Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 1

Laboratory 1: Getting Acquainted with Python

Submitted by:
Surio, Rovil Jr., M.

Instructor:
Engr. Dylan Josh D. Lopez

September 23, 2020

I. Objectives

This laboratory activity aims to practice the students in implementing the principles and techniques needed in getting started with coding using the Python programming language.

II. Methods

- Implementing methods and functions of Python.
  - Usage of the function and method.
- To learn how to use the specific function or method.
  - Achieved by reading the documentation of each function or method.

III. Results

```
party = ['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra']
levels = [15, 11, 18, 5, 14]

pokemons = "\n".join("{} {} {}".format(party, 'at level', levels)
for party, levels in zip (party, levels)); print(pokemons)
```

```
Charmander at level 15
Pidgey at level 11
Sandshrew at level 18
Rattata at level 5
Abra at level 14
```

Figure 1 Cell 1 Code & Output

Figure 1 shows the code executed to have an identical output to what has been presented in the given problem. It was achieved by using the function zip() to put together the two lists and convert it together into a string by using the join string method. Also, the word 'at level' was added in-between of the Pokémon names and level in the positional argument from the format string method so that when it prints, it will follow the format that was assigned which is the ("{} {} {}".format(party, ‘at level’, levels). So, the first index from the 'party', will go to the first ‘{}’ to be followed by the word ‘at level’ into the second ‘{}’ then lastly the first index from the ‘level’ list into last ‘{}’. This process is in a loop until it reached the last index of the list or ‘Abra’ and level 14 in the ‘levels’ list.

```
pick = []
reserves = [
    ('Onix',10),
    ('Slowpoke',18),
    ('Dialga', 2),
    ('Magikarp', 32),
    ('Feebas', 22),
    ('Swablu', 19),
    ('Regigigas', 3),
    ('Unown', 50)
]
reserves.sort(reverse=True,key=lambda x: x[1])
pick.append(reserves[0][0])
pick.append(reserves[1][0])
pick.append(reserves[2][0])
print(pick)
```

```
['Unown', 'Magikarp', 'Feebas']
```

Figure 2 Cell 2 Code & Output

Figure 2 shows the block of code created to have an identical output to what is presented in the problem. To begin, a variable ‘pick’ was defined as a list as shown in the first line of code in Figure 2 and the ‘reserves’ list was sorted out into descending order by using the reverse keyword as one of a parameter of the sort method. It was set to True because the default value of reverse is False. Also, a lambda expression or anonymous function was used as the second parameter of the sort method to set the location of the index that will be accessed from the list of tuple/Pokémon with levels that contain 0 and 1 index. The sorting of the list of tuple/pokemon with levels by their 2nd index was possible because of the indication in the lambda expression to access the second index which is in code is “key=lambda x: x[1]”. Therefore, If the lambda expression was not used, the sort method will sort the list alphabetically based on the 0 index which is the names of Pokémon. Lastly, after sorting it, the first 3 elements were appended to the variable ‘pick’ empty list that been defined, and by printing it, the output shows the 3 Pokémon parties desired.

```
def create_party(party,candidates):
    suggested_parties = None
    party.append(pick[0])
    print(party) ; party.remove("Unown")
    party.append(pick[1])
    print(party) ; party.remove("Magikarp")
    party.append(pick[2])
    print(party)

    return suggested_parties

create_party(party,pick)
```

```
['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Unown']
['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Magikarp']
['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Feebas']
```

Figure 3 Cell 3 Code & Output

As shown in Figure 3, a block of code was created and executed to present the three possible parties. Using the create_party (party, candidates) function given, a block of code was created that displaying the possible parties while keeping the original members of the 'party' list when the function was called. It was achieved by appending one by one of the selected Pokémon from the 'pick' variable from cell 2. The process includes accessing their index and appending the first pokemon from the 'pick' list which is 'Unown' to the 'party' list using the code “party.append(pick[0])” and printing the current elements of the 'party' list. Next is removing 'Unown' by the code “party.remove(“Unown”)” and replace it with 'Magikarp' by the code “party.append(pick[1])” and print again the current elements of the 'party' list. Lastly, replacing 'Magikarp' by 'Feebas' using these codes “party.remove(“Magikarp”)” and “party.append(pick[2])”. Now that the three possible parties are completed, calling the function to execute the block code shows the identical output to what is presented in the problem.

IV. Conclusion

In my perspective, the difference between Python programming language and C++ language is mainly from syntax and structure. For example in C++, we need curly brackets and semicolons while in python, it only depends on the indention of codes [1]. Also, we need to declare the data type of variable in C++ while in python, we don't do such thing because this is smart enough to identify that's why it was known for flexibility too.

For the graded cell 1, two functions are used and these are the join string and zip. The zip() was defined as according to its documentation [2], "This function returns a list of tuples, where the *i*-th tuple contains the *i*-th element from each of the argument sequences or iterables." The main usage of the zip is to join together 2 or more lists and return it at a list of tuples based on the indexes of each list. Next is the join string method. This is used to join elements from list, tuple, string, dictionary, set, and return it one at a time as described by [3].

The sort method and lambda expression were used in graded cell 2. According to [4], python lists have a built-in list.sort() method that modifies the list in-place and this is the main usage of it. Also, the lambda expression is used to create an anonymous function [5]. With this, pinpointing a specific index is possible in a list when sorting it. Lastly, the usage of the append method is to add an item or element to the end of the list [6].

The advantage of using python is that it has much documentation and its community is big so there are a lot of tutorials. Also, comparing to other languages, Python programming language is more flexible in terms of syntax. Likewise, in Jupyter Notebook, it is extremely useful in data science because of its capability to produce such as graphs and vectors which is a good representation of many things. The only problem I have is I don't know if there is another way to open a project but I find it a hassle to run a project because I need to open the Anaconda Prompt and locate the drive, locate the file and open the Jupyter Lab. It is very time-consuming in my perspective.

References

[1]"2.1.7 Indentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/2.0/ref/indentation.html. [Accessed: 23- Sep- 2020].

[2]"2. Built-in Functions — Python 2.7.18 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/2/library/functions.html#zip. [Accessed: 12- Sep- 2020].

[3]"join() function in Python - GeeksforGeeks", *GeeksforGeeks*, 2020. [Online]. Available: https://www.geeksforgeeks.org/join-function-python/. [Accessed: 12- Sep- 2020].

[4]"Sorting HOW TO — Python 3.8.6rc1 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3/howto/sorting.html. [Accessed: 12- Sep- 2020].

[5]"6. Expressions — Python 3.8.6rc1 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3/reference/expressions.html. [Accessed: 12- Sep- 2020].

[6]"5. Data Structures — Python 3.8.6rc1 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3/tutorial/datastructures.html. [Accessed: 12- Sep- 2020].

Appendix

The link for the GitHub Repository — https://github.com/RovilSurioJr/-Laboratory-1