# AL Test - Personal data management API

## Introduction

The API can be used to add set of personal data where each record contains name, address, phone number.
The data can be saved/queried in/from .json or .yaml file.
Along with this, additional data format and query format can be queried and added too.

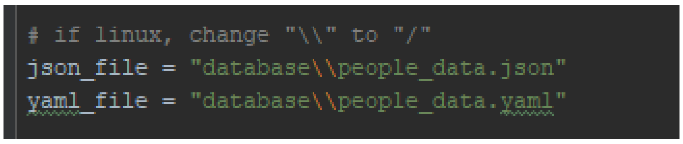
## Required Libraries

Python 3.10 or python 2.7 with optparse, json, yaml

## Data Format and Database Files

Two formats have been used: YAML and JSON
User can use either of these two.
Arg: ---*dataformat json or ---dataformat yaml*
Note: ---*dataformat* is must for save/query. The argument needs to be added in the command line.

**Database files location***: "database/people_data.json"* and *"database/people_data.yaml"*



```
# if linux, change "\\" to "/"
json_file = "database\\people_data.json"
yaml_file = "database\\people_data.yaml"
```

## Command Line interface and Arguments

```
main.py --add --name Harvey_spectre --address Flat302,NY --phonenumber 88451226465 --sex Male --dataformat json
```

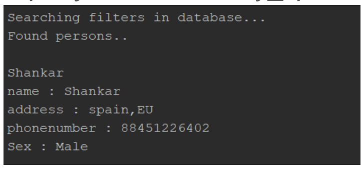
**There are 3 types of categories are specified known as primary args:**

--add:     to add new records
--query:   to query data
--querylist: to query list/set of data

**To Add new records, following secondary args are added next to primary args:**
--name, --address, --phonenumber, --sex
(Additional args/format can be added, the process is explained here)

***Ex:** Main.py –add –name Shankar_Senapati –address Goa,India –phonenumber 986515359 –sex male ---dataformat json*

**To query a person's data(name, address, phonenumber), following args can be used:**
**Ex:**
*--query –name Shankar –dataformat json*
*--query –name Harvey_spectre –dataformat yaml*



```
Searching filters in database...
Found persons..

Shankar
name : Shankar
address : spain,EU
phonenumber : 88451226402
Sex : Male
```

**To query a filtered data, following args can be used:**
**Ex:**
*--query –name Shan –dataformat json*
*--query –sex Male –dataformat yaml*

*--query –phonenumber 88  –dataformat yaml*
**:returns:** any phone number has “88”

```
Shankar
name : Shankar
address : spain,EU
phonenumber : 88451226402
Sex : Male

Harvey_spectre
name : Harvey_spectre
address : Flat302,India
phonenumber : 88451226402
Sex : Male
```

*Or*
**Multiple filters with non-complete syntax can be used. No need to add *India or Shan*:**
If you would like to find all Male persons in India from database.
**Ex:**
*--query –sex Male –Address India –dataformat yaml*
*returns: “Harvey_spectre”: “Goa, India”*

(The code searches both “Male” and “India” in the sex and Address field.)

**To query a predefined list of data, following args are added:**
*--querylist –listallqueryformats*
**:returns:** name, address, phonenumber, sex
*--querylist –listallnames*
**:returns:** Shankar, Harvey_specter, Spane_dalvi
*--querylist –listallphonenumber*
**:returns:** Shankar : 8956226, Harvey_specter : 695438636, Spane_dalvi :87954125
(Additional args can be added, the process is explained here)

## Design and execution

Python Parser has been used for arguments based CLI design.

There are 3 classes.
**Person**: Person object, parse person details from user inputs and export to dict.

```
{'Name': 'Shankar',
'Address': 'Flat302,maithri',
 'Phone number': 884512264}
```

**PersonAssistant**: The Assistant has 3 jobs/attributes to do.
reads the person's database file.
Save data to database.
Also returns all persons names found in database.
**UserListQueries**: Inheriting from **PersonAssistant**
It has 3 attriubutes. It queries list listallqueryformats, listallnames, listallphonenumber. More can be added.

