Práctica 1

1. Investigá y probá en un intérprete de Ruby (irb, por ejemplo) cómo crear objetos de los siguientes tipos básicos, tanto mediante el uso de literales como utilizando el constructor new(cuando sea posible).

- Symbol
  - Un Symbol es un tipo de dato inmutable y liviano que se utiliza para representar nombres, identificadores o "etiquetas" en Ruby.
  - Los Symbol en Ruby no se crean con new, solo mediante literales o usando el método to_sym.
  - Con literal
    - `:mi_simbolo`
  - Con to_sym
    - `"mi_simbolo".to_sym`
- String
  - Con literal
    - `"hola mundo"`
    - `'hola mundo'`
  - Con constructor
    - `String.new("hola mundo")`
- Array
  - Con literal
    - `[1, 2, 3]`
  - Con constructor
    - `Array.new` `# Array vacío`
    - `Array.new([1, 2, 3])` `# Inicializa con copia del array dado`
    - `Array.new([array])` `# Inicializa con copia del array dado`
    - `Array.new(3, "a")` `# ["a", "a", "a"]`
- Hash

- Un hash es una estructura de datos que almacena pares de clave-valor (key-value), permitiéndote acceder rápidamente a un valor dado su clave.
- Características principales
  - Cada clave es única dentro del hash.
  - El valor asociado puede ser de cualquier tipo.
  - Las claves suelen ser symbols o strings, pero pueden ser de cualquier tipo de objeto comparable.
- Con literal
  - `{clave: "valor", otro: 2}`
  - `{"clave" => "valor", "otro" => 2}`
- Con constructor
  - `Hash.new  # Hash vacío`
  - `Hash.new("valor_por_defecto")  # Hash con valor por defecto`
  - `Hash[clave: "valor", otro: 2]  # Alternativa a literal`

2. De qué forma(s) se puede convertir un objeto (cualquiera fuere su tipo o clase) en String?

- Metodo to_s
  - Devuelve una representación en texto del objeto:
    - `123.to_s  # => "123"`
    - `:simbolo.to_s  # => "simbolo"`
    - `[1, 2, 3].to_s  # => "[1, 2, 3]"`
    - `{a: 1}.to_s  # => "{:a=>1}"`
- Interpolación de String
  - Cuando se intorpola cualquier objeto dentro de una cadena (usando #{obj}), Ruby llama automáticamente a to_s:
    - `nombre = :juan`
    - `"El nombre es #{nombre}"  # => "El nombre es juan"`
- Método inspect

- Muestra una representación "legible para humanos" del objeto, es muy útil para depuración:
  - `[1, 2, 3].inspect  # => "[1, 2, 3]"`
  - `{a: 1}.inspect  # => "{:a=>1}"`
  - `objeto.inspect  # => "#<Clase:0x0000abcd>"`

3. ¿De qué forma(s) se puede convertir un objeto String en un símbolo?

   - Método .to_sym
     - `"hola".to_sym  # => :hola`
     - `"clave123".to_sym  # => :clave123`
   - Método .intern
     - `"hola".intern  # => :hola`
   - El símbolo creado es inmutable y único.
   - Si el String contiene espacios o caracteres especiales, el símbolo también los tendrá.

4. ¿Qué devuelve la siguiente comparación? ¿Por qué?
   - 'TTPS Ruby'.object_id == 'TTPS Ruby'.object_id

   Retorna falso, esto sucede debido a que cada vez que se escribe un literal de String, Ruby crea un nuevo objeto String en memoria, incluso aunque el texto sea el mismo.

5. Escribí una función llamada reemplazar que, dado un String que recibe como parámetro, busque y reemplace en éste cualquier ocurrencia de "{" por "do\n" y cualquier ocurrencia de "}" por "\nend", de modo que convierta los bloques escritos con llaves por bloques multilínea con do y end.

6. Escribí una función que expresé en palabras la hora que recibe como parámetro, según las siguientes reglas:
   - Si el minuto está entre 0 y 10, debe decir “en punto”,
   - si el minuto está entre 11 y 20, debe decir “y cuarto”,

- si el minuto está entre 21 y 34, debe decir “y media”,
- si el minuto está entre 35 y 44, debe decir “menos veinticinco” con la hora siguiente,
- si el minuto está entre 45 y 55, debe decir “menos cuarto” con la hora siguiente,
- y si el minuto está entre 56 y 59, debe decir “Casi son las” con la hora siguiente

7. Escribí una función llamada contar que reciba como parámetro dos String y que retorne la cantidad de veces que aparece el segundo String en el primero, en una búsqueda case- insensitive (sin distinguir mayúsculas o minúsculas).

8. Modificá la función anterior para que sólo considere como aparición del segundo String cuando se trate de palabras completas.

