PRAKTEK 1

```
stack = []
```

Penjelasan:

Membuat stack kosong menggunakan list Python.

```
stack.append('A')
stack.append('B')
stack.append('C')
print ("Stack : ", stack)
```

Penjelasan:

Menambahkan tiga elemen ('A', 'B', 'C') ke stack. Dengan append, elemen ditambahkan ke akhir list (top of the stack). Hasil: ['A', 'B', 'C'].

```
if len(stack) != 0:
    print("Pop : ", stack.pop())
    print("Stack : ", stack)
else :
    print("Stack Kosong!")
```

Penjelasan:

Jika stack tidak kosong, hapus elemen terakhir (top stack, yaitu 'C') dan tampilkan stack setelah pop.

```
if not bool(stack):
    print("Stack Kosong!")
else:
    print("stack : ", stack)
    print("Top/Peek : ", stack[-1])
```

Penjelasan:

Cek apakah stack kosong. Jika tidak, tampilkan elemen paling atas (terakhir dalam list), yaitu 'B'.

```
if not bool(stack):
    print("Stack Kosong ")
else :
    print("Stack tidak kosong! ", stack)
```

Penjelasan:

Cek kembali apakah stack kosong menggunakan bool(stack).

```
print("Stack Size : ", len(stack))
```

Penjelasan:

Cetak jumlah elemen stack menggunakan len().

PRAKTEK 2

```
while True:
    isi_elemen = input("Masukkan isi elemen (ketik Selesai jika tidak): ")
    if isi_elemen.lower() == 'selesai':
        break
    stack = tumpukan.append(int(isi_elemen))
    print("tumpukan : ", tumpukan)
```

Penjelasan

Perulangan menerima input dari user. Jika user mengetik "selesai", maka berhenti. Jika tidak, input dikonversi menjadi integer dan ditambahkan ke tumpukan

```
    if len(tumpukan) != 0:
        hapus = input('Apakah ingin menghapus elemen [ya/tidak] : ')
        if hapus.lower() == 'tidak' :
            break
        print("Pop : ", tumpukan.pop())
        print("tumpukan : ", tumpukan)
    else :
        print("Stack Kosong!")
```

Penjelasan

Melakukan pop (hapus elemen paling atas) sebanyak jumlah elemen dalam tumpukan. Sebelum setiap pop, tanya user apakah ingin menghapus. Kalau tidak, perulangan berhenti.

PRAKTEK 3

```
def buat_node(data):
    return {'data': data, 'next': None}
```

Penjelasan

Fungsi membuat node stack dalam bentuk dictionary: data adalah nilai, next adalah referensi ke node berikutnya.

```
def push(head, data):
    new_node = buat_node(data)
    new_node['next'] = head
    return new_node
```

Penjelasan

Fungsi untuk push. Membuat node baru dan menghubungkannya ke head lama (node paling atas). Mengembalikan node baru sebagai head (top of the stack).

```
def pop(head):
    if head is None:
        print("Stack kosong.")
        return None, None
    popped = head['data']
    head = head['next']
    return head, popped
```

Penjelasan

Fungsi pop. Jika kosong, tampilkan pesan dan kembalikan None. Jika tidak, ambil data dari node atas dan kembalikan head baru (node berikutnya) dan data yang di-pop.

```
def peek(head):
    if head is None:
        return None
    return head['data']
```

Penjelasan

Mengembalikan data node paling atas tanpa menghapusnya

```
def is_empty(head):
    return head is None
```

Penjelasan

Mengembalikan True jika stack kosong.

```
stack = None  # Stack awal kosong

stack = push(stack, 10)
stack = push(stack, 20)
stack = push(stack, 30)
```

Penjelasan

Mendorong 3 elemen ke stack: 10, 20, 30. Sekarang top stack adalah 30.

```
print("Top stack:", peek(stack))  # Harusnya 30
```

Penjelasan

Menampilkan elemen paling atas (peek), yaitu 30.

```
stack, val = pop(stack)
print("Pop:", val)              # Harusnya 30

stack, val = pop(stack)
print("Pop:", val)              # Harusnya 20
```

Penjelasan

Menghapus dua elemen berturut-turut. Pertama 30, lalu 20.

```
print("Top stack sekarang:", peek(stack))  # Harusnya 10
```

Penjelasan

Sekarang elemen paling atas adalah 10.