To run program, two main functions are there:

```
args = parse_args_optparse()
run_program()
```

***parse_args_optparse():*** Python Parsers are set here.

***run_program():*** checks types of task like args.adduser or args.query or args.querylist from user inputs. Then redirect the user input data to other functions.

***query_person_data():*** collects/Parse data from user inputs name, address, phone number

## Extend the save/query/filter system for Developer

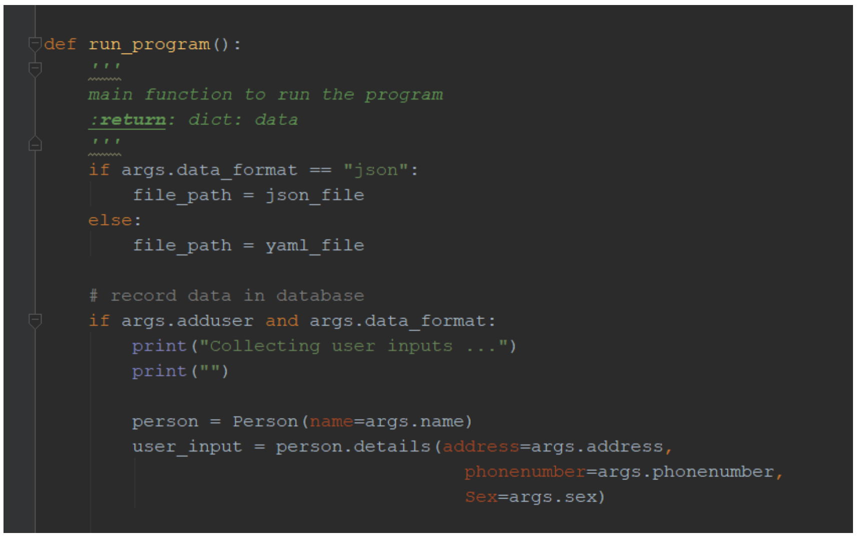
The API is designed in such a way that multiple arguments can be added/extended for querying/adding additional storage formats.

**Let's learn adding the new records storage formats.**
For example, we need to add "age", "language" details.

There are 3 places to add multiple formats.

1. Add new arg to 2 list variables in the code: all_data_store_formats, all_query_format_filters
   ex:
   all_data_store_formats = ["name", "address", "phonenumber", "sex**", "language", "age"**]
   all_query_format_filters = [args.name, args.address, args.phonenumber, args.sex, **arg.language, arg.age**]

2. inside function: run_program() > user_input = person_details(address=args.address, phonenumber=args.phonenumber, Sex=args.sex**, language=args.language**, **age=args.age**)



```
def run_program():
    '''
    main function to run the program
    :return: dict: data
    '''
    if args.data_format == "json":
        file_path = json_file
    else:
        file_path = yaml_file

    # record data in database
    if args.adduser and args.data_format:
        print("Collecting user inputs ...")
        print("")

        person = Person(name=args.name)
        user_input = person.details(address=args.address,
                                    phonenumber=args.phonenumber,
                                    Sex=args.sex)
```

4. Finally, call the arg through parsing arguments
Ex: *main.py --add --name Harvey_spectre --address Flat302,NY --phonenumber 88451226465 --sex* ***Male –language Spanish –age 42*** *--dataformat json*

**Adding additional querylist formats**
Apart from predefined querylist arguments (*--listallqueryformats, --listallnames, --listallphonenumber*), new format can be added.

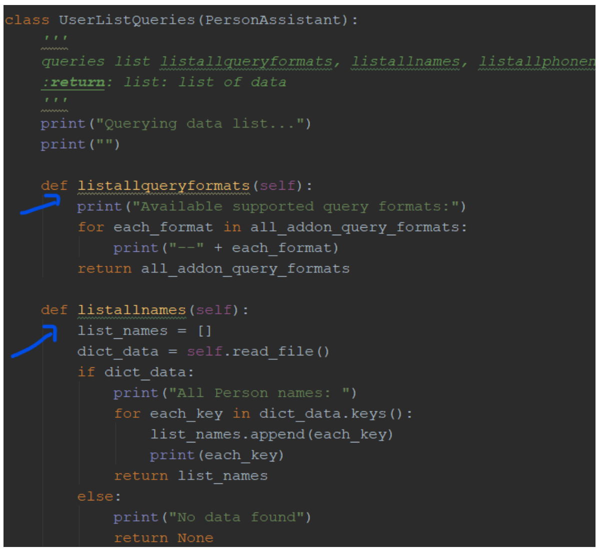
1. Add new arg to list variable: *all_addon_query_formats*

