# ANSWERS

# PYTHON ASSIGNMENT – 12

1. **A  python program to create a linked list and performed operations on the list.**

**Ans -**

class Node:

    def \_\_init\_\_(self, dataval=None):

        self.dataval = dataval

        self.nextval = None

class SinglyLinkedList:

    def \_\_init\_\_(self):

        self.headval = None

    # Print the linked list

    def listprint(self):

        printval = self.headval

        while printval is not None:

            print (printval.dataval)

            printval = printval.nextval

    # Inserting at the Beginning of the Linked List

    def AtBegining(self, newdata):

        NewNode = Node(newdata)

        # Update the new nodes next val to existing node

        NewNode.nextval = self.headval

        self.headval = NewNode

    # Inserting at the End of the Linked List

    # Function to add newnode

    def AtEnd(self, newdata):

        NewNode = Node(newdata)

        if self.headval is None:

            self.headval = NewNode

            return

        laste = self.headval

        while(laste.nextval):

            laste = laste.nextval

        laste.nextval=NewNode

    # Inserting in between two Data Nodes

    # Function to add node

    def Inbetween(self,middle\_node,newdata):

        if middle\_node is None:

            print("The mentioned node is absent")

            return

        NewNode = Node(newdata)

        NewNode.nextval = middle\_node.nextval

        middle\_node.nextval = NewNode

    # Removing an Item form a Liked List

    # Function to remove node

    def RemoveNode(self, Removekey):

        Head = self.headval

        if (Head is not None):

            if (Head.dataval == Removekey):

                self.headval = Head.nextval

                Head = None

                return

        while (Head is not None):

            if Head.dataval == Removekey:

                break

            prev = Head

            Head = Head.nextval

        if (Head == None):

            return

        prev.nextval = Head.nextval

        Head = None

list = SinglyLinkedList()

list.headval = Node("Mon")

e2 = Node("Tue")

e3 = Node("Wed")

# Link first Node to second node

list.headval.nextval = e2

# Link second Node to third node

e2.nextval = e3

list.headval.nextval = e2

e2.nextval = e3

list.AtBegining("Sun")

list.AtEnd("Thu")

list.Inbetween(list.headval.nextval,"Fri")

list.RemoveNode("Tue")

list.listprint()

1. **A  python program to create a Stack class that can perform some important operations.**

**Ans -**

# There are two types of operations in Stack-

# push – To add data into the stack.

# Pop – To remove data from the stack.

class Stack:

    def \_\_init\_\_(self):

        self.stack = []

    # PUSH into a Stack

    def add(self, dataval):

        # Use list append method to add element

        if dataval not in self.stack:

            self.stack.append(dataval)

            return True

        else:

            return False

    # POP from a Stack

    # Use list pop method to remove element

    def remove(self):

        if len(self.stack) <= 0:

            return ("No element in the Stack")

        else:

            return self.stack.pop()

    # Use peek to look at the top of the stack

    def peek(self):

        return self.stack[-1]

AStack = Stack()

AStack.add("Mon")

AStack.add("Tue")

AStack.peek()

print("Element at the top: ", AStack.peek())

AStack.add("Wed")

AStack.add("Thu")

print("Element at the top: ",AStack.peek())

print("Removed top element is: ", AStack.remove())

print("Removed top element is: ", AStack.remove())

1. **A python program to perform various operations on a stack using Stack class.**
2. **A python program to create a Queue class using list methods.**

**Ans -**

class Queue:

    def \_\_init\_\_(self):

        self.queue = list()

    # Adding Elements to a Queue

    def addtoq(self,dataval):

        # Insert method to add element

        if dataval not in self.queue:

            self.queue.insert(0,dataval)

            return True

        return False

    # Removing Element from a Queue

    # Pop method to remove element

    def removefromq(self):

        if len(self.queue) > 0:

            return self.queue.pop()

        return ("No elements in Queue!")

    def size(self):

        return len(self.queue)

TheQueue = Queue()

TheQueue.addtoq("Mon")

TheQueue.addtoq("Tue")

TheQueue.addtoq("Wed")

print(TheQueue.size())

print(TheQueue.queue)

print(TheQueue.removefromq())

print(TheQueue.removefromq())

1. **A python program to perform some operations on a queue.**

**Ans -**

# Queue using list

queue = ["Amar", "Akbar", "Anthony"]

# Add new element into the queue

queue.append("Ram")

queue.append("Iqbal")

print(queue)

# Removes the first item

print(queue.pop(0))

print(queue)

# Removes the first item

print(queue.pop(0))

print(queue)

1. **A python program to create and use deque.**

**Ans -**

import collections

DoubleEnded = collections.deque(["Mon","Tue","Wed"])

# Adding element to the right side

DoubleEnded.append("Thu")

DoubleEnded.append("Mon")

print ("Appended at right: ")

print (DoubleEnded)

# Adding element to the left side

DoubleEnded.appendleft("Sun")

print ("Appended at left is: ")

print (DoubleEnded)

# Deleting element from the right side

DoubleEnded.pop()

print ("Deleting from right: ")

print (DoubleEnded)

# Deleting element from the left side

DoubleEnded.popleft()

print ("Deleting from left: ")

print (DoubleEnded)

# Inserting the value Thu at 2nd position

DoubleEnded.insert(2, "Fri")

print ("Inserting the element: ")

print (DoubleEnded)

# Count the occurrences of  Mon

print ("Count of element: ")

print (DoubleEnded.count("Mon"))

# Remove the first occurrence of Mon

print ("Deleting the first occurence of Mon: ")

DoubleEnded.remove("Mon")

print (DoubleEnded)