y notas de un alumno y devuelva la nota mas alta.

Ejemplos:

- `nota_mas_alta(data[0]) => 9`
- `nota_mas_alta(data[1]) => 8`

Tips:

- Devolver no es mostrar en pantalla.
- El archivo no será el mismo mostrado, pueden ser más columnas y más filas.
- El espacio principal del programa no será probado, puedes ocuparlo para probar el código llamando a los métodos, recuerda transformar las notas a número.
- El nombre siempre será el primer elemento.
- Todos los alumnos tendrán la misma cantidad de notas.
- Cuidado con los tipos de datos, el arreglo contiene strings y números.

## Desafío 3

---

Se tiene un archivo `notas.data` con las notas de un curso

```
Javiera,9,5,3,9
Francisca,8,3,5,5
Juan,9,5,5,9
Pedro,5,4,6,8
Cecilia,8,7,8,8
```

Se pide:

- Crear un archivo llamado `calculo_notas2.rb`.
- Crear un método llamado `notas_mas_alta` dentro del archivo `calculo_notas2.rb` que reciba un arreglo con el nombre y notas de un alumno y devuelva una arreglo con la nota mas alta de cada alumno.

Tips:

- Devolver no es mostrar en pantalla.
- El archivo no será el mismo mostrado, pueden ser más columnas y más filas.
- El espacio principal del programa no será probado, puedes ocuparlo para probar el código llamando a los métodos, recuerda transformar las notas a número.
- El nombre siempre será el primer elemento.
- Todos los alumnos tendrán la misma cantidad de notas.
- Cuidado con los tipos de datos, el arreglo contiene strings y números.

## Desafío 4

Crear el archivo `grafico.rb` que contenga el método `chart` que construya un gráfico en la consola a partir de un arreglo con datos. El método no debe devolver ningún valor.

Uso:

```
→ irb

2.6.0 :001 > require_relative "grafico"
=> true
2.6.0 :002 > chart([5, 3, 2, 5, 10])
| *****
| *****
| ****
| *****
| *****
| *****
| *****
>-----
 1 2 3 4 5 6 7 8 9 10
```

Tips:

- Por cada dato se debe imprimir el `|` y una cantidad de `*` equivalente el número o X veces el número.
- Una solución más avanzada podría ajustar la cantidad de `*` a mostrar por números, la solución inicial debe fijar 2 `*` por cada número.
- Es importante encontrar el número máximo.