WKN - Webknot Numbers DB

From the week 2 track you have grasped lots of important low level and core functionalities in Go. To put all these to the test we have decided to give you a task of designing a simple REPL Database - what can be more low level than that.

The database only operates on numbers and has limited functionality. It is similar to SQLite, it uses a single file to store everything, the goal is to have persistent data. The database can store integer arrays and perform some operations on them.

The database should be concurrent and support all the following functions:
(Suppose ***wkn*** is the name of the generated binary)

1] Initialise

```
> wkn new
# This will create a new db file in the cwd and start the REPL. File name should be .wkn. If a .wkn file already exists throw an error - “Db file already present”

> wkn
# Check if there is already a .wkn in the cwd and start the REPL. If file is not present show help for mentioning path to a custom .wkn file.

> wkn --db-path ./path_to_file
# Check if the mentioned file is a valid .wkn file and start the REPL, else throw an error - “File is corrupted”
```

All commands below will work only inside the REPL console once the db is started.

2] Functionality

```
# Create an empty array
wkn> new array_name
CREATED

# Create an array with data, in the output, 8 represents the length of created array
wkn> new array_name 1,2,3,4,7,0,-1,4
CREATED (8)

# Print an existing array - empty one
wkn> show array_1
[ ]

# Print an existing array - with data
wkn> show array_2
[1, 2, 6, 9, 0]
```

```
# Print a non existent array
wkn> show array_3
Error: “array_3” does not exist

# Delete an array
wkn> del array_1
DELETED

# Delete a non existent array
wkn> del array_2
Error: “array_2” does not exist

# Merge Two arrays - the data from 2nd array gets merged to first array, without removing data from 2nd array.
wkn> merge array_1 array_2
MERGED

wkn> merge array_1 array_2
Error: “array_1” does not exist

# To exit
wkn> exit
Bye!

# Unknown command
wkn> sort array_1
Error: “sort” is not a supported operation
```

Important Points to be followed

1] When the db loads, data from the .wkn file should be loaded efficiently into the memory.

2] After each operation, data should be persisted back to the file (The design of the file is up to you, how you want to store data in it).

3] If for any command the required number of arguments or no arguments are passed, show a help on required arguments to run the command. Also make sure that all edge cases and error scenarios are handled properly.

4] Arrays follow the same naming conventions as go lang variable names.

5] Concurrency is the key - if two instances of the same db are opened in two different terminals, both should present the same data. Ensure to handle deadlocks here, if both instances try to modify the same array.

Example

# In terminal 1 	
`wkn> new array_2 1,5,6`
`CREATED (3)`
	# In terminal 2
	`wkn> show array_2`
	`[1, 5, 6]`