ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ PROJECT

ΜΑΛΩΝΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
ID: 7115112200020
EMAIL: cs22200020@di.uoa.gr

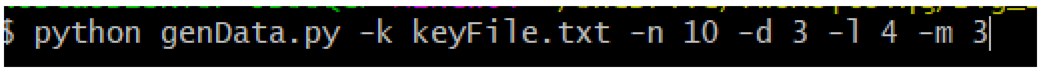
Data creation

For the data creation utility I have created two files. The genData.py and the genDatalib.py where all the functions for the creation of the data reside. In order to execute the genData.py and create data we write the following:

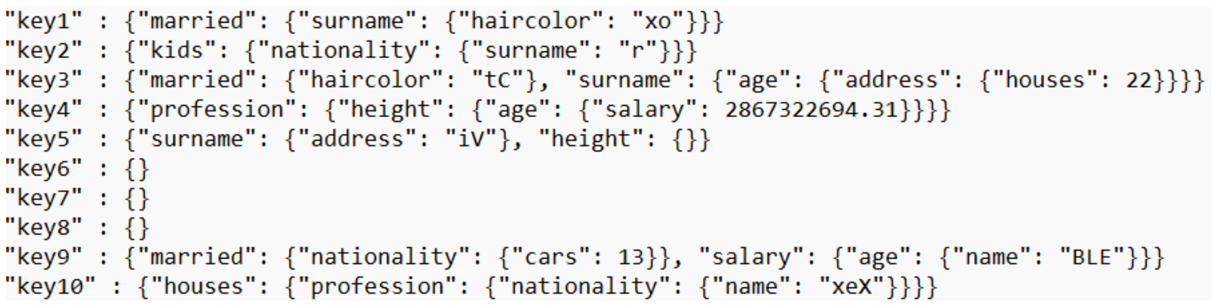
python genData.py [-k K] [-n N] [-d D] [-l L] [-m M]

Where K is the name of the text file with the keys and the type of value each key will store, N the number of lines that we want to create, D the number that indicates the maximum nesting, L a number that indicates the maximum string length and M is a number that indicates the maximum number of nested keys each key can have. In case the user types incorrect input (wrong value or wrong parameter symbol) the program shows a message and exits.

In the genDatalib.py file we create N nested lists which we later transform into dictionaries with the create_list_recurcively function. That function calls the create fake values function to generate fake values according to the type of each key inside the text file (keyFile.txt).

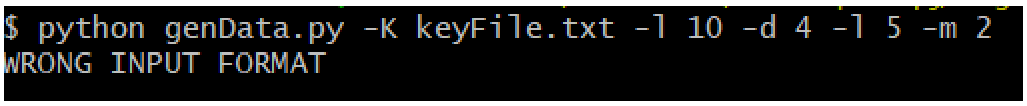


```
$ python genData.py -k keyFile.txt -n 10 -d 3 -l 4 -m 3
```



```
"key1" : {"married": {"surname": {"haircolor": "xo"}}}
"key2" : {"kids": {"nationality": {"surname": "r"}}}
"key3" : {"married": {"haircolor": "tC"}, "surname": {"age": {"address": {"houses": 22}}}}
"key4" : {"profession": {"height": {"age": {"salary": 2867322694.31}}}}
"key5" : {"surname": {"address": "iV"}, "height": {}}
"key6" : {}
"key7" : {}
"key8" : {}
"key9" : {"married": {"nationality": {"cars": 13}}, "salary": {"age": {"name": "BLE"}}}
"key10" : {"houses": {"profession": {"nationality": {"name": "xeX"}}}}
```

In case the user types the input in the wrong format the below message appears.