Ex: *all_addon_query_formats = ["listallqueryformats", "listallnames", "listallphonenumber", "**listallpersonage**"]*

2. inside calss: *UserListQueries.*
   *PersonAssistant.read_File()* reads the json/yaml database file and returns a dict of all injected data.

   *Ex: dict_data = {'Shankar': {'**name**': 'Shankar', 'address': 'spain,EU', 'phonenumber': '88451226402', 'Sex': 'Male', '**age**': 42}}*

   Just like all other functions inside ***UserListQueries***, new function can be added.
   listallage()
   listalllanguage()

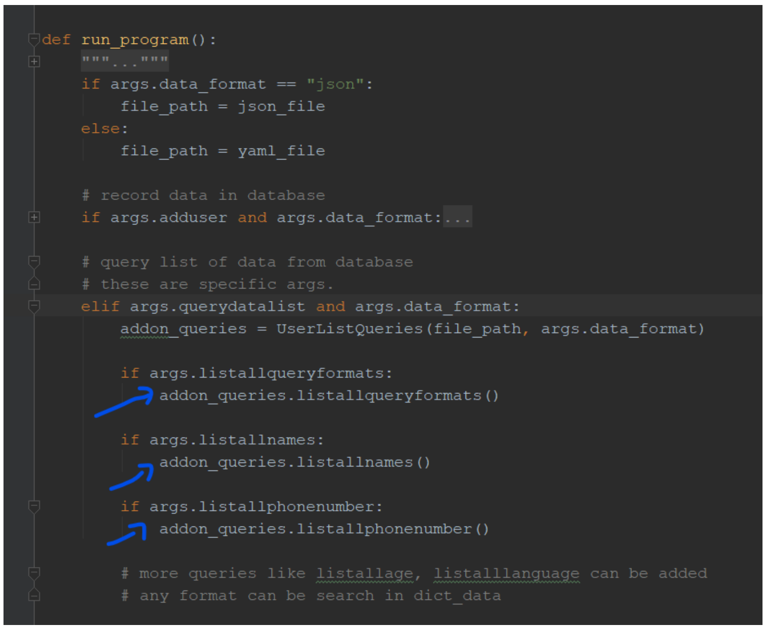


```
class UserListQueries(PersonAssistant):
    '''
    queries list listallqueryformats, listallnames, listallphonen
    :return: list: list of data
    '''
    print("Querying data list...")
    print("")

    def listallqueryformats(self):
        print("Available supported query formats:")
        for each_format in all_addon_query_formats:
            print("--" + each_format)
        return all_addon_query_formats

    def listallnames(self):
        list_names = []
        dict_data = self.read_file()
        if dict_data:
            print("All Person names: ")
            for each_key in dict_data.keys():
                list_names.append(each_key)
                print(each_key)
            return list_names
        else:
            print("No data found")
            return None
```

*And Finally call above function inside **run_program()***



```
def run_program():
    """..."""
    if args.data_format == "json":
        file_path = json_file
    else:
        file_path = yaml_file

    # record data in database
    if args.adduser and args.data_format:...

    # query list of data from database
    # these are specific args.
    elif args.querydatalist and args.data_format:
        addon_queries = UserListQueries(file_path, args.data_format)

        if args.listallqueryformats:
            addon_queries.listallqueryformats()

        if args.listallnames:
            addon_queries.listallnames()

        if args.listallphonenumber:
            addon_queries.listallphonenumber()

        # more queries like listallage, listalllanguage can be added
        # any format can be search in dict_data
```

## Bugs to fix

Currently, it does not take whitespace in string.
Not tested in Mac OS.
Uppercase/Lowercase issues are not fixed.
Sometime querying multiple filtered formats does not output data as expected and single filter works well.