BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 5

DCA3104

PYTHON PROGRAMMING

Unit 8

Lists

Table of Contents

1. INTRODUCTION

In the previous units, you learned about different kinds of data structures that are used in Python. These data structures are used to store data in the desired sequence. You have also learnt the properties of different data structures and sequence data types and how they store data in the computer memory. With this unit, you will understand the concept of lists used in Python. Lists are one of the most essential and basic data types. They work similarly to array in languages C and C++. They store data such as numbers, characters, and even other data structures in some form of structure.

Python has six built-in sequential data types. Among these, lists, tuples, dictionaries are the most common ones. The chapter will detail the functioning and operations that are possible in a list. Other built-in sequence types are the ones that we have already covered in the previous units. These are strings, Unicode strings, buffer objects, and xrange objects.

Python allows you to store data in a sequence in such a way that you can change the sequence in the future. You can also use data types that do not support the feature of changing the sequence while the data structure is being actively used.

Sequential data structures are used when dealing with the collection of values. You can choose to store similar or different type of data types in a single data structure. Using them can ease the difficulty level of the code and shorten it so that it is understandable to the user as well. By using data structures such as lists and tuples, you can make sure that your code is clean, well-arranged, and easy to manipulate and work with.

For example, you can create a data structure for each student that stores their name, roll number, address, and marks obtained. Such a data structure will require saving different data types such as string, integer, and floating-point real numbers. Or, you can create a data structure that exclusively stores the marks obtained by students in a class in the sequence of their roll numbers. In such cases, using data structures is the best way to make sense of the huge data in hand.

1.1 Learning Objectives

After studying the chapter, you will be able to:

- *Understand the concept of lists along with indexing and splitting.*
- *Know how to update a list value in the given list.*
- *Describe various list operations and apply them in a program when needed.*
- *Detail the working of built-in functions for lists.*

2. LISTS

Lists are sequential data structures that can contain different data types. They are mutable; thus, you can change the content and the indexing of the list while working on it. This feature makes them different from other data structures such as tuples and dictionaries and makes them one of the most versatile data structures in Python.

STUDY NOTE

Python has two kinds of data structures, or called container as they contain data. These two types of sequence containers and mappings container.

In a list, each element is assigned a unique identity. This identity refers to the address in the memory where the data is located. The values stored in a list all have different addresses and positions. These positions are known as an index. We will learn more about indexing further in the chapter.

In Python, lists are created by adding the elements of the list inside square brackets []. The elements inside the brackets are separated by commas. A list can have any kind of data type such as integer, float, strings, etc. They can also be any number of elements in a list, the limit depends upon the memory space of the computer system that you are working on.

Example:

```
a_list = []
a_list = [1, 3, 5]
a_list = [1, ‘harish’, 3]
a_list = [44, [76, 88, 99], 45, ‘a’]
```

All the above examples are valid in Python. Thus, a list can have mixed data variables, as well as a list, can be nested into another list. Such lists are called nested list.

2.1 Indexing and Splitting

For accessing all the properties of a list, one must know how to access each element of a list. For this, the elements of the list are provided with an index. As in most programming languages, in Python as well, the indexing starts from 0 (zero). The operator [] is used to access an item in the list.

If you try to access a value of the index in a string that does not exist, then the interpreter will show you an IndexError. Also, one should note that the indexes are always integers numbers. Using float or other data types as indexes will result in an error called TypeError.

Indexing in nested lists is called nested indexing. Examples are shown below.

Example:

```
a_list = [1, 3, ‘a’, ‘b’, [4, 5, ‘list’]]
print (a_list [2])
print (a_list [4] [1])

#The output of this will be as follows:
a
5
```

Python also allows negative indexing of the elements in a list. These are used when the list is to be read from right to left, that is, from the last element of the list to the first. Thus, to reverse a list, you only need to use negative indexing.

STUDY NOTE

Negative indexing begins from -1 instead of 0.