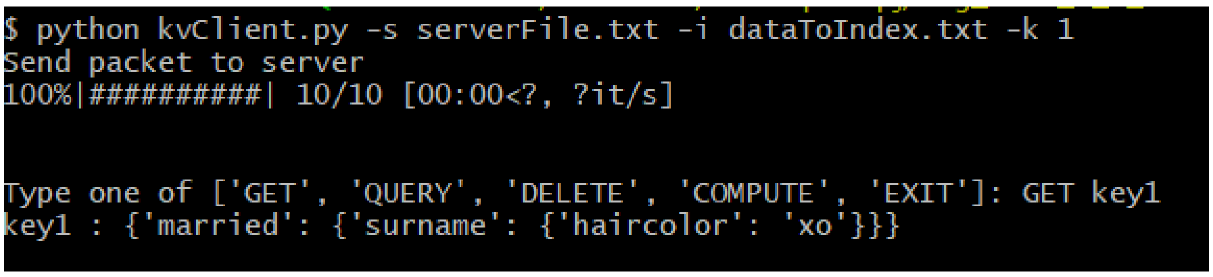
```
$ python genData.py -K keyFile.txt -l 10 -d 4 -l 5 -m 2
WRONG INPUT FORMAT
```

KV-Client

For the client initialization I have created the files kvClient.py and kvClientlib.py where all the functions for the client utilities reside (socket initialization, connection to the servers e.t.c.). To execute the client script we follow the same logic as before. We type our input and if it is not in the correct format a message appears and the program exits. The correct format to run the client program is as follows:

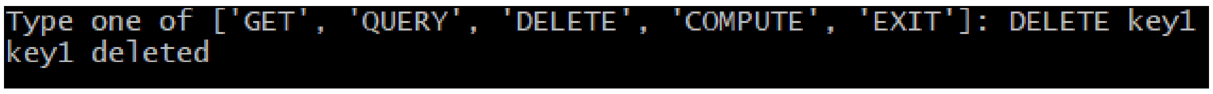
python kvClient.py [-s S] [-i I] [-k K]

Where S is the server text file name, where we have our hosts and the port that each host listens to, I is the file where our data from the previous step are stored and K is the number of servers where each line from the data file will be stored randomly. To store the data we must execute the server program first. After getting the data created in the previous step, the client calls the connect_to_servers function from the kvClientlib.py file to initiate a thread for each host and create a server object with the host and port specified as attributes. After that we connect to each server over the different sockets with the initialize_socket_threads function, we send the data packets to the servers (PUT queries) with the send_packet_to_server function and finally we execute query commands with the execute_query_commands function.
Below I have attached some images from the execution of the client program and the results of the queries I am getting.

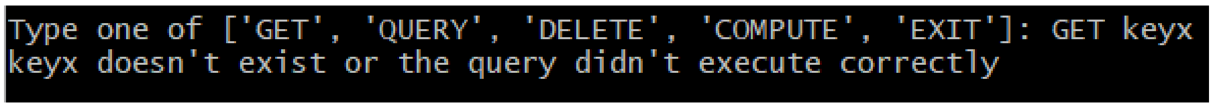


```
$ python kvClient.py -s serverFile.txt -i dataToIndex.txt -k 1
Send packet to server
100%|##########| 10/10 [00:00<?, ?it/s]


Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: GET key1
key1 : {'married': {'surname': {'haircolor': 'xo'}}}
```



```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: DELETE key1
key1 deleted
```

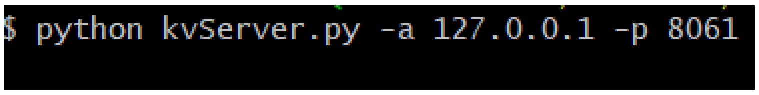


```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: GET keyx
keyx doesn't exist or the query didn't execute correctly
```

KV-Server
For the server's creation I have created three files. The kvServer.py file where we initialize our trie structure, create a server object with the host and the port specified as object attributes and we accept the connection from the client, the kvServerlib.py where all the functions for the execution of the queries are and finally the Server.py where the Server class resides. To execute the kvServer.py script we write the following:

python kvServer.py [-a A] [-p P]

