e-ISBN : 978-967-25608-8-3
SIG : e-Learning@CS
https://appspenang.uitm.edu.my/sigcs/
Publication Date : 04-Apr-2023

DATA STRUCTURE UNIQUENESS IN PYTHON PROGRAMMING LANGUAGE

*Jamal Othman
* *jamalothman@uitm.edu.my*[1]

[1]Jabatan Sains Komputer & Matematik (JSKM),
Kolej Pengajian Pengkomputeran, Informatik dan Media,
Universiti Teknologi MARA Cawangan Pulau Pinang, Malaysia

**Corresponding Author*

ABSTRACT

Python is categorized as general-purpose language which means it can be applied to a variety of problems or not specialized to a specific area. Python is a powerful programming language which supports the creation of application systems, penetration of security testing and conduct complex data analysis on huge volume of data (data science). Hence, Python helps the programmer to develop the application faster because the syntaxes are almost like and simpler than C, C++ and Java programming languages. Among of the uniqueness of Python programming language is the implementation of its data structure. Data structures as implemented in the Python programming are easier to be organized or manipulated. Set, List, Tuple and Dictionary are among the most important built-in data structure in Python. These data structures have been simplified as compared to array and linked list data structures as used in C or C++. Each data structure in Python has its own strengths and characteristics in terms of mutable or immutable which means the ability to modify or change the value of data structure element's. Versatility of Python, make the Pythonistas are highly demanded amongst IT companies if they know better the usage of different kind of data structures in manipulating bulks of huge data to generate very valuable results.

Keywords: python, list, set, tuple, dictionary

Introduction & Literature Review

Python is a programming language which created by Guido van Rossum. It has been released in 1991 (w3School 2023). Generally, Python is used for web & software development, complex mathematical computations, system scripting, security system monitoring and data analysis. Python programming is popular because the strength of the libraries and simplicity of the commands as compared to Java, C++ or C language (Amos et all 2020). Another strength about Python is the data structures which can store and process high volume of data. Instead of manually transferring data from the Web or from spreadsheets, you can use Python to scrape bulks of data sources or spreadsheets in the time it takes you to do just once automatically.

Python is categorized as dynamic programming language because it supports varieties of programming paradigms such Imperative, Object-Oriented and Functional paradigms (Linner et all 2021). Similar to Java Programming, Python is executed in the Python interpreter to generate the Python byte code. The syntax of Python programming is very easy to learn because it's mimicking or like a

pseudocode. Hence, it helps the developer to construct the codes faster and minimize erroneous in the application program. The Python libraries are comprehensive because it also wrapped to C or C++ libraries to perform varieties of tasks and one of the major tasks is data analysis processing (data science). According to trends of peoples asked questions about programming problems through *StackOverflow* forum, figure 1 shows that questions on Python problems relatively exponentially increase through the years starting from 2009 as compared to other programming languages (Amos et all 2020).

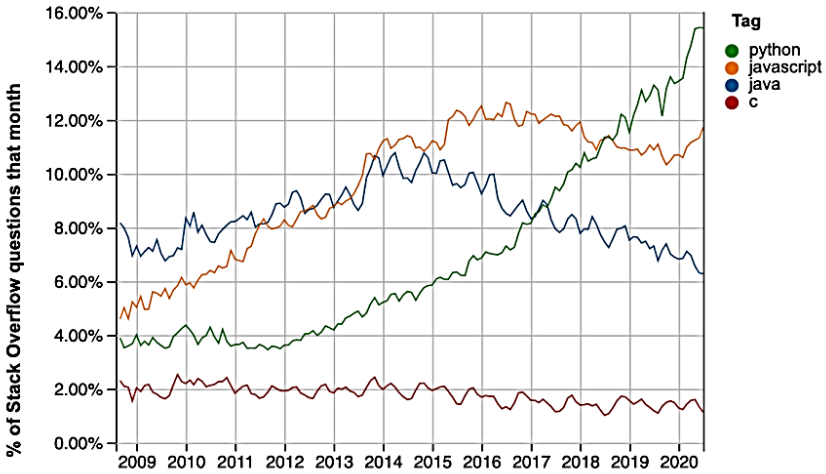

Figure 1: Frequencies of programming questions being asked at *StackOverflow* according to programming language types (Python, JavaScript, Java, C)

Similar survey has been conducted *by StackOverflow* to almost 1000 respondents asking on the popularity or preferability of the programming language as show in the figure 2. Its shows that the Python is topping in the chart.