In the above example, add the code provided below.

```
print (a_list [-2])
```

The output of the command will be b.

When accessing a single element of a list, you will use indexing. But what if you need to access more than 1 elements of a list? You can, no doubt, use indexing several times. But if the elements that you want to access are sequences together in the list, then you can simply use slicing or splitting.

Slicing is the method through which you can access ranges of elements. This can be done by separating the indices of the first and last element of the desired range with a colon.

One should note that the index used on the left of the colon refers to the first element that you want to include while splitting. Whereas, the index on the right of the colon should be the number of the element that lies after the slice. Go through the example given below to better understand this.

Example:

```
odd_numbers = [1, 3, 5, 7, 9, 11, 13, 15]
print (odd_numbers [1:4])

#The output of the program will be as follows:
3, 5, 7
```

Write another command line given below.

print (odd_numbers [0:1])

The output will be 1.

Thus, the first index is inclusive in the slicing while the second index is exclusive.

Activity 1

Create a list that includes marks obtained by a student in various subjects. In the list, try accessing various elements through indexing and slicing. Use negative indices as well and find if all combinations yield the desired result. Take note where you find that the slicing does not work as expected.

2.2 Update a List Value

As lists are mutable data structures, you can change the values present in the list while it is active. This allows the user to update the list within the program without changing any of the previous code. To update a list value, you will use the assignment operator. The element that needs to be accessed will be naturally called with the help of its index.

Example:

```
even = [2, 4, 6, 8, 10]
even [0] = 0
print (even)

#The output will be as shown below.
0, 4, 6, 8, 10
```

You can also update the values of several elements in a list with the help of slicing. Use the code provided below in the above example and find the output.

even [1:4] = [2, 4, 6, 8]

SELF-ASSESSMENT QUESTIONS – 1

1. ___________________ data structures are used when you need to store the data in order.
2. Lists are enclosed in ___________________ brackets.
3. The list is a mutable data structure. (True\False)
4. Lists can include different data types at a time. (True\False)
5. Which of the following is not a valid list?
   a) [1, 2, ‘string’]
   b) (1, 2, 3)
   c) [‘string’, 1, 2, 3]
   d) [‘h’, ‘p’, ‘r’]

3. LIST OPERATORS

There are various things through which lists can be used and manipulated. You will find that some of these operations are common with all other sequence data structures. Python offers various methods through which the elements and values of a list can be changed and updated. These were discussed briefly in the previous units as well. Some of the most common operations are explained here.

3.1 Repetition

A list can be repeated n number of times with the help of the operator *. The operator creates a list that has the original list repeated the desired number of times. Here is an example of repetition in lists.

```
mylist = [ 3, 6, 9, 12, 15]
print (mylist * 3)

#The output of the program will look like this:
3, 6, 9, 12, 15, 3, 6, 9, 12, 15, 3, 6, 9, 12, 15
```

As you might know that in C when you declare an array, you have to provide the number of elements included in the array during its initialisation. However, in Python, there is no such requirement. There are cases when you need the list to acquire a definite sequence in the memory.

For example, you create an empty list. There are no elements in the list but you need space to store at least 10 elements in the list. You can introduce one element and repeat it using the * operator like [0] * 10.

If you want the interpreter to read that there is nothing in the list, or that the list is empty, then you can right None inside the square brackets of the list. An example is shown below.

Example:

```
mylist = [None] * 10
print (mylist)

#The output of the program will yield the keyword None printed 10 times.
None, None, None, None, None, None, None, None, None, None
```

3.2 Concatenation

Two or more lists can be joined together with the operator "+". This joining of one sequence with another is called concatenation. The lists can have different data types stored in them and can be of different lengths.

STUDY NOTE

Only two similar types of data structures can be concatenated together. For example, you cannot concatenate a string with a list. s

Example:

```
lista = [1, 2, 3]
listb = [4, 5, 6]
print (lista + listb)

#The output of the code will be:
1, 2, 3, 4, 5, 6
```

3.3 Membership

Membership operators are used to checking whether an element is present in a list or not. If the element is found in the list, then the operator will yield True as the output. If no such element is present in the given list, the output will be false. There are two member operators in Python, in and not in.

