Praktek 1

```python
# Membuat Stack
stack = []

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

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

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


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


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

- # Membuat Stack

  stack = []

  Baris ini mendeklarasikan sebuah list kosong bernama **stack**, yang akan digunakan untuk menyimpan elemen-elemen stack.

- # Push
  stack.append('A')
  stack.append('B')
  stack.append('C')
  print ("Stack : ", stack)

  Di sini, kita melakukan operasi "push" untuk menambahkan elemen ke dalam stack. Tiga elemen ('A', 'B', dan 'C') ditambahkan satu per satu menggunakan metode **append()**. Setelah itu, stack dicetak, dan outputnya akan menunjukkan isi stack saat ini, yaitu **['A', 'B', 'C']**.

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

  Pada bagian ini, kita melakukan operasi "pop" untuk menghapus elemen teratas dari stack. Sebelum melakukan pop, kita memeriksa apakah stack tidak kosong dengan memeriksa panjangnya (**len(stack) != 0**). Jika stack tidak kosong, elemen teratas dihapus dan dicetak, dan kemudian isi stack yang baru juga dicetak. Jika stack kosong, pesan "Stack Kosong!" akan ditampilkan.

- # top/peek
  if not bool(stack):
  print("Stack Kosong!")
  else:

  print("stack : ", stack)

  print("Top/Peek : ", stack[-1])

  Di sini, kita melakukan operasi "top" atau "peek" untuk melihat elemen teratas dari stack tanpa menghapusnya. Kita memeriksa apakah stack kosong. Jika kosong, kita mencetak "Stack Kosong!". Jika tidak, kita mencetak isi stack dan elemen teratas (elemen terakhir dalam list) menggunakan **stack[-1]**

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

Pada bagian ini, kita memeriksa apakah stack kosong dengan menggunakan **bool(stack)**. Jika stack kosong, kita mencetak "Stack Kosong". Jika tidak, kita mencetak "Stack tidak kosong!" beserta isi stack.

- # size
  print("Stack Size : ", len(stack))

  Di sini, kita mencetak ukuran stack dengan menggunakan **len(stack)**, yang memberikan jumlah elemen yang ada di dalam stack.

Output

```
Stack :  ['A', 'B', 'C']
Pop :  C
Stack :  ['A', 'B']
stack :  ['A', 'B']
Top/Peek :  B
Stack tidak kosong!  ['A', 'B']
Stack Size :  2
PS D:\python\tugas-PRE-PRAKTIKUM>
```

Praktek 2

```python
# Membuat program untuk menerima input dari pengguna untuk operasi push dan pop stack
# langkah 1 menyiapkan stack array
tumpukan = []

# langkah 2 push elemen ke stack
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)

# langkah 3 pop elemen dari stack
for i in range(len(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!")
```

- # Membuat program untuk menerima input dari pengguna untuk operasi push dan pop stack
  # langkah 1 menyiapkan stack array
  tumpukan = []

Baris ini mendeklarasikan list kosong bernama **tumpukan**, yang akan digunakan untuk menyimpan elemen-elemen stack berdasarkan input pengguna. Komentar di atas menjelaskan bahwa langkah pertama adalah menyiapkan array stack.

- # langkah 2 push elemen ke stack
  while True:
  isi_elemen = input("Masukkan isi elemen (ketik Selesai jika tidak): ")

  Di sini, kita memulai loop **while** yang akan terus berjalan hingga dihentikan secara eksplisit. Di dalam loop, program meminta pengguna untuk memasukkan elemen yang ingin ditambahkan ke stack. Jika pengguna mengetik "Selesai", loop akan berhenti.

- if isi_elemen.lower() == 'selesai':
  break

  Baris ini memeriksa apakah input dari pengguna (setelah diubah menjadi huruf kecil) adalah "selesai". Jika ya, loop akan dihentikan dengan perintah **break**.

- stack = tumpukan.append(int(isi_elemen))

  Di sini, program mencoba untuk mengonversi input pengguna menjadi integer dan menambahkannya ke dalam list **tumpukan** menggunakan metode **append()**. Namun, ada kesalahan di sini: **stack** tidak perlu dideklarasikan, karena **append()** mengembalikan **None**. Seharusnya hanya **tumpukan.append(int(isi_elemen))** tanpa penugasan ke **stack**.

- print("tumpukan : ", tumpukan)

  Setelah menambahkan elemen ke stack, program mencetak isi dari **tumpukan** untuk menunjukkan elemen-elemen yang telah ditambahkan.

- # langkah 3 pop elemen dari stack
  for i in range(len(tumpukan)):

  Di sini, kita memulai loop **for** yang akan berjalan sebanyak jumlah elemen dalam **tumpukan**. Loop ini digunakan untuk melakukan operasi pop pada elemen stack.

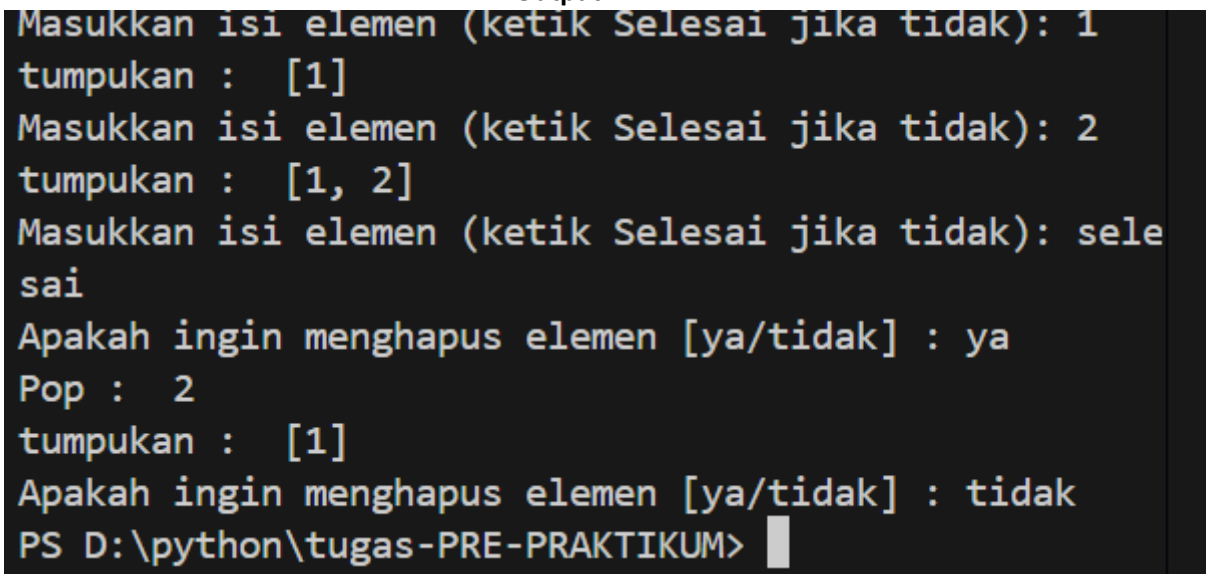
- if len(tumpukan) != 0:

  Baris ini memeriksa apakah stack tidak kosong sebelum mencoba untuk menghapus elemen. Jika stack kosong, kita tidak dapat melakukan operasi pop.

- hapus = input('Apakah ingin menghapus elemen [ya/tidak] : ')

  Program meminta pengguna untuk mengonfirmasi apakah mereka ingin menghapus elemen dari stack. Pengguna dapat menjawab "ya" atau "tidak".

Output



Praktet 3

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

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

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

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

def is_empty(head):
    return head is None

# Uji coba
stack = None  # Stack awal kosong

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

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

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

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

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

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

  Fungsi `buat_node` ini digunakan untuk membuat sebuah node baru dalam stack. Node ini adalah sebuah dictionary yang memiliki dua kunci: `data`, yang menyimpan nilai yang diberikan, dan `next`, yang akan menunjuk ke node berikutnya (diinisialisasi dengan `None`).

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

