

**CMPE 30052 LAB ACTIVITY NO.
1 – APPLICATION OF PYTHON
LIST**

LIST APPLICATION

Source Code:
<https://github.com/MichaelViernes271/CMPE-30052>

M I C H A E L V I E R N E S

PROGRAM OUTPUT

```
$ python address_book.py
-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 1
First name: lambda
Surname: delta
Address: pasig city
Contact: 12345768
Added Successfully!

-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 1
First name: alpha
Surname: beta
Address: pasay city
Contact: 87654321
Added Successfully!

-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 1
First name: epsilon
Surname: gamma
```

Figure 1. Adding contacts to the address book (a)

PROGRAM OUTPUT

```
=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 1
First name: epsilon
Surname: gamma
Address: caloocan city
Contact: 12341234
Added Successfully!

=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 1
First name: omega
Surname: zeta
Address: pasay city
Contact: 09123412
Added Successfully!

=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 4
No. First Name Last Name Contact Address
*****
0 lambda delta 1234568 pasig city
```

Figure 2. Adding contacts to the address book (b)

PROGRAM OUTPUT

```
-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 4
No. First Name Last Name Contact Address
-----
0 lambda delta 12345768 pasig city
1 alpha beta 87654321 pasay city
2 epsilon gamma 12341234 caloocan city
3 omega zeta 9123412 pasay city

-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 2
Select entry number to change: 0
----- BEFORE CHANGE
0 lambda delta 12345768 pasig city

First name: eta
Surname: theta
Address: caloocan city
Contact: 123445678
-----
AFTER CHANGE
0 eta theta 123445678 caloocan city

Changed Successfully!
```

Figure 2. Adding contacts to the address book (b)

PROGRAM OUTPUT

```
-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 4
No. First Name Last Name Contact Address
-----
0 lambda delta 12345768 pasig city
1 alpha beta 87654321 pasay city
2 epsilon gamma 12341234 caloocan city
3 omega zeta 9123412 pasay city

-----ADDRESS BOOK-----
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 2
Select entry number to change: 0
----- BEFORE CHANGE
0 lambda delta 12345768 pasig city

First name: eta
Surname: theta
Address: caloocan city
Contact: 123445678
-----
AFTER CHANGE
0 eta theta 123445678 caloocan city

Changed Successfully!
```

Figure 3. Viewing the contents of the address book.
Editing the first entry in the address book, and
showing the changes made thereafter.

PROGRAM OUTPUT

```
=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 4
No. First Name Last Name Contact Address
*****
0 eta theta 123445678 caloocan city

1 alpha beta 87654321 pasay city
2 epsilon gamma 12341234 caloocan city
3 omega zeta 9123412 pasay city

=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 5
Options:
1. First Name
2. Last Name
3. Contact
4. Address

Select Number: 2

Search for:zeta
*****RESULT*****
Name: omega zeta
Phone No.: 9123412
Address: pasay city

=====ADDRESS BOOK=====
```

Figure 4. Searching for a keyword in the address book to search for the name zeta.

PROGRAM OUTPUT

```
=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 3
Delete by index: 3
omega zeta 9123412 pasay city
Deleted Successfully.

=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 4
No. First Name Last Name Contact Address
*****
0 eta theta 123445678 caloocan city

1 alpha beta 87654321 pasay city

2 epsilon gamma 12341234 caloocan city

=====ADDRESS BOOK=====
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit

Choose action by number: 6
Closing...

MAYO@VIERNES MINGU64 /c/pup/year2/CMPE 30052/lab (main)
$
```

Figure 5. Deleting the fourth entry (3rd index).
Reviewing the contents of address book before
closing the main application.

SOURCE CODE

author: Viernes, Michael E.

yr/lvl: BSCOE 2-1

objective: create a address book in which it can create, edit, delete, search, view and exit the program

```
# init lists
fname = []
lname = []
address = []
contact = []

def menu():
    print("ADDRESS BOOK".center(80, "="))
    menudisplay = \
    """\
What would you like to do?
1. Add Contact
2. Edit Contact
3. Delete Contact
4. View Contacts
5. Search Address Book
6. Exit
    """
    print(menudisplay)

def create(): # asks for user entries
    try:
        usr_fname = input("First name: ")
        usr_lname = input("Surname: ")
        usr_address = input("Address: ")
        usr_contact = int(input("Contact: "))

        fname.append(usr_fname)
        lname.append(usr_lname)
        address.append(usr_address)
        contact.append(usr_contact)
    except ValueError as e:
        print("Invalid Contact. Try again.\n")
    except Exception as e:
        print("Unexpected entry. Try again.")
    print("Added Successfully!\n")
```

Source Code:

