Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 1

# Getting acquainted with Python

*Submitted by:*

Parco, Jovian Asher G

*Instructor:*

Engr. Dylan Josh D. Lopez

September 29, 2020

## I. Objectives

This laboratory activity aims to implement the principles and techniques of essential Python functions and list manipulation.

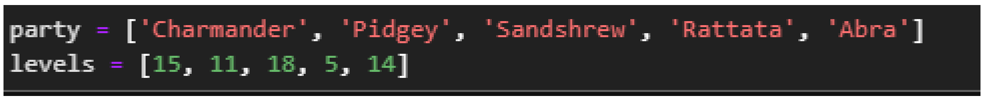
## II. Methods

- In this laboratory activity, the practices consist of four python core functions such as zip() function, sorted() function, pop() function, and append() function.
    - A zip() function returns a zip object, which is an iterator of tuples where each item in each passed iterator is paired together [1].
    - A sorted() function return a new sorted list from an iterable [2], with two arguments consisting of key argument and reverse argument. The key specifies a function of one argument that is used to extract a comparison key from each list element [2], while the reverse argument is used to determine the element order.
    - A pop() function removes the item at the given position in the list and returns it [3]. The argument specifies the index position.
    - An append() function adds an item to the end of the list [3].

- The deliverables of this activities are printing two lists in one line of code, sorting a list while picking the highest value, and creating a function that will combine two lists that will output in a certain way.
    - Printing two lists in one line of code was achieved using a "for loop" statement and the zip() function.
    - Sorting a list and picking the highest value was achieved using the sorted() function with a specific argument to attain the desired output.
    - Finally, creating a function that will combine two lists in a certain way was achieved using a while loop and the combination of pop() function and append() function.

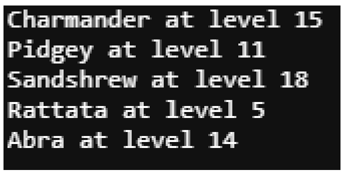
## III. Results

The complete repository for the activity are available at Github(https://bit.ly/2HDIrW6) Part 1 of the lab activity is a straight forward problem, where you need to output (Figure 2) using two sets of the list (Figure 1) while using one line of code to get a higher score.



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

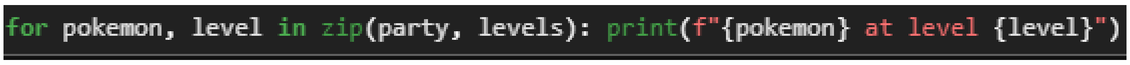
Figure 1 Set of lists given part 1



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

Figure 2 Problem output in part 1

To tackle this problem, one must know how to browse python documentation and understand the searched function correctly; With this method zip() function and the for loop function were used to create the desired output (Figure 2).

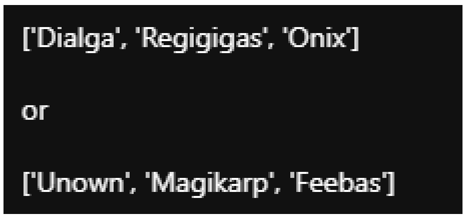


```
for pokemon, level in zip(party, levels): print(f"{pokemon} at level {level}")
```

Figure 3 Solution to the problem in part 1

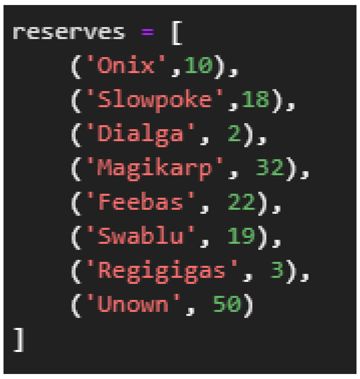
Combining zip() function and for loop (Figure 3), the researcher can traverse the two list (Figure 1), meaning they keep their elements in the same order in which they were introduced[4] and output it the same way.

Part 2 of the lab activity consists of sorting a list of Pokemon(Figure 5) while adding one more Pokemon to complete a team and creating a new list "picks" with the top 3 highest or lowest Pokemon level (Figure 4).



```
['Dialga', 'Regigigas', 'Onix']

or

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

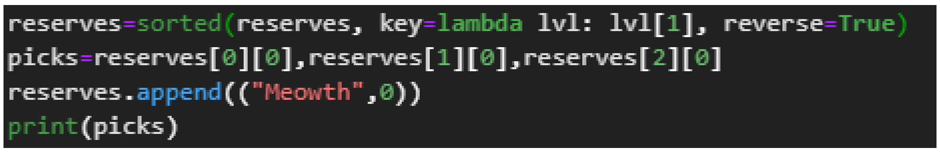
Figure 4 Desired output in part 2