Example:

```
a_list = [‘b’, ‘c’, ‘d’, ‘e’, 1, 2, 3]
print (‘a’ in a_list)
print (4 not in a_list)

#The output of the program will be as follows.
False
True
```

Membership operators have multitudes of uses in real life, especially for security purposes. For example, it can be used to check whether the entered password matches the one that is saved.

3.4 Iteration

There are times when you need to manipulate all the elements present in the list with a similar set of codes. In such cases when the code needs to be performed on all elements of the list, the list is iterated. There are various ways you can use iteration in the list. These are explained below.

Using for Loop

One of the easiest ways to iterate over the complete list is through for loop. It is depicted through the example below.

```
list = [1, 2, 3, 4, 5]
for i in list:
        print (i)

#Output
1
2
3
4
5
```

Using loop and range

If you want the iterations to be operated on elements present in a set range, then you can use a loop with range (). It is shown through the example below.

```
list = [1, 2, 3, 4, 5]
length = len (list)
for i in range (length):
        print (list [i])

#Output:
1
2
3
4
5
```

Using While Loop

You can also use the while loop; however, the program may require additional statements.

```
list = [1, 2, 3, 4, 5]
length = len (list)
i = 0
while i< length:
        print (list [i])
i += 1

#Output:
1
2
3
4
5
```

Using list comprehension way

This can be the shortest and most accurate way to iterate over a list.

```
list = [1, 2, 3, 4, 5]
[print (i) for i in list]

#Output:
```

```
1
2
3
4
5
```

Using enumerate ()

The method enumerates () is used when you need to convert the list into a list of tuples that can be iterated. Its working is shown through the example.

```
list = [1, 3, 5, 7, 9]
for i, val in enumerate (list):
        print (i, ",", val)

#Output:
0, 1
1, 3
2, 5
3, 7
4, 9
```

Using Numpy

When dealing with lists with n number of dimensions, it is better to use an external library such as NumPy that can speed up the process.

```
import numpy as mylist
a = mylist.arange (9)
a = a.reshape (3, 3)
for x in mylist.nditer (a):
        print(x)

#Output:
0
1
2
3
```

```
4
5
6
7
8
```

3.5 Length

In some cases, you may need to find the length of the list. This can be done through several methods. Without using a special function, you can find the length of the list, that is, the total number of elements includes in a list, in the following way.

```
list = [1, 3, 5, 7, 9]
i = 0
for j in list:i = i +1
print (str(i))

#The output will be 5.
```

3.6 Adding Elements to a List

To add elements more items to the list, we can use the method append (). Through this method, new items can be added to the end of the list.

Example:

```
#even = [2, 4, 6, 8]
even.append (10)
print (even)
#The output will be: 2, 4, 6, 8, 10
```

Using the method extend (), you can add several items to the end of the list.

Example:

```
odd = [1, 3, 5, 7]
th=[9, 11, 13]

odd.extend (th)

print (odd)
```

```
#The output of such a program will be: 1, 3, 5, 7, 9, 11, 13
```

If you want to add an element at a desired location in the list, then the method insert () is used.

STUDY NOTE
You can add several items at a desired location in the list by using slicing.

Example:

```
odd = [1, 3, 5, 7]
odd.insert (1, 2)
print (odd)
```

Note that in the function insert (), the first number represents the index where the new element is to be included and the second number represents the value that needs to be included.

```
#The output of this program will be: 1, 2, 3, 5, 7
```

3.7 Removing Elements in a List

The keyword del is used to delete an element or several elements from the list. You can also use the keyword to delete the entire list. Here is an example that details the different ways the del keyword can be used.

```
list1 = [1, 3, 5, 7, 9, 11]
del list1[2]
print (list1)
del list1[1:3]
print (list1)
del list1
print (list1)

#The output will be as follows:
1, 3, 7, 9, 11
1, 11
```

NameError: name ‘list1’ is not defined.

