

```
1 print("WELCOME! TO DATA SCIENCE") # Kurt Russel Villamor CPE22S3
```

```
WELCOME! TO DATA SCIENCE
```

```
1 ""SINGLY"" # Traversing only forward
2 class Node:
3     def __init__(self, data=None, next = None):
4         self.data = data
5         self.next = next
6
7     def add_node(self, node):
8         if self.next is None:
9             self.next = node
10        else:
11            while self is not None:
12                if self.next is None:
13                    self.next = node
14                    break
15                self = self.next
16
17    def delete_node(self, data):
18        if self.data == data:
19            if self.next is None:
20                self.data = None
21            else:
22                self.data = self.next.data
23                self.next = self.next.next
24        else:
25            while self is not None:
26                if self.next is None:
27                    pass
28                else:
29                    if self.next.data == data:
30                        self.next = self.next.next
31                    self = self.next
32
33    def print_nodes(self):
34        while self is not None:
35            print(self.data)
36            self = self.next
37
38
```

```
1 # Nodes can be added Different Approaches
```

```
2 root = Node(data = 'C')
3 root.next=Node(data = 'P')
4 root.add_node(Node(data = 'E'))
```

```
1 root.print_nodes() # Print Nodes
```

```
C
P
E
```

```
1 root.delete_node('P') # Deletes The Middle Node in C-P-E
```

```
1 root.print_nodes() # Successfully Deletes P
```

```
C
E
```

```
1 root.delete_node('C' ) # Deletes The Root Node in C-P-E
```

```
1 root.print_nodes() # Successfully Deletes C
```

```
E
```

```
1 root.delete_node('E') # Deletes The Root Node/Last Node in C-P-E
```

```
1 root.print_nodes() # Successfully Deletes E and Returns None
```

None

```
1 """DOUBLY""" # Traversing forward and Backward
2 class DNode:
3     def __init__(self, data, next = None, prev = None):
4         self.data = data
5         self.next = next
6         self.prev = prev
7
8     def add_node(self, node):
9         if self.next is None:
10             self.next = node
11             node.prev = self
12         else:
13             while self is not None:
14                 if self.next is None:
15                     self.next = node
16                     node.prev = self
17                     break
18                 self = self.next
19
20     def delete_node(self, data):
21         while self is not None:
22             if self.next is not None:
23                 if self.next.data == data:
24                     self.next = self.next.next
25                     self.next.prev = self
26                 self = self.next
27
28     def print_nodes(self):
29         while self is not None:
30             print(self.data)
31             self = self.next
32
33
34 1 # Nodes can be added Different Approaches
35 2 Droot = DNode(data = 'C')
36 3 Droot2 = DNode(data = 'P')
37 4 Droot.add_node(Droot2)
38 5 Droot.add_node(DNode(data = 'E'))
39
40
41 1 Droot.print_nodes() # Print Nodes
42
43     C
44     P
45     E
46
47
48 1 print(Droot2.prev.data) # Print the previous data of P
49
50     C
51
52
53 1 print(Droot.prev) # Print the previous data of C
54
55     None
56
57
58 1 Droot.delete_node('P') # Deletes the Middle Node in CPE
59
60
61 1 Droot.print_nodes() # Succesfully Deletes P
62
63     C
64     E
65
66
67 1 print(Droot.next.data) # Printing E
68
69     E
70
71
72 1 trial = Droot.next # Setting trial for checking E
73
74
75 1 print(trial.prev.data) # prints the Previous of E
```

C

```
1 """CIRCULAR""" # Traversing forward and Backward and head is also Tail
2 class DNode:
3     def __init__(self, data, next = None, prev = None):
4         self.data = data
5         self.next = next
6         self.prev = prev
7
8     def add_node(self, node):
9         if self.next is None:
10             self.next = node
11             node.prev = self
12         else:
13             while self is not None:
14                 if self.next is None:
15                     self.next = node
16                     node.prev = self
17                     break
18                 self = self.next
19
20     def delete_node(self, data):
21         while self is not None:
22             if self.next is not None:
23                 if self.next.data == data:
24                     self.next = self.next.next
25                     self.next.prev = self
26                 self = self.next
27
28
29     def print_nodes(self):
30         while self is not None:
31             print(self.data)
32             self = self.next
```

```
1 # Nodes can be added Different Approaches
2 Droot = DNode(data = 'C')
3 Droot2 = DNode(data = 'P')
4 Droot.add_node(Droot2)
5 Droot.add_node(DNode(data = 'E'))
```

```
1 print(Droot2.next.next)
```

None