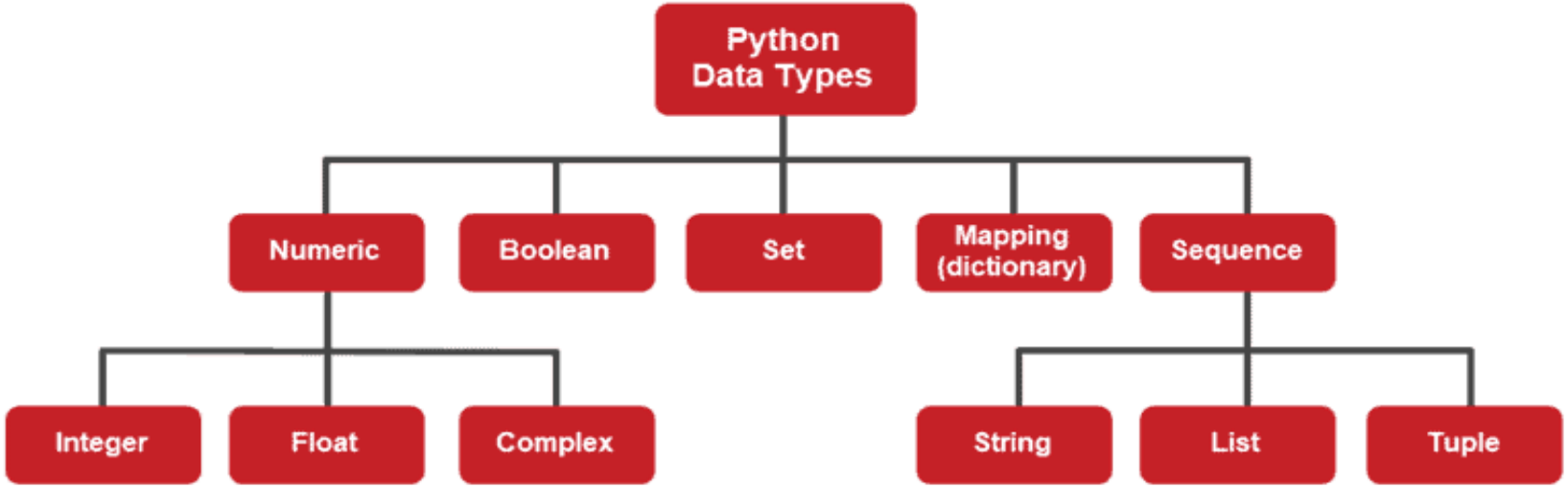
Stack in Python

Data types in Python

- Till now, we have learnt about different data types in Python for handling values, and sequence data types can contain elements of either same type or different types.



Data structure

- Grouping of different types in Python for storing values is referred as data structure.

- A data structure defines a mechanism to store, organise and access data along with operations that can be efficiently performed on the data.

  For example,

  - string is a data structure containing a sequence of elements where each element is a character.
  - list is a sequence data structure in which each element may be of different types.

Data structure

- We can apply different operations like reversal, slicing, counting of elements, etc. on list and string.
- Hence, a data structure organises multiple elements in a way so that certain operations on each element as well as the collective data unit could be performed easily.
- In programming, stack is another data structure which has many applications.

Introduction to stack

- You must have seen piles of books in the library or stack of plates at home.
- To put another book or another plate in such a pile, we always place (add to the pile) the object at the top only.
- Likewise, to remove a book or a plate from such a pile, we always remove the object from the top only. This is because in a large pile, it is inconvenient to add or remove an object from in between or bottom.