Other methods can be used to delete an element from the list. These include remove (), pop (), and clear (). remove () is used to delete a given item. pop () is used to delete an item at the given index. the clear () method is used to empty the complete list.

SELF-ASSESSMENT QUESTIONS – 2

6. ____________ method removes all the elements from the list.
7. _____________ method is used to add an element at the end of the list.
8. ___________ operator is used to concatenate two or more lists.
9. The operator used to check if an element is not present in the list, such that the result obtained is True, is ___________.
10. Which of the following is a keyword used to remove elements of the list?
    a) remove
    b) del
    c) pop
    d) append
11. To check whether an element is present in the list, a membership operator is used. (True\False)
12. When one needs to iterate over a list as well as its indices, enumerate () function is used. (True\False)
13. The keyword _________ is used to delete an element or several elements from the list.
14. Two or more lists can be joined together with the operator _________.
    a) +
    b) –
    c) &
    d) ^
15. ________________ operators are used to checking whether an element is present in a list or not?

4. BUILT-IN FUNCTIONS

Python provides various in-built functions that can be used to perform various processes on the list and its elements. You can use these functions to ease the code or to know more about the given list. The prominent functions in Python for lists are explained below.

4.1 LEN (LIST)

To determine the length, that is, the number of elements present in the list, the len () function is used.

Example:

```
odd = [1, 3, 5, 7]
print (len (odd))
#The output will be 4.
```

4.2 MAX (LIST)

The function max () finds the largest value in a list.

Example:

```
odd = [1, 3, 5, 7]
print (max (odd))
#The output will be 7.
```

4.3 MIN (LIST)

The min () function returns the smallest element present in the given list.

Example:

```
odd = [1, 3, 5, 7]
print (min (odd))
#The output will be 1.
```

4.4 LIST (SEQ)

There are times when you will need to change a string or tuple into the list. In Python, you can do that with the help of the list () function. The constructor returns a list, if there are no parameters given, then it will return an empty list.

```
string1 = “python”

print (list (string1))

#The output will be: [‘p’, ‘y’, ‘t’, ‘h’, ‘o’, ‘n’]
```

SELF-ASSESSMENT QUESTIONS – 3

13. Find the output of the program given below.
    tuple1 = (‘a’, ‘e’, ‘I’, ‘o’, ‘u’)
    print (list (tuple1))
    a) (‘a’, ‘e’, ‘I’, ‘o’, ‘u’)
    b) [‘a’, ‘e’, ‘I’, ‘o’, ‘u’]
    c) (aeiou)
    d) aeiou
14. Find the output of the program given below.
    even = [10, 8, 6, 4, 2, 0]
    print (min (even))
    a) 10
    b) 0
    c) 2
    d) 1
15. If the parameter in the list () function is empty, then the list created will be ________________.
16. When comparing two lists, if the same indices in two lists contain an integer and a string, then ________________ is considered as greater.
17. ________________ value will be obtained in result if both the lists that are compared have equal values and are equal in length. iterate over a list as well as its indices, enumerate () function is used. (True or False)

5. SUMMARY

- Lists are sequential data structures that work as an array. They store multiple items or values in a single variable.
- Lists are ordered and allow duplicate values. They are mutable data structures, thus, the values in a list can be changed after its definition.
- Items in a list have a definite order or sequence which does not change during its operation. The order is represented with indices.
- The index is an integer value given to each element present in a list. They start from 0 (zero).
- One can update a value in the list with the help of the assignment operator.
- To update two or more elements, present in a list, splitting or slicing can be used. Slicing is the method through which one can access a part of the list.
- Python allows various operations that can be operated on a list. By using various operators, the list can be manipulated in different ways.
- Using the multiplication operator can repeat the values included in a list n number of times.
- By using the addition operator, one can merge two lists such that their elements create one greater list. This merging process is called concatenation.
- Membership operators are used to testing whether a value is present in a list. These are in and not in. They yield Boolean values, that is, True and False.
- One can use iterations through various methods and loops so that they can iterate a certain set of commands overall or selected elements present in a list.
- One can find the length of the list, that is, the number of elements present in the list with the in-built function available in the Python library called len ().
- There are various methods to add an element to a list. Among them, append () is used to add an element at the end of the list and insert () is used to add one or more element at the desired location in the list.
- To remove an element from the list, one can use methods such as remove (), pop (), and clear (). The keyword del can also be used to remove one or more elements from the list.

