Nama : Wildhan Dzikrihantara
NIM : J0403251098
Angka Akhir NIM : Genap

Latihan 1, 2, dan 4.

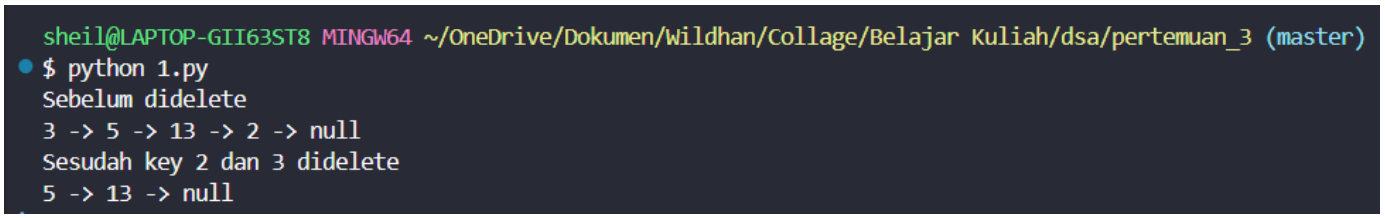
1. **Latihan 1**

```python
"""
================================
Latihan 1

Wildhan Dzikrihantara
J0403251098
================================
"""

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None
        self.tail = None # Tambahkan pointer tail

    def insert_at_end(self, data):
        new_node = Node(data)
        if not self.head: # Jika linked list kosong
            self.head = new_node
            self.tail = new_node # Tail juga menunjuk ke node
pertama
        else:
            self.tail.next = new_node # Sambungkan tail ke node
baru
```

```
            self.tail = new_node # Update tail ke node baru

    def display(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("null")

    # Fungsi delete_node menghapus node berdasarkan value node
tersebut
    def delete_node(self, key):
        temp    =   self.head
        if temp and temp.data == key:
            self.head = temp.next
            temp = None
            return
        prev = None
        while temp and temp.data != key:
            prev = temp
            temp = temp.next
        if temp is  None:
            return
        prev.next = temp.next
        temp = None

# Contoh Penggunaan
ll = LinkedList()
ll.insert_at_end(3)
ll.insert_at_end(5)
ll.insert_at_end(13)
ll.insert_at_end(2)
ll.display() # output : 3 -> 5 -> 13 -> 2 -> null
ll.delete_node(2) # Contoh Penggunaan delete_node, 2 dihapus dari
linked
ll.delete_node(3) # Contoh Penggunaan delete_node, 3 dihapus dari
```

```
ll.display() # output : 5 -> 13 -> null
```

Berikut merupakan screenshot hasil latihan 1.




## 2. Latihan 2

```python
"""
================================
Latihan 2

Wildhan Dzikrihantara
J0403251098
================================
"""

class Node:
    def __init__(self,  data):
        self.data   =   data
        self.next   =   None

class CircularSinglyLinkedList:
    def __init__(self):
        self.head   =   None
        self.tail   =   None # Tambahkan pointer tail

    # Function insert node dengan banyak value atau single value
    def insert_node(self, data):
```

