1. Print 394 in binary & hex using fmt.Printf

2. Use fmt.Sprintf to string print to a variable the number 394 in hex; Then use fmt.Println to print that variable to standard output.

3. Use fmt.Printf to print the type of a variable

4. Create a variable of type int initialized to its zero value. Use fmt.Scan to receive input from the user @ the terminal & store that input in the variable of type int which you created.

5. Use ~~short hand~~ VAR notation to initialize a variable to zero value of type rune. Assign a rune to the variable. Print the type of the variable. Why is it int32?

(6) Create a loop that prints A lot of UTF-8 characters

(7) Create a variable of type string & use backtick characters to assign a raw string literal to the variable.

(8) Convert a string to a slice of ~~bytes~~ bytes Then print that.

(9) Print the length of a string

(10) Access a rune in a string using an index

(11) Slice a string

(12) Concatenate a string

(13) Use Atoi & Itoa from strconv

14. Convert an int to a float64

15. Convert a float64 to an int

16. Convert a slice of bytes to a string

17. Convert a string to a slice of bytes

18. Create your own type w/ an underlying type of an empty interface. Create a variable of the user defined type you just created. Assign a string to the variable. Use Assertion to prove the underlying type is a string.

19. Create a slice of int. Iterate over the slice using "range" & printing each int.

20. Create a slice of int. Iterate over the slice using a for loop with ~~out~~ init, cond, post expressions. Print each int.

(21) Slice a slice

(22) Create a slice using Make & specifically setting length & capacity.

(23) Append to a slice. Print len & capacity.

(24) Delete from a slice

(25) Create a map that stores a ~~[illegible]~~ words & definitions

(26) update an entry in the Map

(27) Delete an entry in the Map

28. Create a STRUCT to store "person" fields of data

- create a value of that type; store it in a variable
- print the fields in that variable