Where A is the host address and P is the port where that host listens to. If we pass the wrong input to the program it exits after showing an exit message. As mentioned above inside the kvServerlib.py file we have all the functions that we need for performing queries. Each function calls the appropriate function from the trie structure file (for searching or deleting data) and returns the results. The queries we can perform are PUT, GET, DELETE, QUERY, COMPUTE. These functions are called from the Server class. Inside the server class we accept the packet (str) with the query the user wants to perform, we check what type of query should be executed with the check_query_type function which also calls the correct function from kvServerlib.py and returns the result of that query. Inside initialize_server method after connection is established we accept queries from the client, we check what type of query should be executed with the check_query_type and the function of that query is called from inside the kvServerlib.py and finally the result of the query is returned to the client. Below I attach an image from the execution of the kvServer.py (it just awaits for the client to connect).



```
$ python kvServer.py -a 127.0.0.1 -p 8061
```

Trie structure

For the storing and retrieval purposes of the data I implement the a Node class with attributes that indicate the letter value of that specific node, the parent of the node, the children nodes, a flag that indicates that we reached the end of the word(leaf), a variable where we will store the whole key-word, a list where we store the children keys of an upper level key, a variable where we store the value of each vertice, a flag variable that indicates if the key is an upper level key and a variable where the nested nodes are stored.

Also in the Trie_Structure.py file the following methods exist:

- add_new_key: For adding a new key in the trie structure. First it searches if part of the key already exists and if yes it adds the rest of the nodes that needed for that key.
- add_nested_node: For iterating recursively over the nested vertices of an upper level key and adding a new nested vertice.
- add_nested_key_value: For inserting a new upper level key with its respective nested keys.
- get_nested_val: For recursively iterating over the trie structure vertices and getting a value of a nested vertice.
- find_nested_key_value: For getting the value of an upper level key. In case other keys are provided also it returns the value of the last node with the get_nested_val function.
- find_key: For finding the key provided and its value. Returns the vertice and a flag that has been found or not .
- delete_key: For deleting an upper level key of trie structure.

Codes

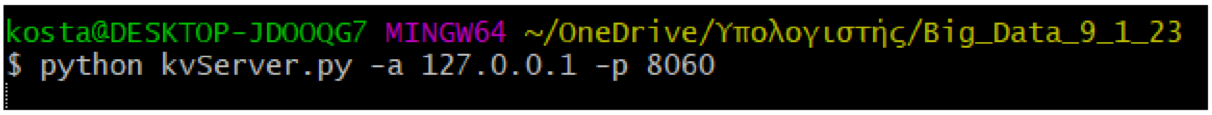
As a good programming practice I created the Codes.py file where I assigned at each value of type string that we might use in our program a variable name (code), so in later versions if we want to change something we will change the value of the that specific variable.

Running the program

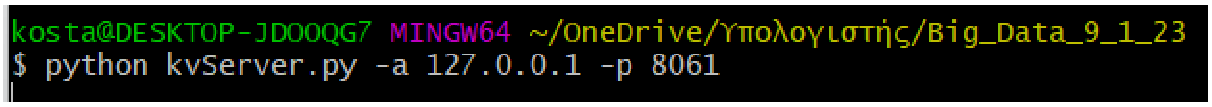
Below I attach some images from the execution of the program for 3000 lines of data. I spawn three servers, as I have three hosts in my serverFile.txt and I store each line of the data file to two servers at random.

serverFile.txt:

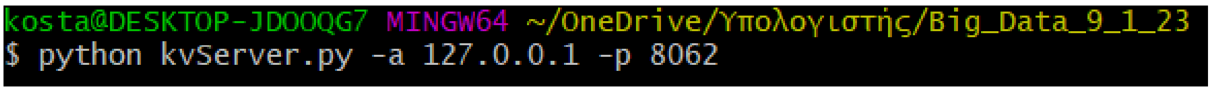
127.0.0.1 8060
127.0.0.1 8061
127.0.0.1 8062