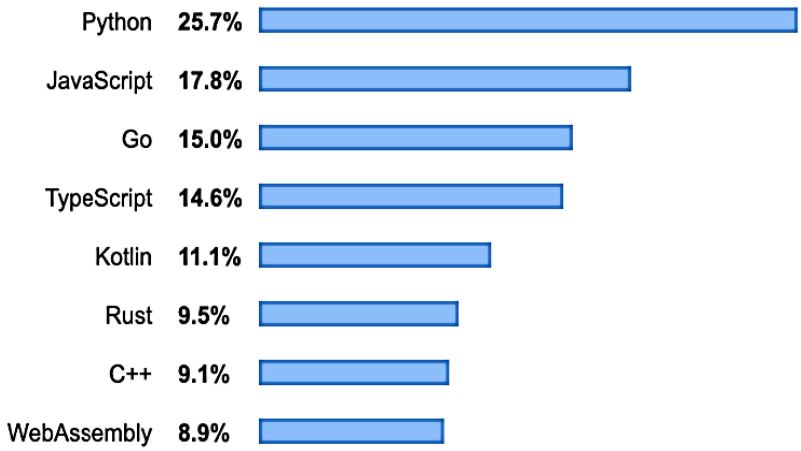

Figure 2: Survey on the popularity of programming languages amongst 1000 respondents

Special characteristic in Python is no variable declaration. This is similar to PHP, Scheme and PROLOG which all this programming languages are categorized as Dynamic typing. Type of variable will be determined during the execution (run time) of the codes. Unlike to C, C++ and Java, all variables must be declared before it can be used in the codes (Othman et all 2019). Python is unique because it's function able to return multiple variables. In contrast to return function in C, C++ or Java, the function can return single variable only, unless the programmer uses the void function to return multiples values by using the reference parameters passing as in C++, object parameter passing as in Java or pointer or array parameters passing as in C programming language (Zelle 2009).

Python has four (4) main non primitive data structures namely Lists, Tuple, Dictionary and Set. These data structures are easier to apply, managed, manipulated and stored as compared to Queue, Stack, Tree or Graph as implemented in C, C++ or Java programming language. Data Structures allows you to organize your data in such a way that enables you to store collections of data, relate them and perform operations on them accordingly. Lists, Tuple, Dictionary and Set are classified is linear data structures because elements of data structure are stored sequentially on the memory. Since the data are stored sequentially, it can be accessed or traversed through in a single execution (R20A0503, 2022). This article will elaborate more on the implementation of data structure such as List, Tuple, Dictionary and Set in Python programming language in the following section.

Comparison of Data Structure

Data structures allows you to manage, organize and store the data in such a way that enable you to perform certain operations or data manipulation (Edureka 2022). Generally, the data structures in Python can be divided into two (2) categories such as built-in and user-defined data structures. The built-in data structures in Python are list, tuple, dictionary and set. While the user-defined data structures are the queue, stack, tree, graph, linked lists and HashMaps. This article will focus discussions on the built-in data structures only.

i) Lists

Lists are the common data structures which applied by the most of python programmers to store either temporary or static data. Lists are similar with the implementation of array in C or C++ programming languages. But one of the special features of the list data structure in Python is, it enables us to store different data types in sequential order. The index value of the list starts with index 0 travelling from the first (front) element until the last (back) element of the list. But if we start from the last element of the list, it will be started with index -1 and the index value decremented travelling the list until the first (front) element of the list. The following are example of lists implementation in Python programming language. The symbol for list is '[]'.

Example 1.1
Example of lists which stores the same data types.

```
#nameList stores the string data type
>>> nameList =["ALI","ABU","IBRAHIM","BAKAR","ZAMRI","JUSOH","AMINAH"]
>>> nameList
['ALI', 'ABU', 'IBRAHIM', 'BAKAR', 'ZAMRI', 'JUSOH', 'AMINAH']

#marksList stores the integer data type
>>> marksList = [67,82,61,70,68,80,77]
>>> marksList
[67, 82, 61, 70, 68, 80, 77]

#GPAList stores the float data type
>>> GPAList = [3.00,4.00,2.67,3.33,3.00,4.00,3.67]
>>> GPAList
[3.0, 4.0, 2.67, 3.33, 3.0, 4.0, 3.67]
```

Example 1.2
Example of methods to retrieve elements from the list.

```
#Original list
>>> listName =["ALI","ABU","IBRAHIM","BAKAR","ZAMRI","JUSOH","AMINAH"]

#to retrieve element from the list at index 1
>>> listName[1]
'ABU'

#to retrieve elements from the list at index 0 until 1, index 2 is excluded
>>> listName[0:2]
['ALI', 'ABU']

#to retrieve the first three elements from the list
>>> listName[:3]
['ALI', 'ABU', 'IBRAHIM']

#to retrieve the last element from the list
>>> listName[-1]
'AMINAH'

#to retrieve the last three elements from the list
>>> listName[-3:]
['ZAMRI', 'JUSOH', 'AMINAH']
```

