Actividad 013 - Ruby Hashes

- Para poder realizar este actividad debes haber realizado los cursos previos junto con los videos online correspondientes a la experiencia 7.
- Crea una carpeta y guarda cada archivo .rb con el número de la pregunta, siguiendo las instrucciones de manera local con **Sublime** o **Atom**.
- Luego guarda los cambios y súbelos a tu repositorio de Github.
- Luego de pusheados los últimos cambios, sube el link de Github en el desafío de la sección correspondiente en la plataforma.

Ejercicio 1: Sintaxis básica

1. Dado el siguiente hash:

```
h = { :claveuno => 10, :clavedos => 20, :clavetres => 30 }
```

   Modificar el hash para utilizar la sintaxis de Ruby 2.0+

Ejercicio 2: Corrección de errores

1. Se tiene el siguiente hash:

```
productos = {'bebida' => 850, 'chocolate' => 1200,
'galletas' => 900, 'leche' => 750}
```

   y se realiza la siguiente consulta para conocer los productos existentes:

```
Productos.each { |valor, producto| puts producto }
```

   Corrige el error para mostrar la información solicitada.

2. Se quiere agregar un nuevo producto al hash:

```
producto[2200] = cereal
```

Corrige la instrucción para agregar el producto.

3. Se quiere actualizar el precio de la bebida:

```
producto[:bebida] = 2000
```

Corrige la instrucción para actualizar el valor del producto existente.

4. Convertir el hash en un array y guardarlo en una nueva variable.
5. Eliminar el producto 'galletas' del hash.

Ejercicio 3: Operaciones básicas

Dado el hash:

```
h = {"x": 1, "y":2}
```

- Agregar el string z con el valor 3.
- Cambiar el valor de x por 5.
- Eliminar la clave y.
- Si el hash tiene una clave llamada z mostrar en pantalla "yeeah".
- Invertir el diccionario de forma que los valores sean las llaves y las llaves los valores
    - Ejemplo:

```
x = {"a": "hola" }
```

Se transforme en:

```
x = {"hola": "a"}
```

Ejercicio 4: Array y Hashes

Se tienen dos arrays uno con el nombre de personas y otro con las edades, se pide generar un hash con el nombre y edad de cada persona (se asume que no existen dos personas con el mismo nombre)

```
personas = ["Carolina", "Alejandro", "Maria Jesús", "Valentín"]
edades = [32, 28, 41, 19]
```

1. Se pide generar un hash con la información:

   ```
   personas_hash = {"Carolina": 32, "Alejandro":28,
   "María Jesús":41, "Valentín":19}
   ```

2. Crear un método que reciba el hash y devuelva la edad del hash pasado como argumento.

Ejercicio 5: Array y Hashes

Dados los siguientes array:

```
meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo']
ventas = [2000, 3000, 1000, 5000, 4000]
```

Generar un hash que contenga los meses como llave y las ventas como valor:

```
h = {'Enero': 2000, 'Febrero': 3000 ... }
```

En base al hash generado:

1. Invertir las llaves y los valores del hash.
2. Obtener el mes mayor cantidad de ventas (a partir del hash invertido.)

Ejercicio 6: Operaciones típicas sobre un hash

Escribir un hash con el menu de un restaurant, la llave es el nombre del plato y el valor es el precio de este.

```
restaurant_menu = { "Ramen" => 3, "Dal Makhani" => 4, "Coffee" => 2 }
```

1. Obtener el plato mas caro.
2. Obtener el plato mas barato.
3. Sacar el promedio del valor de los platos.
4. Crear un arreglo con solo los nombres de los platos.
5. Crear un arreglo con solo los valores de los platos.
6. Modificar el hash y agregar el IVA a los valores de los platos (multiplicar por 1.19).
7. Dar descuento del 20% para los platos que tengan un nombre de más 1 una palabra.

Ejercicio 7: Ejercicio completo con un hash

Se tiene un hash con el inventario de un negocio de computadores.

```
inventario = {"Notebooks": 4, "PC Escritorio": 6, "Routers": 10,
"Impresoras": 6}
```

Se pide:

- Crear un menú de 4 opciones, es decir, el usuario puede ingresar 1 2 3 o 4 y según eso el programa realizará distintas funciones.
- Si el usuario ingresa 1, podrá **agregar** un item y su stock en un solo string y agregarlo al hash. Para separar el nombre del stock el usuario debe utilizar una coma.
    - Ejemplo del input: "Pendrives, 100"
- Si el usuario ingresa 2, **podrá eliminar** un item.
- Si el usuario ingresa 3, puede **actualizar** la información almacenada (item y stock).
- Si el usuario ingresa 4, podrá ver el **stock total** (suma del stock de cada item) que hay en el negocio.
- Si el usuario ingresa 5, podrá ver el **ítem que tiene la mayor cantidad de stock**.
- Si el usuario ingresa 6 podrá ingresar y preguntarle al sistema si un item **existe en el inventario** o no. Por ejemplo, el usuario ingresará "Notebooks" y el programa responderá "Sí".
- El programa debe repertirse hasta que el usuario ingrese 7 (salir).