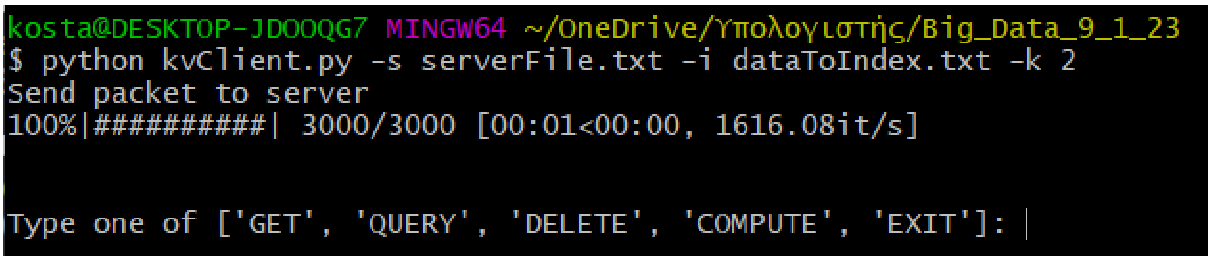
```
kosta@DESKTOP-JDOOQG7 MINGW64 ~/OneDrive/Υπολογιστής/Big_Data_9_1_23
$ python kvServer.py -a 127.0.0.1 -p 8060
```



```
kosta@DESKTOP-JDOOQG7 MINGW64 ~/OneDrive/Υπολογιστής/Big_Data_9_1_23
$ python kvServer.py -a 127.0.0.1 -p 8061
```



```
kosta@DESKTOP-JDOOQG7 MINGW64 ~/OneDrive/Υπολογιστής/Big_Data_9_1_23
$ python kvServer.py -a 127.0.0.1 -p 8062
```

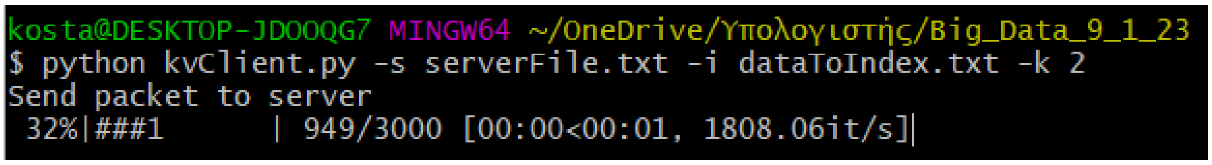


```
kosta@DESKTOP-JDOOQG7 MINGW64 ~/OneDrive/Υπολογιστής/Big_Data_9_1_23
$ python kvClient.py -s serverFile.txt -i dataToIndex.txt -k 2
Send packet to server
100%|##########| 3000/3000 [00:01<00:00, 1616.08it/s]


Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: 
```

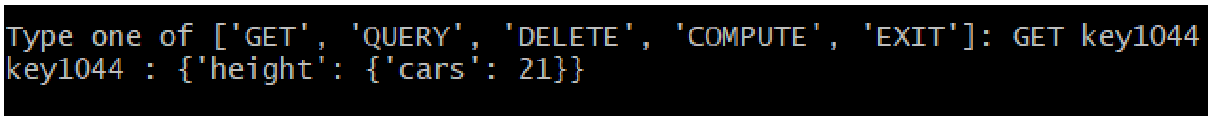
IMPORTANT NOTE

Sometimes when I execute the kvClient the program stuks and the data does not get sent to the servers. Please close the terminals, open new ones, and try again with the execution. Below I attach an image where my program is stuck at 32% of the data loading.