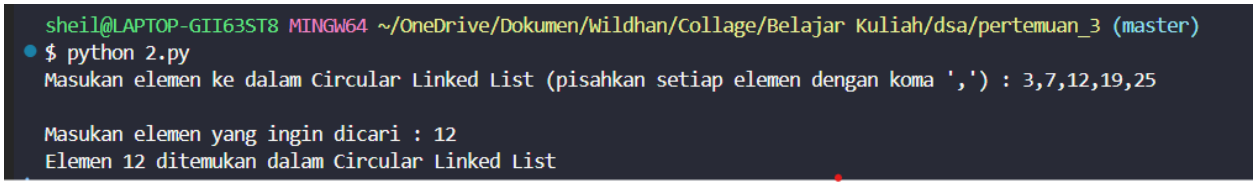
```
        # Inisialisasi dan assign value dengan mengecek banyak
value terlebih dahulu agar tipenya tepat
        data_set = None

        if len(data) == 1:
            data_set = data
        elif len(data) > 1:
            data_set = data.split(",")

        # Insert values ke nodes
        if data_set is not None:
            for data_insert in data_set:
                new_node = Node(data_insert)
                if  not self.head:  # Jika linked list kosong
                    self.head = new_node
                    self.tail = new_node
                    self.tail.next = self.head # Circular link ke
dirinya sendiri
                else:
                    self.tail.next  =   new_node # Sambungkan
tail ke node baru
                    self.tail   =   new_node # Update tail ke
node baru
                    self.tail.next  =   self.head # Circular link
kembali ke
        else:
            new_node = Node(data_set)
            if  not self.head:  # Jika linked list kosong
                self.head = new_node
                self.tail = new_node
                self.tail.next = self.head # Circular link ke
dirinya sendiri
            else:
                self.tail.next  =   new_node # Sambungkan tail ke
node baru
```

```
                self.tail   =   new_node # Update tail ke node
baru
                self.tail.next  =   self.head # Circular link
kembali ke

    def search_key(self, key):
        if self.head.data is None:
            print("Circular Linked List kosong, tidak ada elemen
yang bisa dicari")
            return

        temp = self.head
        while True:
            if temp.data == key:
                print(f"Elemen {key} ditemukan dalam Circular
Linked List")
                return
            temp = temp.next
            if temp == self.head:
                break

        print(f"Elemen {key} tidak ditemukan dalam Circular
Linked List")



    def display(self):
        if  not self.head:
            print("List is  empty")
            return

        temp = self.head
        print(temp.data, end=" -> ")
        temp = temp.next
        while temp != self.head:
            print(temp.data, end=" -> ")
```

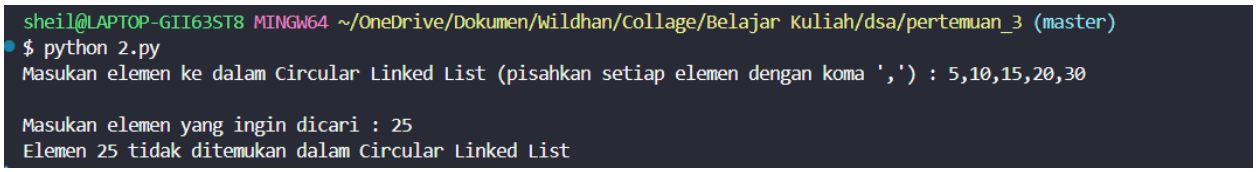
```
            temp = temp.next

        print("...  (back   to  head)")

# Penerapan Latihan 2
cll = CircularSinglyLinkedList()

# Insert value/values
insert_values = input("Masukan elemen ke dalam Circular Linked
List (pisahkan setiap elemen dengan koma ',') : ")
cll.insert_node(insert_values)

# Search key
search_val = input("\nMasukan elemen yang ingin dicari : ")
cll.search_key(search_val)
```

Berikut merupakan screenshot latihan 2.