Example 1.3
Example of list which stores different data types in the same list.

```
#the studentList stores different data types. Starts with string and followed by integer and float.
>>> studentList = ["ALI",67,3.00]
>>> studentList
['ALI', 67, 3.0]
```

Example 1.4
Examples of commands or instructions in Python programming to manipulate elements in the list.

```
#to append new element in the list, the new element will be appended at the last index
>>> nameList.append("KALISHAH")
>>> nameList
['ALI', 'ABU', 'IBRAHIM', 'BAKAR', 'ZAMRI', 'JUSOH', 'AMINAH', 'KALISHAH']
#to remove the value from the list
>>> nameList.remove("BAKAR")
>>> nameList
['ALI', 'ABU', 'IBRAHIM', 'ZAMRI', 'JUSOH', 'AMINAH', 'KALISHAH']
```

Example 1.5
Example of Python code to process the data stored in the list.

```
def main():
            marksList = [67,82,61,70,68,80,77]
            sumMarks = 0

            for x in range(len(marksList)):
                sumMarks = sumMarks + marksList[x]

            average = sumMarks/len(marksList)

            print("The average mark is {0:0.2f}".format(average))
main()
```

Output:

```
The average mark is 72.14
```

ii) Tuples

Tuples are similar as lists but the only different is the elements of the tuple cannot be changed or updated. The term used is immutable. List is mutable. Tuple allows you stores different data types or you may store the same data type in the tuple. Parenthesis '()' is used to create the tuple.

Example 2.1
Example of assigning the values in a tuple.

```
>>> numberTuple = (1,2,3,4,5,6,7,8,9,10)
>>> numberTuple
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Example 2.2
To retrieve the elements from the tuple.

```
#to retrieve element from the tuple at index 2
>>> nameTuple[2]
3

#to retrieve elements from the tuple at index 3 until 4, index 5 is excluded
>>> nameTuple[2:5]
(3, 4)

#to retrieve the last four elements from the tuple
>>> print(nameTuple[-4:])
(1, 2, 3, 4)
```

Example 2.3
To add new elements in the tuple.

```
#Two numbers added in the tuple
>>> numberTuple = numberTuple + (11,12)
>>> numberTuple
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
```

Example 2.4
Implementation of code Python using tuple.

```
def main():

            numberTuple = (1,2,3,4,5,6,7,8,9,10)
            cntOdd=0
            cntEven=0

            for x in range(0,10,1):
                if (numberTuple[x] % 2 == 0):
                    cntEven=cntEven+1
                else:
                    cntOdd=cntOdd+1

            print("Number of odd numbers found: ",cntOdd)
            print("Number of even numbers found: ",cntEven)
main()
```

Output:

```
Number of odd numbers found: 5
Number of even numbers found:  5
```

iii) Dictionaries

Dictionaries data structure in Python consists of two components which are the keys and values. The values in dictionaries are mutable, which the value can be updated. The symbol used for dictionary is the curly bracket '{}' and colon ':' to separate the key on the left and values on the right of the dictionary elements.

Example 3.1
Example of assigning the values in a dictionary.

```
#the elements of dictionary consists of key and value
>>> dicStudent = {1001:"ALI",1002:"ABU",1003:"BAKAR",1004:"MAN",1005:"BAKRI"}
>>> dicStudent
{1001: 'ALI', 1002: 'ABU', 1003: 'BAKAR', 1004: 'MAN', 1005: 'BAKRI'}
```

Example 3.2
To access or retrieve value from the dictionary by using the key of dictionary.

```
#access the value by using the key number but not the index number as
 implemented in list or tuple
>>> print(dicStudent[1003])
```

```
BAKAR


>>> for x in dicStudent:
      print(dicStudent[x], end=" ")
```

Output:

```
ALI ABU BAKAR OSMAN BAKRI
```

Example 3.3

To manipulate (update, insert or remove) values from the dictionary.