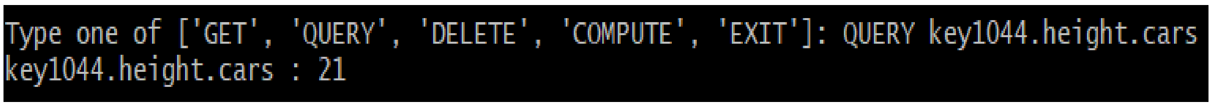
```
kosta@DESKTOP-JDOOQG7 MINGW64 ~/OneDrive/Υπολογιστής/Big_Data_9_1_23
$ python kvClient.py -s serverFile.txt -i dataToIndex.txt -k 2
Send packet to server
 32%|###1      | 949/3000 [00:00<00:01, 1808.06it/s]
```

GET

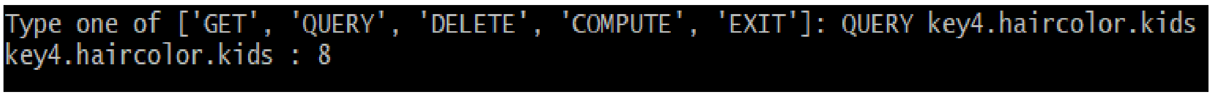


```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: GET key1044
key1044 : {'height': {'cars': 21}}
```

QUERY



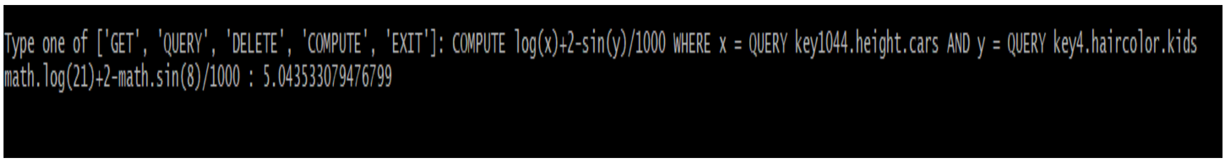
```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: QUERY key1044.height.cars
key1044.height.cars : 21
```



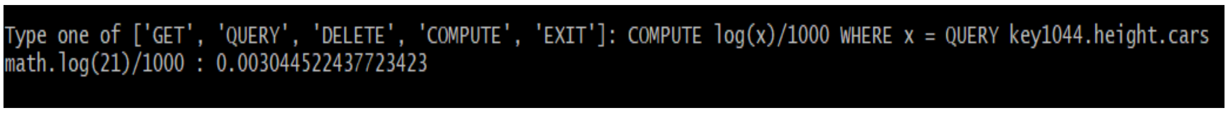
```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: QUERY key4.haircolor.kids
key4.haircolor.kids : 8
```

COMPUTE

We can execute the functions log, sin, tan, cos

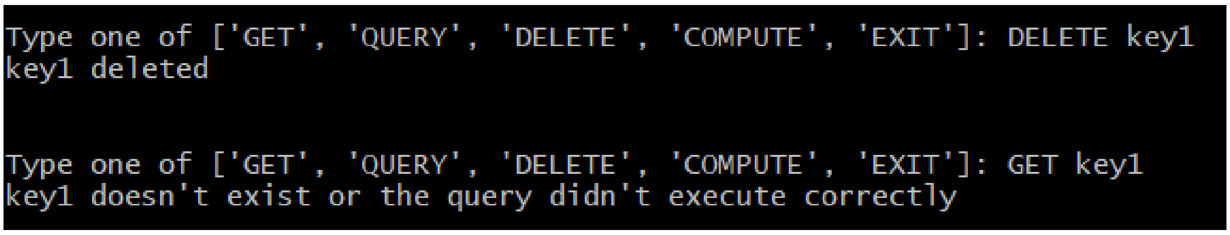


```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: COMPUTE log(x)+2-sin(y)/1000 WHERE x = QUERY key1044.height.cars AND y = QUERY key4.haircolor.kids
math.log(21)+2-math.sin(8)/1000 : 5.043533079476799
```



```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: COMPUTE log(x)/1000 WHERE x = QUERY key1044.height.cars
math.log(21)/1000 : 0.003044522437723423
```

DELETE



```
Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: DELETE key1
key1 deleted


Type one of ['GET', 'QUERY', 'DELETE', 'COMPUTE', 'EXIT']: GET key1
key1 doesn't exist or the query didn't execute correctly
```

Assumptions

- The user does not type anything with brackets single or double
- In the COMPUTE query the user types the mathematical expression without any spaces. Besides that the user types everything else with spaces.