```
reserves = [
    ('Onix',10),
    ('Slowpoke',18),
    ('Dialga', 2),
    ('Magikarp', 32),
    ('Feebas', 22),
    ('Swablu', 19),
    ('Regigigas', 3),
    ('Unown', 50)
]
```

Figure 5 Pokemon list in part 2

Using the methods tackled in part 1, the researcher finds the sorted() function and append() function will be a useful function to face the problem; the sorted() function is used to return a sorted list with two optional arguments such as key argument and reverse argument, while the append() function is used to add a new element on the list.



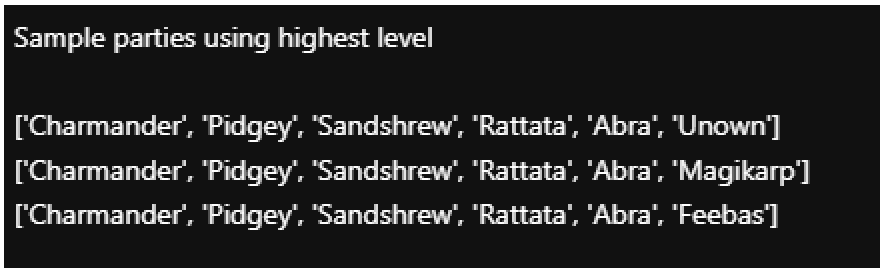
```
reserves=sorted(reserves, key=lambda lvl: lvl[1], reverse=True)
picks=reserves[0][0],reserves[1][0],reserves[2][0]
reserves.append(("Meowth",0))
print(picks)
```

Figure 6 Solution to the problem in part 2

Using the sorted() function and append() function (Figure 6), the researcher was able to solve the problem quickly; the solution is in the arguments of the sorted() function, the key argument was used with the lambda function to sort the list by Pokemons level or the first index, and the reverse arguments specify the order of the list.

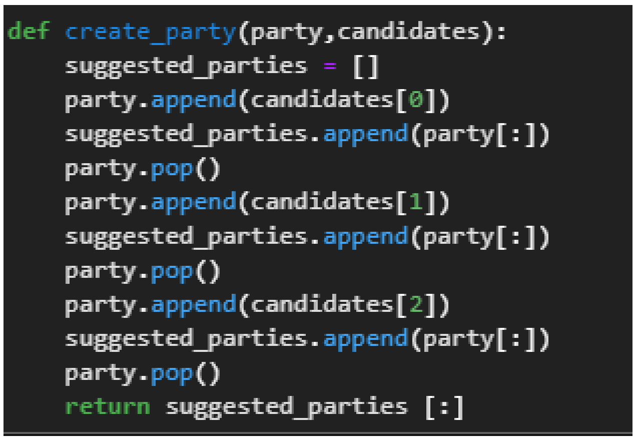
The variable "picks" contain the only top 3 pokemon in the sorted list of reserves, the append() function is used to add new Pokemon and their level to the reserves list.

Part 3 of the lab activity is the most complicated between the rest because it involves a function creation and using the variable of part 1(Figure 1) & 2(Figure 4) to output all the lists in a certain way (Figure 7).



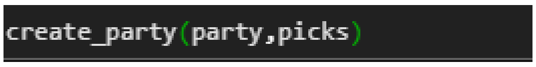
```
Sample parties using highest level

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

Figure 7 Desired output in part 3



```python
def create_party(party,candidates):
    suggested_parties = []
    party.append(candidates[0])
    suggested_parties.append(party[:])
    party.pop()
    party.append(candidates[1])
    suggested_parties.append(party[:])
    party.pop()
    party.append(candidates[2])
    suggested_parties.append(party[:])
    party.pop()
    return suggested_parties [:]
```

Figure 8 Solution to the problem in part 3

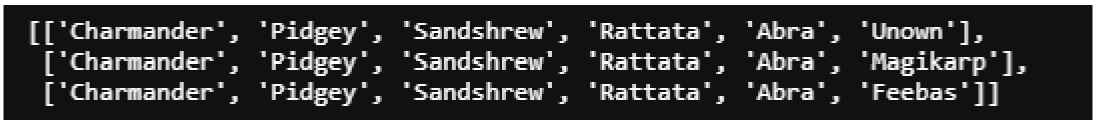


```python
create_party(party,picks)
```