```

  Fungsi `push` digunakan untuk menambahkan elemen baru ke atas stack. Pertama, kita membuat node baru dengan memanggil `buat_node(data)`. Kemudian, kita mengatur `next` dari node baru untuk menunjuk ke node yang saat ini menjadi `head` (elemen teratas stack). Akhirnya, kita mengembalikan node baru sebagai `head` yang baru, sehingga node baru menjadi elemen teratas stack.

- def pop(head):
  if head is None:
  print("Stack kosong.")
  return None, None

  Fungsi `pop` digunakan untuk menghapus elemen teratas dari stack. Pertama, kita memeriksa apakah stack kosong dengan memeriksa apakah `head` adalah `None`. Jika kosong, kita mencetak pesan "Stack kosong." dan mengembalikan `None` untuk `head` dan `None` untuk nilai yang dipop.

- popped = head['data']
- head = head['next']
- return head, popped

```

  Jika stack tidak kosong, kita menyimpan nilai dari elemen teratas (`head['data']`) ke dalam variabel `popped`. Kemudian, kita memperbarui `head` untuk menunjuk ke node berikutnya (`head['next']`). Akhirnya, kita mengembalikan `head` yang baru dan nilai yang dipop.

- def peek(head):
- if head is None:

- return None
- return head['data']

```

Fungsi `peek` digunakan untuk melihat nilai elemen teratas dari stack tanpa menghapusnya. Jika stack kosong, fungsi ini mengembalikan `None`. Jika tidak kosong, fungsi ini mengembalikan nilai dari elemen teratas (`head['data']`).

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

```

Fungsi `is_empty` digunakan untuk memeriksa apakah stack kosong. Fungsi ini mengembalikan `True` jika `head` adalah `None`, dan `False` jika tidak.

- # Uji coba
- stack = None # Stack awal kosong

```

- Di sini, kita mendeklarasikan variabel `stack` dan menginisialisasinya dengan `None`, yang menunjukkan bahwa stack awalnya kosong.

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

```

Kita melakukan beberapa operasi `push` untuk menambahkan elemen ke stack. Pertama, kita menambahkan `10`, kemudian `20`, dan terakhir `30`. Setiap kali kita memanggil `push`, kita memperbarui `stack` dengan `head` yang baru yang dikembalikan oleh fungsi `push`.

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

```

- Di sini, kita memanggil fungsi `peek` untuk melihat nilai elemen teratas dari stack. Karena kita telah menambahkan `30` terakhir, output yang diharapkan adalah `30`.

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

```

- Kita memanggil fungsi `pop` untuk menghapus elemen teratas dari stack. Nilai yang dipop disimpan dalam variabel `val`, dan kita mencetak nilai tersebut. Output yang diharapkan adalah `30`, karena itu adalah elemen teratas yang dihapus.

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

```

- Kita memanggil fungsi `pop` lagi untuk menghapus elemen teratas yang sekarang adalah `20`. Nilai yang dipop disimpan dalam `val`, dan kita mencetaknya. Output yang diharapkan adalah `20`.

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

```

- Terakhir, kita memanggil fungsi `peek` lagi untuk melihat nilai elemen teratas dari stack setelah dua operasi pop. Karena kita telah menghapus `30` dan `20`, elemen teratas yang tersisa adalah `10`, sehingga output yang diharapkan adalah `10`.

Output