# Singly Linked List – Music Playlist

class Node:

def \_\_init\_\_(self, data: str):

self.data = data

self.next = None

class Playlist:

def \_\_init\_\_(self):

self.head = None

# ---------- insert ----------

def insert\_at\_beginning(self, song: str) -> None:

new = Node(song)

new.next = self.head

self.head = new

def insert\_at\_end(self, song: str) -> None:

new = Node(song)

if self.head is None:

self.head = new

return

cur = self.head

while cur.next:

cur = cur.next

cur.next = new

def insert\_at\_position(self, pos: int, song: str) -> None:

"""1-based position. If pos <= 1, insert at beginning.

If pos is greater than length+1, inserts at end."""

if pos <= 1 or self.head is None:

self.insert\_at\_beginning(song)

return

new = Node(song)

cur = self.head

i = 1

while cur.next and i < pos - 1:

cur = cur.next

i += 1

new.next = cur.next

cur.next = new

# ---------- delete ----------

def delete\_song(self, song: str) -> bool:

"""Delete first occurrence of song. Returns True if deleted."""

cur = self.head

prev = None

while cur:

if cur.data == song:

if prev is None:

self.head = cur.next

else:

prev.next = cur.next

return True

prev, cur = cur, cur.next

return False

# ---------- search ----------

def find(self, song: str) -> int:

"""Return 1-based position of song, or -1 if not found."""

cur = self.head

pos = 1

while cur:

if cur.data == song:

return pos

cur = cur.next

pos += 1

return -1

# ---------- display ----------

def display(self) -> None:

if self.head is None:

print("Playlist is empty.")

return

cur = self.head

print("Playlist:", end=" ")

while cur:

arrow = " -> " if cur.next else ""

print(cur.data, end=arrow)

cur = cur.next

print()

def menu():

pl = Playlist()

options = {

"1": "Insert at beginning",

"2": "Insert at end",

"3": "Insert at position",

"4": "Delete a song",

"5": "Find a song",

"6": "Display playlist",

"0": "Exit",

}

while True:

print("\n--- Music Playlist (Singly Linked List) ---")

for k in sorted(options):

print(f"{k}. {options[k]}")

choice = input("Enter choice: ").strip()

if choice == "1":

song = input("Song name: ")

pl.insert\_at\_beginning(song)

elif choice == "2":

song = input("Song name: ")

pl.insert\_at\_end(song)

elif choice == "3":

song = input("Song name: ")

pos = int(input("Position (1-based): "))

pl.insert\_at\_position(pos, song)

elif choice == "4":

song = input("Song to delete: ")

if pl.delete\_song(song):

print("Deleted.")

else:

print("Song not found.")

elif choice == "5":

song = input("Song to search: ")

pos = pl.find(song)

print(f"Found at position {pos}." if pos != -1 else "Not found.")

elif choice == "6":

pl.display()

elif choice == "0":

print("Bye!")

break

else:

print("Invalid choice.")

if \_\_name\_\_ == "\_\_main\_\_":

menu()

Output

--- Music Playlist (Singly Linked List) ---

0. Exit

1. Insert at beginning

2. Insert at end

3. Insert at position

4. Delete a song

5. Find a song

6. Display playlist

Enter choice: 2

Song name: song a

--- Music Playlist (Singly Linked List) ---

0. Exit

1. Insert at beginning

2. Insert at end

3. Insert at position

4. Delete a song

5. Find a song

6. Display playlist

Enter choice: 2

Song name: song b

--- Music Playlist (Singly Linked List) ---

0. Exit

1. Insert at beginning

2. Insert at end

3. Insert at position

4. Delete a song

5. Find a song

6. Display playlist

Enter choice: 2

Song name: song c

--- Music Playlist (Singly Linked List) ---

0. Exit

1. Insert at beginning

2. Insert at end

3. Insert at position

4. Delete a song

5. Find a song

6. Display playlist

Enter choice: 6

Playlist: song a -> song b -> song c