Introduction to stack

- Such an arrangement of elements in a linear order is called a stack.
- We add new elements or remove existing elements from the same end, commonly referred to as the *top of the stack*.
- *It thus follows the Last-In-First-out (LIFO) principle. That is, the element which was inserted last (the most recent element) will be the first one to be taken out from the stack.*

Examples of stack

- Some of the examples of stack in real-life are:
  - Pile of clothes in an almirah
  - Multiple chairs in a vertical pile
  - Bangles worn on wrist
  - Pile of boxes of eatables in pantry or on a kitchen shelf

Applications of stack

- Some examples of application of stack in programming are as follows:

1. When we need to reverse a string, the string is traversed from the last character till the first character. i.e. characters are traversed in the reverse order of their appearance in the string. This is very easily done by putting the characters of a string in a stack.

2. We use text/image editor for editing the text/image where we have options to redo/undo the editing done. When we click on the redo /undo icon, the most recent editing is redone/undone. In this scenario, the system uses a stack to keep track of changes made.

Applications of stack

3. While browsing the web, we move from one web page to another by accessing links between them. In order to go back to the last visited web page, we may use the back button on the browser.

- Let us say we accessed a web page P1 from where we moved to web page P2 followed by browsing of web page P3. Currently, we are on web page P3 and want to revisit web page P1. We may go to a previously visited web page by using the BACK button of the browser. On clicking the BACK button once, we are taken from web page P3 to web page P2, another click on BACK shows web page P1.
- In this case, the history of browsed pages is maintained as stack.

Operations on stack

- As we have already discussed, a stack is a mechanism that implements LIFO arrangement hence elements are added and deleted from the stack at one end only.
- The end from which elements are added or deleted is called TOP of the stack.
- Two fundamental operations performed on the stack are
  - PUSH and
  - POP

Operations on stack: PUSH

- PUSH adds a new element at the TOP of the stack. It is an insertion operation. We can add elements to a stack until it is full.

- A stack is full when no more elements can be added to it. Trying to add an element to a full stack results in an exception called ‘overflow’.

Operations on stack: POP

- POP operation is used to remove the top most element of the stack, that is, the element at the TOP of the stack. It is a delete operation.

- We can delete elements from a stack until it is empty i.e. there is no element in it. Trying to delete an element from an empty stack results in an exception called ‘underflow’.

Implementation of stack in Python

- We have learnt so far that a stack is a linear and ordered collection of elements. The simple way to implement a stack in Python is using the data type *list*. We can fix either of the sides of the list as TOP to insert/remove elements.

- It is to be noted that we are using built-in methods append() and pop() of the list for implementation of the stack. As these built-in methods insert/delete elements at the rightmost end of the list, hence explicit declaration of TOP is not needed.

Implementation of stack in Python

- Let us write a program to create a STACK in which we will:
  - insert/delete elements
  - check if the STACK is empty
  - find the number of elements in the STACK
  - read the value of the topmost element in the STACK

- The program shall define the following functions to perform these operations:

Create an empty stack

- Let us create an empty stack named Stack. We will do so by assigning an empty list to the identifier named Stack:

```
#creating an empty stack
Stack = list()
```

Check if the stack is empty

- A function named isEmpty to check whether the stack Stack is empty or not. Remember trying to remove an element from an empty stack would result in 'underflow'. This function returns True if the stack is empty, else returns False.

```
#function to check whether Stack is empty
def isEmpty(Stack):
    if len(Stack)==0:
        return True
    else:
        return False
```

Inserting an element in the stack

- A function named opPush to insert (PUSH) a new element in stack. This function has two parameters –

  - the name of the stack in which the element is to be inserted (Stack) and
  - the element that needs to be inserted

- We know that insertion of an element is always done at the TOP of the stack. Hence, we shall use the built-in method append() of list to add an element to the stack that always adds at the end of the list.