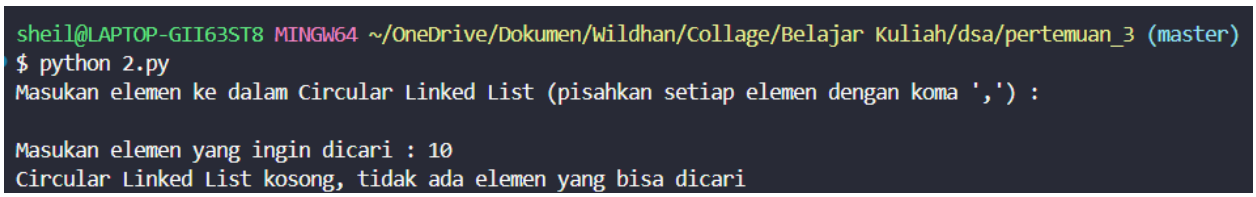
1. Contoh tampilan #1




2. Contoh tampilan #2




3. Contoh tampilan #3



### 3. Latihan 4

Python

```
"""
================================
Latihan 4

Wildhan Dzikrihantara
J0403251098
================================
"""

class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None
        self.tail = None # Tambahkan pointer tail

    # Function insert node dengan banyak value atau single value
    def insert_node(self, data):
        # Inisialisasi dan assign value dengan mengecek banyak
value terlebih dahulu agar tipenya tepat
        data_set = None

        # Mengecek banyaknya key/value agar bisa disesuaikan
untuk dimasukan ke nodes
        if len(data) == 1:
            data_set = data
        elif len(data) > 1:
            data_set = data.split(",")
        else:
            return
```

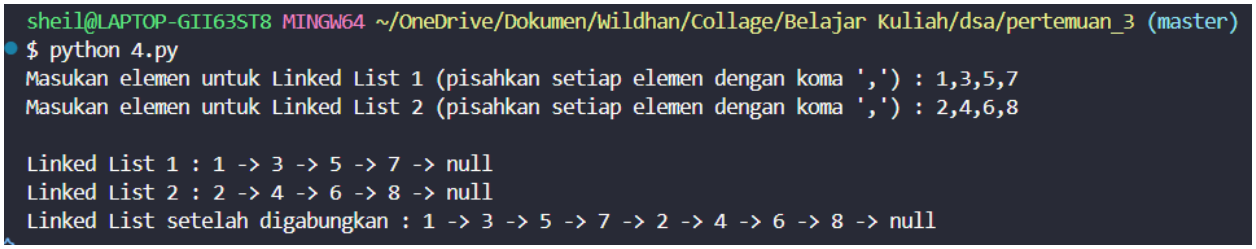
```
        # Insert values ke nodes
        if data_set is not None:
            for data_insert in data_set:
                new_node = Node(data_insert)
                if  not self.head:  # Jika linked list kosong
                    self.head = new_node
                    self.tail = new_node
                else:
                    self.tail.next  =   new_node # Sambungkan
tail ke node baru
                    self.tail   =   new_node # Update tail ke
node baru
        else:
            new_node = Node(data_set)
            if  not self.head:  # Jika linked list kosong
                self.head = new_node
                self.tail = new_node
            else:
                self.tail.next  =   new_node # Sambungkan tail ke
node baru
                self.tail   =   new_node # Update tail ke node
baru

    # Menampilkan nodes
    def display(self):
        result = ""
        temp = self.head
        while temp:
            result += str(temp.data) + " -> "
            temp = temp.next
        result += "null"

        # Jika kosong maka return string "kosong"
        if result == "null":
            return "kosong"
```

```
        return result

    # Fungsi untuk menyatukan
    def concat(self, other_list):
        # Jika list 1 kosong maka return list 2
        if not self.head:
            self.head = other_list.head
            self.tail = other_list.tail
            return self

        # Jika list 2 kosong maka tidak perlu apa-apa
        if not other_list.head:
            return self

        # Sambungkan tail list 1 ke head list 2
        self.tail.next = other_list.head
        self.tail = other_list.tail

        return self

# Latihan 3
ll1 = LinkedList()
ll2 = LinkedList()

# Insert node list 1
insert_values = input("Masukan elemen untuk Linked List 1
(pisahkan setiap elemen dengan koma ',') : ")
ll1.insert_node(insert_values)

# Insert node list 2
insert_values = input("Masukan elemen untuk Linked List 2
(pisahkan setiap elemen dengan koma ',') : ")
ll2.insert_node(insert_values)

# Menampilkan Hasil
result_list1 = ll1.display()
```

```
result_list2 = ll2.display()
result_concat = ll1.concat(ll2).display()

list1 = f"Linked List 1 : {result_list1}"
list2 = f"Linked List 2 : {result_list2}"
concat = f"Linked List setelah digabungkan : {result_concat}"
print(f"\n{list1}\n{list2}\n{concat}")
```

Berikut merupakan screenshot hasil latihan 4.

1. Contoh tampilan #1




2. Contoh tampilan #2