9. Dada una cadena cualquiera, y utilizando los métodos que provee la clase String, realizá las siguientes operaciones sobre dicha cadena, implementando métodos que funcionen de la siguiente forma:
   - string_reverso: retorna el string con los caracteres en orden inverso.
   - string_sin_espacio: elimina los espacios en blanco que contenga.
   - string_a_arreglo_ascii: retorna un arreglo con cada uno de los caracteres conver tidos a su correspondiente valor ASCII.
   - string_remplaza_vocal: cambia las vocales por números:
     - "a" o "A" por "4",
     - "e" o "E" por "3",
     - "i" o "I" por "1",
     - "o" u "O" por "0",
     - y "u" o "U" por "6".

10. ¿Cuál es el valor de retorno del siguiente código?

```
[:upcase, :downcase, :capitalize, :swapcase].map do |meth|
"TTPS Ruby".send(meth)
end
```

Devuelve:

```
["TTPS RUBY", "ttps ruby", "Ttps ruby", "ttps rUBY"]
```

- Se tiene un arreglo de símbolos con nombres de métodos de String.
- Con map, se recorre cada símbolo dentro del arreglo.
- El bloque do |meth| ... end recibe en meth el elemento actual del array.
    - En este caso, meth será primero :upcase, luego :downcase, después :capitalize, y por último :swapcase.
- "TTPS Ruby".send(meth)
    - Invoca el método cuyo nombre está en meth sobre el string "TTPS Ruby".

11. Tomando el ejercicio anterior como referencia, ¿en qué situaciones usarías los métodos send y public_send definidos en la clase Object? ¿Cuál es la principal diferencia entre esos dos métodos?

    Ambos métodos (send y public_send) permiten invocar un método dinámicamente en un objeto a partir de su nombre (un símbolo o string), incluso pasándole argumentos.

    - send
        - Puede invocar cualquier método, incluso si es private o protected.
        - Se usa mucho en meta-programación, aunque puede romper el encapsulamiento.
    - public_send
        - Solo puede invocar métodos públicos.
        - Si se intenta llamar a un método privado o protegido, lanza un NoMethodError.

12. Escribí una función longitud que dado un arreglo que contenga varios String cualesquiera, retorne un nuevo arreglo donde cada elemento es la longitud del String que se encuentra en la misma posición del arreglo recibido como parámetro.

13. Escribí una función llamada listar que reciba un Hash y retorne un String con los pares de clave/valor formateados en una lista ordenada en texto plano.

14. Mejorar la función anterior en una nueva llamada listar_mejorada para que además reciba opcionalmente un parámetro llamado pegamento (su valor por defecto debe ser ": ") que sea el que utilice para unir los pares de clave/valor.

15. Escribí un método que reciba un argumento y retorne un valor booleano indicando si la cadena recibida como argumento es pentavocálica o panvocálica (contiene todas las vocales). El chequeo no debe ser sensible a minúsculas y mayúsculas.

16. Escribí un script en Ruby que le pida a quien lo ejecute que ingrese su nombre por entrada estándar (el teclado), y que lo utilice para saludarl@ imprimiendo en pantalla.

17. Escribí un nuevo script, que de manera similar al implementado en el punto anterior haga el saludo usando un nombre que se provea, pero que en lugar de solicitar que el nombre se ingrese por entrada estándar, éste se reciba como argumento del script.

18. Implementá las funciones necesarias para que, dado un color expresado en notación RGB, se pueda obtener su representación en las notaciones entera y hexadecimal. La notación entera se define como red + green * 256 + blue * 256 * 256 y la hexadecimal como el resultado de expresar en base 16 el valor de cada color y concatenarlos en orden.

19. Investigá qué métodos provee Ruby para:
    - Obtener la lista de ancestros (superclases) de una clase.
        - Clase.ancestors
    - Conocer la lista de métodos de una clase.
        - Clase.methods
    - Conocer la lista de métodos de instancia de una clase.
        - Clase.instance_methods
    - Conocer las variables de instancia de una clase.
        - Clase.instance_variables

- Obtener el valor de una variable de instancia de un objeto (sin utilizar un método generado con attr_reader o similar) accediéndolo desde fuera de éste.
    - objeto.instance_variable_get(:@nombre_variable)
- Establecer el valor de una variable de instancia de un objeto (sin utilizar un método generado con attr_writer o similar) desde fuera de éste.
    - objeto.instance_variable_set(:@nombre_variable, valor)

attr_reader / attr_writer / attr_accessor → uso normal, seguro y recomendado en POO.

instance_variable_get/set → herramienta de bajo nivel, sirve para metaprogramación, depuración o cuando no hay acceso a los métodos públicos.

20. Escribí una función que encuentre la suma de todos los números naturales múltiplos de 3 y 5 (ambos) que sean menores que un número tope que reciba como parámetro.

21. Creá otra función, similar a la anterior, que busque los números naturales múltiplos de N números enteros que recibirá como parámetro en un arreglo.