Inserting an element in the stack

- As there is no limit on the size of list in Python, the implemented stack will never be full unless there is no more space available in memory. Hence, we will never face 'overflow' (no space for new element) condition for stack.

```
#function to insert an element
def opPush(Stack, element):
    Stack.append(element)
```

find number of elements in the stack

- A function named size to read the number of elements in the Stack.
- We will use the len() function of list in Python to find the number of elements in the Stack.

```
#function to determine number of elements in the stack
def size(Stack):
    return len(Stack)
```

Read the value of the topmost element

- A function named top to read the most recent element (TOP) in the Stack.

```
#function to read the value of the topmost element
def top(Stack):
    if isEmpty(Stack):
        print("Stack is empty")
        return None
    else:
        x = len(Stack)
        element = Stack[x-1]
        return element
```

Delete the topmost element of stack

- A function named opPop to delete the topmost element from the stack. It takes one parameter –

  - the name of the stack (Stack) from which element is to be deleted

  and returns the value of the deleted element.

- The function first checks whether the stack is empty or not. If it is not empty, it removes the topmost element from it. We shall use the built-in method pop() of Python list that removes the element from the end of the list.

Delete the topmost element of stack

```
#function to delete the topmost element of the stack
def opPop(Stack):
    if isEmpty(Stack):
        print("Underflow")
        return None
    else:
        return(Stack.pop())
```

Display the contents of the stack

- A function named display to show the contents of the stack.

```
#function to display the contents of the stack
def display(Stack):
    x = len(Stack)
    print("Current elements in the stack are: ")
    for i in range(x-1, -1, -1):
        print(Stack[i], end=" ")
```

Code:

```
"""
Implementation of Stack in Python using list
@author: Himanshu Mudgal
"""

#function to check whether Stack is empty
def isEmpty(Stack):
    if len(Stack)==0:
        return True
    else:
        return False

#function to insert an element
def opPush(Stack, element):
    Stack.append(element)

#function to determine number of elements in the stack
def size(Stack):
    return len(Stack)

#function to read the value of the topmost element
def top(Stack):
    if isEmpty(Stack):
        print("Stack is empty")
        return None
    else:
        x = len(Stack)
        element = Stack[x-1]
        return element

#function to delete the topmost element of the stack
def opPop(Stack):
    if isEmpty(Stack):
        print("Underflow")
        return None
    else:
        return(Stack.pop())
```

Code:

```
38
39  #function to display the contents of the stack
40  def display(Stack):
41      x = len(Stack)
42      print("Current elements in the stack are: ")
43      for i in range(x-1, -1, -1):
44          print(Stack[i], end=" ")
45
46  #main function
47  #creating an empty stack
48  Stack = list()
49
50  #adding an element to the list
51  n = int(input("Enter the number of elements you want to insert: "))
52  for i in range(n):
53      element = input("Enter an element to push into stack: ")
54      opPush(Stack, element)
55
56  #displayting the elements of stack after the addition of element
57  display(Stack)
58
59  #display the last element of the stack
60  print("\nTop element of the stack is:", top(Stack))
61
62  #delete the top element of the stack
63  print("Deleted element of the stack: ", opPop(Stack))
64
65  #diaplay the elements of the stack
66  display(Stack)
67
```

Output:

```
In [8]: runfile('C:/Users/Vaibhav/.spyder-py3/temp.py', wdir='C:/Users/
Vaibhav/.spyder-py3')

Enter the number of elements you want to insert: 5

Enter an element to push into stack: Hi

Enter an element to push into stack: Him

Enter an element to push into stack: Hima

Enter an element to push into stack: Rahul

Enter an element to push into stack: Sharma
Current elements in the stack are:
Sharma Rahul Hima Him Hi
Top element of the stack is: Sharma
Deleted element of the stack:  Sharma
Current elements in the stack are:
Rahul Hima Him Hi
```

Summary

- Introduction to stack
- Applications of stack
- Operations on stack: PUSH and POP
- Implementation of stack in Python using lists

Assignment - 1

- Note down the summary points in your notebook (first 5 points); page- 51
- NCERT questions: Q1 – Q2, page – 52

Programming Assignment - 1

- Write a Python program to implement a stack using list.
- Write a program to reverse a string using stack.