```
#to update the value of 'BAKAR' to 'ZAMRI' by using the key '1003'
>>> dicStudent[1003] = "ZAMRI"
>>> dicStudent
{1001: 'ALI', 1002: 'ABU', 1003: 'ZAMRI', 1004: 'OSMAN', 1005: 'BAKAR'}

#to remove the value of 'ZAMRI' from dictionary by using the key '1003'
>>> del dicStudent[1003]
>>> dicStudent
{1001: 'ALI', 1002: 'ABU', 1004: 'OSMAN', 1005: 'BAKAR'}

#to insert or add new element in the dictionary by using the new key
>>> dicStudent[1003]="JUSOH"
>>> dicStudent
{1001: 'ALI', 1002: 'ABU', 1004: 'OSMAN', 1005: 'BAKAR', 1003: 'JUSOH'}
```

iv) *Sets*

Set is a collection of unordered data that is unique or other word the data is not duplicated each other. In Python, all data in a set will be enclosed or inside of the symbol '{}' or curly bracket. Set does not support the indexing to retrieve the elements or data from the set data structure.

Example 4.1

Example of assigning the values in a set. A set allows to store the same or different data types and this is similar to list or tuple.

```
#assigning numbers to setA and setB
>>> setA = {2,4,6,8}
>>> setB = {1,3,5,7}
>>> setA
{8, 2, 4, 6}
>>> setB
{1, 3, 5, 7}

#assigning numbers and characters in the setC
>>> setC = {1,'a',2,'b',3,'c',4,'d'}
>>> setC
{1, 2, 3, 4, 'a', 'd', 'b', 'c'}
```

Example 4.2

To manipulate (update, insert or remove) values from the set.

```
#to insert or add new value in the setA
>>> setA.add(10)
```

```
>>> setA
{2, 4, 6, 8, 10}

#to insert values in the setA, the inserted new values will be organized
in ascending order
>>> setA
{2, 4, 6, 8, 10}
>>> setB
{1, 3, 5, 7}
>>> setA.update(setB)
>>> setA
{1, 2, 3, 4, 5, 6, 7, 8, 10} #data is organized in ascending order

#to remove an existing value from the setA
>>> setA.discard(3)
>>> setA
{1, 2, 4, 5, 6, 7, 8, 10} #3 is removed from the original setA
```

Example 4.3
Implementation of set using Python programming language.

```
def main():
          setMarks = {45,78,90,42,78}

          cntFail=0
          for i in setMarks:
              if (i < 50):
                  cntFail=cntFail + 1
          print("Total student fail : ",cntFail)
main()
```

Output:

```
Total student fail :  2
```

Example 4.4
Set data structures supports the operations of union (|) and intersection (&) between the sets.

```
>>> setA = {1,2,4,6,7}
>>> setB = {2,7,4,9,3}

>>> setA | setB              #symbol for union is |
{1, 2, 3, 4, 6, 7, 9}
>>> setA.union(setB)         #the method union() can be used
{1, 2, 3, 4, 6, 7, 9}

>>> setA & setB              #symbol for intersection is &
{2, 4, 7}
>>> setA.intersection(setB)#the method intersection() can be used
{2, 4, 7}
```

Conclusion

Based on the detail examples as given for the four (4) different built-in data structures in Python, we conclude that the list, dictionary and set are mutable, while tuple is immutable. The indexing in the set is not applicable as compared to the other 3 built-in data structures. Set has special features of combining

or differentiating the sets by using the union, intersection or difference methods. Through these brief explanation on data structures uniqueness in Python, the readers are expected to gain basic understanding and exposure on how to implement the appropriate data structure for any problem statements.

References:

Amos, D., Bader, D., Jablonski, J. & Heisler, J. (2020), Python Basics: A Practical Introduction to Python 3: Real Python, 4th edition, e-ISBN: 9781775093336, https://static.realpython.com/python-basics-sample-chapters.pdf

Edureka (2022), Data Structures You Need to Learn in Python, Retrieved January 8th 2023, from https://www.edureka.co/blog/data-structures-in-python/

Linner, S., Lischewski, M. & Richerzhagen M. (2023, Jan 5), Python: Introduction to the Basics, Forschungszentrum Jülich, https://juser.fz-juelich.de/record/891478/files/python-2021.pdf

Othman, J., Ahmad, J.I., Abdul Wahab, N., Che Jan, N.Y., & Abd Wahab, Z.I. (2019), Programming Paradigms Concepts, (First ed.), Selangor, Malaysia: Penerbit UiTM, ISBN: 978-967363590

R20A0503. (2022), Data Structures Using Python (Lecture Notes), Department of Computer Science & Engineering (Data Science, Cyber Security, Internet of Things), Malla Reddy College of Engineering & Technology, https://mrcet.com/downloads/digital_notes/CSE/II%20Year/CS/DATA%20STRUCTURES%20using%20PYTHON%20[R20A0503].pdf

W3Schools. (2023, Jan 5), Python Introduction, https://www.w3schools.com/python/python_intro.asp

Zelle., M. J. (2009). Python Programming: An Introduction to Computer Science, Preliminary 2nd Edition, Franklin, Beedle Associates Inc.