- One can compare the values and length of two lists with the help of the in-built function in Python called cmp ().
- To find the least or greatest value in a list, min () and max () functions are used, respectively.
- list () function is used to convert the parameter into the list.

6. GLOSSARY

- **List:** A sequential data function that is mutable and ordered.
- **Index:** Index is the integer value given to each element in a list to identify it uniquely.
- **Slicing:** It is the feature through which one can access parts of a list.
- **Concatenation:** It is the process of joining two lists together from end to end to create a new list.
- **Iteration:** The process of performing a similar function on one item after another.
- **Member operators:** Operators that provide True value if an element is present in a list.

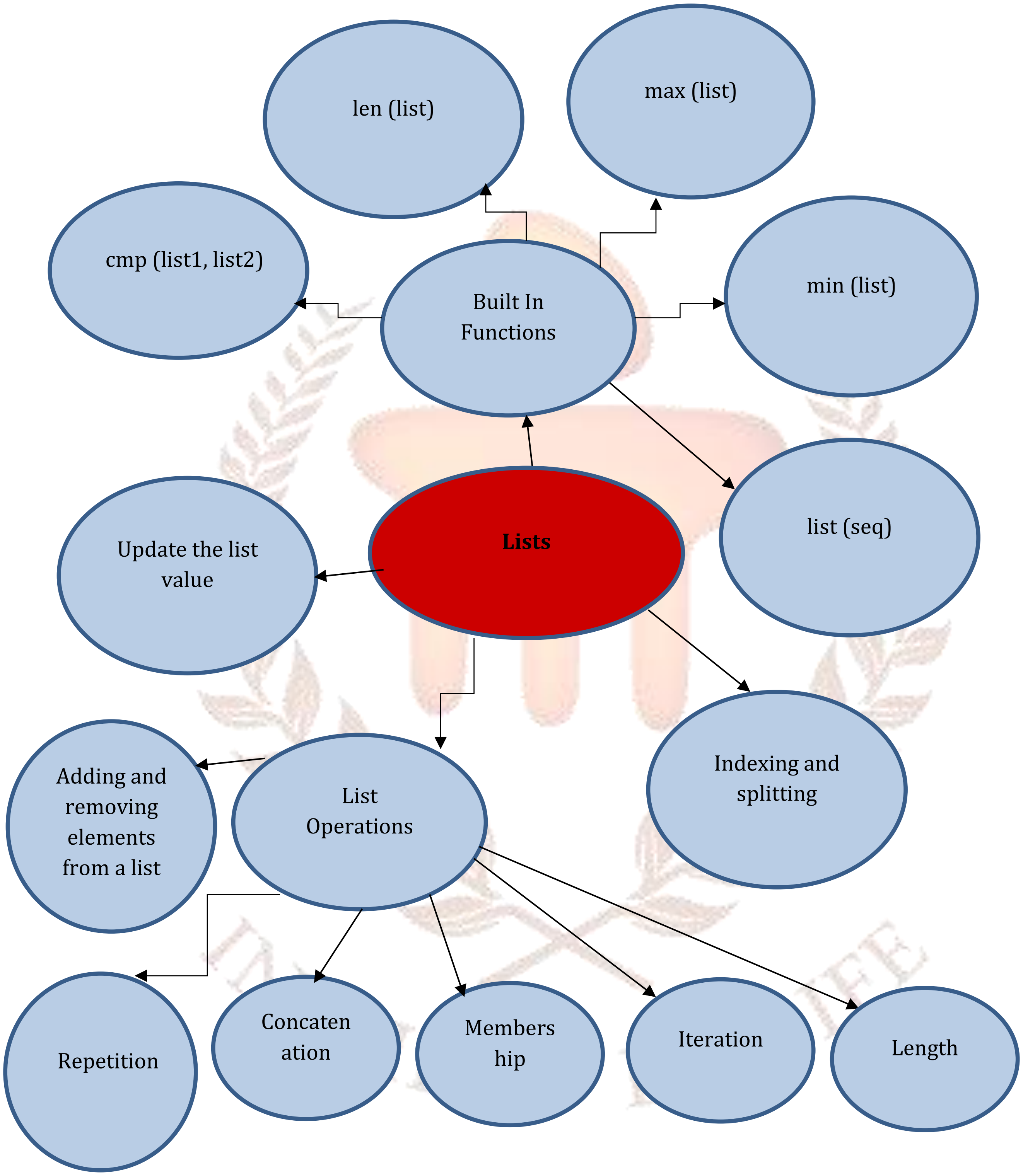

Fig 1: Conceptual Map

7. CASE STUDY

Shopping cart list creation using Python

A store is upgrading its inventory system by making the complete process online. The store owner decides that the best language to make software will be Python for its ease of learning, light-weight structure, and various functions and methods integrated into the library.

The owner wants to create an app that records the list of items that are available in the store. The list of the items should be created such that items can be added, removed, and updated in the list in the future.

Along with that, the owner needs a method through which the sales of all items can be compared every month. That is, there should be a list that records the sales of each item available in the store for each month. Based on the results obtained from the comparison of these lists, the owner plans to decide whether to continue stocking an item in the store or not.

The program should also provide the item that was sold the most in a month and one that was sold the least. It should also let the owner remove various items at a time as their availability and demand changes according to the season.