Figure 9 Input for part 3

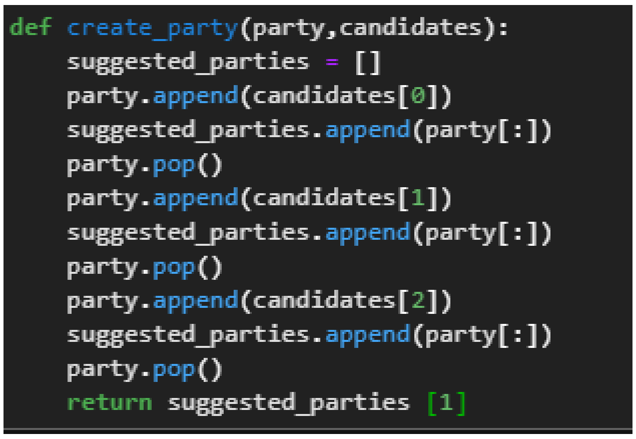
There are many ways to solve the problem, but the researchers want to create a useful function that a user can choose what Pokemon party will be appropriate; The first line in (Figure 8) is the creation of a function with the parameters "party" & "candidates" the second line creates a blank list called "suggested_parties".

The third line uses the append() function to add the first element in the candidate list to the party list; the fourth line uses the append() function again to add the tuple to the list in the "suggested_party" list, creating a list within the list, the next line use the pop() function to remove the last element in the "party" list essential resting it, the third to the fifth line loops two times because of time constrain the researcher cannot create a looping statement that is functional, the final line returns the suggested parties and output (Figure 9) as a list, you can further utilize the researchers work by specifying what output you want by changing the return argument index (figure 10) to get the sample result of (figure 11).



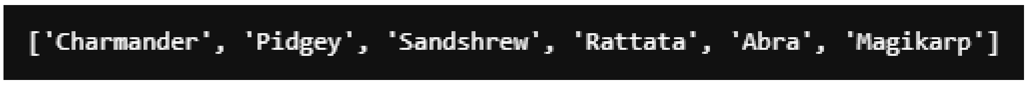
```
[['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Unown'],
 ['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Magikarp'],
 ['Charmander', 'Pidgey', 'Sandshrew', 'Rattata', 'Abra', 'Feebas']]
```

Figure 9 Output for part3



```
def create_party(party,candidates):
    suggested_parties = []
    party.append(candidates[0])
    suggested_parties.append(party[:])
    party.pop()
    party.append(candidates[1])
    suggested_parties.append(party[:])
    party.pop()
    party.append(candidates[2])
    suggested_parties.append(party[:])
    party.pop()
    return suggested_parties [1]
```

Figure 10 Changing retun() function to the 1 index



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

Figure 11 Custom output of the 1index

## IV. Conclusion

For the researcher, the lab activity is a success because he does not have any background in the language in Python, only C++ and C# which is far more complicated than Python because Python language knows what type of data is used without specifying it, and simple function uses less word than C# and C++,

The function used for the lab activity utilized the Python documentation website [5], which is easier than it looks by using the built-in find function in a browser, you can find the function easier. The first function that has been used is a zip() function returns a zip object, which is an iterator of tuples where each item in each passed iterator is paired together [1].
The second function a sorted() function that return a new sorted list from an iterable [2], with two arguments consisting of key argument and reverse argument. The key specifies a function of one argument that is used to extract a comparison key from each list element [2]. In contrast, the reverse argument is used to determine the element order.
The third function is a pop() function that removes the item at the given position in the list and returns it [3]. The argument specifies the index position.
The fourth and final function that has been used is a append() function that adds an item to the end of the list [3].

For the researcher, the only disadvantage of Python as a language is learning to read and analyze syntax in Python for the first time. The rest is easy because of the availability of documentation. Jupyter Notebook, at first glance it looks a complicated compiler because of the cells were in the past complier that the research use only have one run button, unlike Jupyter Notebook where it can run the individual cell, which is useful for this activity; the only problem is that it needs to restart the kernel every time there is a change in a variable but can be remedy with a single tap on the toolbar, and another advantage of Jupyter Notebook is having in build documentation by using a shortcut key that helps to save time.

# References

[1]"Python zip() Function", W3schools.com, 2020. [Online]. Available: https://www.w3schools.com/python/ref_func_zip.asp. [Accessed: 28- Sep- 2020].

[2]"2. Built-in Functions — Python 3.3.7 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3.3/library/functions.html#sorted. [Accessed: 28- Sep- 2020].

[3]"5. Data Structures — Python 3.3.7 documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3.3/tutorial/datastructures.html. [Accessed: 28- Sep- 2020].

[4]R. Python, "Using the Python zip() Function for Parallel Iteration – Real Python", *Realpython.com*, 2020. [Online]. Available: https://realpython.com/python-zip-function/. [Accessed: 28- Sep- 2020].

[5]"3.8.6 Documentation", *Docs.python.org*, 2020. [Online]. Available: https://docs.python.org/3/. [Accessed: 28- Sep- 2020].