<https://github.com/MichaelViernes271/CMPE-30052>

SOURCE CODE

```
def edit():
    idx = int(input("Select entry number to change: "))

    result = f"""
{idx:00}\t\t{fname[idx]}\t\t{lname[idx]}\t\t{contact[idx]}\t\t{address[idx]}
    """

    print("***80, "BEFORE CHANGE\n",result)

    try:
        usr_fname = input("First name: ")
        usr_lname = input("Surname: ")
        usr_address = input("Address: ")
        usr_contact = int(input("Contact: "))

        fname[idx] = usr_fname
        lname[idx] = usr_lname
        address[idx] = usr_address
        contact[idx] = usr_contact
    except ValueError as e:
        print("Invalid Contact. Try again.\n")
    except Exception as e:
        print("Unexpected entry. Try again.")

    result = f"""
{idx:00}\t\t{fname[idx]}\t\t{lname[idx]}\t\t{contact[idx]}\t\t{address[idx]}
    """

    print("***80, "\nAFTER CHANGE\n",result)
    print("Changed Successfully!\n")

def delete(idx): # delete by index
    if type(idx) is int:
        print(fname[idx], lname[idx], contact[idx], address[idx],
              "\nDeleted Successfully.", "\n"*2)
        del fname[idx], lname[idx], contact[idx], address[idx]
    else:
        print("There is no such entry. Try again.")

def view(): # display in table-like format
    print("No. First Name\tLast Name\tContact \tAddress\n", "***80)
    for idx in range(0, len(fname)):
        result = f"""
{idx:2} {fname[idx]}\t\t{lname[idx]}\t\t{contact[idx]}\t\t{address[idx]}
        """

        print(result)
```

Source Code:

<https://github.com/MichaelViernes271/CMPE-30052>

SOURCE CODE

```
def search():
    print("""\
Options:
1. First Name
2. Last Name
3. Contact
4. Address
""")
    idx = int(input("\nSelect Number: "))
    field = fname if idx == 1\
    else lname if idx== 2\
    else contact if idx == 3\
    else address if idx == 4 else 5

    searchfor = input("\nSearch for:")
    if searchfor in field:
        # change "idx" variable from options selection above into field index
        idx = field.index(searchfor)
        usr_fname = fname[idx]
        usr_lname= lname[idx]
        usr_contact = contact[idx]
        usr_address = address[idx]

        # print formatted results
        print("RESULT".center(80, "*"))
        result = "Name: {0} {1}\n\tPhone No.: {2}\n\tAddress: {3}"
        result = result.format(usr_fname, usr_lname, usr_contact, usr_address)
        print(result, "\n"*2)
    elif idx == 5: return "Invalid Option."
```

Source Code:

<https://github.com/MichaelViernes271/CMPE-30052>

SOURCE CODE

```
def main():

    while True:
        menu() # display actions for address book

        idx = (input("Choose action by number: "))

        if(idx == "1"): # create
            create()
        elif(idx == "2"): # edit
            edit()
        elif(idx == "3"): # delete
            usr_input = int(input("Delete by index: "))
            delete(usr_input)
        elif(idx == "4"): # view
            view()
        elif(idx == "5"): # search
            if not (len(lname) == 0):
                search()
            else:
                print("The book is empty. :(\n")
        elif(idx == "6"): # exit
            print("Closing...")
            exit()
        else:
            print("Select correct index.\n")

if __name__ == "__main__":
    main()
```

Source Code:

<https://github.com/MichaelViernes271/CMPE-30052>

REFLECTION

The list data type in Python offers a multitude of methods and use cases, and frequently solves most of the problems that needs immediate receptacle for variables. In this regard, the benefits outweighs any of the very few demerits it has for specific circumstances but it shall.

To start with, I am relieved to say that my code was made out to be organized and simple like how python tries to approach both its developer and user. Using the list data type, there is not a shadow of doubt wherein its usage has come to hand in terms of flexibility and mixed data in structuring quick solutions. No need for excruciating efforts to check whether type is exactly as the developer encoded it the way it was intended because the most important part is that it gives resolution to the problem.

The methods in the list are the cream of the crop when combined with other python expressions. Just in my case, I was measuring the length of the list frame so that I can use it to iterate the values of lists so it displays the data using the view function; limiting up to where it needs to stop iterating. Moreover, it helped me in implementing the code for edit function by simply assigning the values by position the user requested. Voila! Short solutions make powerful effect.

Its fascinating features, however, also come with great consequences. For an instance, as I was trying to manage the search results it had been cumbersome to delineate which position, the index, of the list (e.g. fname, lname, contact, address) to match the results in searching for the desired keyword of the user. Therefore, as it may looked organized from afar there is a necessity for a ideas by thinking outside of the box. Still, it was just one of the incompatibilities among the many advantages of the list.

Overall, utilizing list to its maximum potential can make big differences anytime a sequence of data has to be contained and manipulated. The list is the answer, for most of the time.