Imagine yourself as a part of the team that has been tasked to create such a program. Discuss and derive an answer for the following questions.

***Source**: bestpy.com*

Questions:

1. Which data structure do you plan to use? Discuss the benefits and disadvantages of each data structure to create such a program.
2. Which functions and methods you can employ in the program so that it delivers every expectation of the store owner?

8. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. Write a Python program and use the concept of indexing to print a date, given year, month, and day as numbers.

Q2. Write a Python program that prints a text in a box of a given length. The output should look something like this:

```
+————————————————————————————————————————+
|                                        |
|              I am a code               |
|                                        |
+————————————————————————————————————————+
```

Q3. Write a Python program that checks the user's name and PIN code of a user.

Q4. Write a Python program to print the sum of all elements in a list.

Q5. Write a Python program that prints the largest number present in a list.

LONG ANSWER QUESTIONS

Q1. Write a Python program that removes duplicate from a list.

Q2. Write a Python program that finds if the list of words is longer than a given list of words.

Q3. Write a Python program that shuffles the order of the list to print it in a specified order.

Q4. Write a Python program to select an element from a list randomly.

Q5. Write a Python program to check whether a list contains a sub-list.

8.1 Answers

1. List
2. Square
3. True
4. True
5. B
6. clear ()
7. append ()
8. +
9. Membership

10. b) del
11. True
12. True
13. Del
14. +
15. Membership
16. B) [‘a’, ‘e’, ‘I’, ‘o’, ‘u’]
17. B) 0
18. empty
19. string
20. 0

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS:

Answer 1:

```
months = [
 'January',
 'February',
 'March',
 'April',
 'May',
 'June',
 'July',
 'August',
 'September',
 'October',
 'November',
 'December'
]
endings = ['st', 'nd', 'rd'] + 17 * ['th'] + ['st', 'nd', 'rd'] + 7 * ['th'] + ['st']
year = input('Year: ')
month = input('Month (1-12): ')
```

```
day = input('Day (1-31): ')
month_number = int(month)
day_number = int(day)
month_name = months[month_number-1]
ordinal = day + endings[day_number-1]
print (month_name + ' ' + ordinal + ', ' + year)
```

sAnswer 2:

```
sentence = input("Sentence: ")
screen_width = 80
text_width = len(sentence)
box_width = text_width + 6
left_margin = (screen_width - box_width) // 2
print(" ")
print (' ' * left_margin + '+' + '-' * (box_width-2) + '+')
print (' ' * left_margin + '| ' + ' ' * text_width + ' |')
print (' ' * left_margin + '| ' + sentence + ' |')
print (' ' * left_margin + '| ' + ' ' * text_width + ' |')
print (' ' * left_margin + '+' + '-' * (box_width-2) + '+')
```

Answer 3:

```
database = [
 ['albert', '1234'],
 ['dilbert', '4242'],
 ['smith', '7524'],
 ['jones', '9843']
]
username = input('User name: ')
pin = input('PIN code: ')
if [username, pin] in database:
  print ('Access granted')
```

Answer 4:

```
def sum_list(items):
sum_numbers = 0
   for x in items:
sum_numbers += x
   return sum_numbers
print(sum_list([1,2,-8]))
```

Answer 5:

```
def max_num_in_list(list ):
   max = list [ 0]
   for a in list:
      if a > max:
         max = a
   return max
print (max_num_in_list([1, 2, -8, 0]))
```

LONG ANSWER QUESTIONS:

Answer 1:

```
a = [10,20,30,20,10,50,60,40,80,50,40]
dup_items = set ()
uniq_items = []
for x in a:
   if x not in dup_items:
uniq_items.append(x)
dup_items.add(x)
print(dup_items)
```

Answer 2:

```
def is_first_list_longer(list1, list2):
   return len(list1) > len(list2)
words1 = ["apple", "banana", "cherry"]
words2 = ["dog", "elephant"]
```

```
print(is_first_list_longer(words1, words2))  # Prints: True
words1 = ["apple", "banana"]
words2 = ["dog", "elephant", "zebra", "lion"]
print(is_first_list_longer(words1, words2))  # Prints: False
```

Answer 3: from random import shuffle

```
color = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
shuffle(color)
print(color)
```

Answer 4: import random

```
color_list = ['Red', 'Blue', 'Green', 'White', 'Black']
print(random.choice(color_list))
```

Answer 5:

```
def is_Sublist(l, s):
sub_set = False
        if s == []:
sub_set = True
elif s == l:
sub_set = True
eliflen(s) >len(l):
sub_set = False
        else:
                for i in range(len(l)):
                        if l[i] == s[0]:
                                n = 1
                                while (n <len(s)) and (l[i+n] == s[n]):
                                        n += 1
                                if n == len(s):
sub_set = True
        return sub_set
a = [2,4,3,5,7]
```

```
b = [4,3]
c = [3,7]
print (is_Sublist(a, b))
print (is_Sublist(a, c))
```

9. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

1. Eric Matthes (2016), Python Crash Course: A Hands-On, Project-Based Introduction to Programming.
2. John M. Zelle (2009), Python Programming: An Introduction to Computer Science (Preliminary Second Edition).
3. Mark Lutz (2011), Python Programming: A Powerful Object-Oriented Programming (Fourth Edition).
4. Sebastian Raschka (2017), Python Machine Learning - Machine Learning and Deep Learning with Python (Edition 2)

E-REFERENCES:

- Python Programming Certification Training Course, last viewed on March 25, 2021 <https://www.edureka.co/python-programming-certification-training >
- Python Tutorials and Sample Programs, last viewed on March 25, 2021 < https://www.w3schools.com/python/ >
- Integrated Development Environments, last viewed on March 25, 2021 < https://wiki.python.org/moin/IntegratedDevelopmentEnvironments >