MODUL PERKULIAHAN

TFC251 – Praktikum Struktur Data

Stack Dalam Python

Penyusun Modul	:	Suamanda Ika Novichasari, M.Kom
Minggu/Pertemuan	:	3
Sub-CPMK/Tujuan Pembelajaran	:	1. Mahasiswa mampu menerapkan konsep stack pada bahasa pemrograman python
Pokok Bahasan	:	1. Stack

Program Studi Teknologi Informasi (S1)
Fakultas Teknik
Universitas Tidar
Tahun 2024

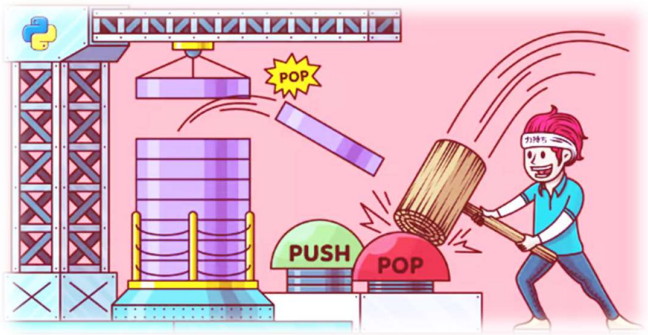
Materi 3
STACK DALAM PYTHON

Petunjuk Praktikum :

- Cobalah semua contoh kode program yang terdapat pada semua sub bab dalam modul ini menggunakan google colab.
- Dokumentasikan kegiatan dalam bentuk laporan praktikum sesuai template yang telah ditentukan.
- Setelah selesai mempraktekan semua materi, silakan kerjakan tugas untuk persiapan praktikum pertemuan selanjutnya.

Stack adalah jenis struktur data linier yang dapat dibandingkan dengan tumpukan objek dalam kehidupan nyata. Dalam stack, penambahan dan penghapusan elemen hanya bisa terjadi pada bagian paling atas, atau dengan kata lain, mengikuti prinsip LIFO (*Last in, First out*) atau FILO (*First In Last Out*). Stack memiliki dua operasi utama, yaitu:

a. Push(item), untuk tambah item pada elemen teratas.
b. Pop(item), untuk hapus elemen teratas pada stack.
c. empty(), untuk mengetahui apakah tumpukan kosong
d. size() untuk mengetahui ukuran tumpukan
e. top() / peek(), merujuk pada ke elemen paling atas tumpukan



Kelebihan	Kekurangan
• Merupakan struktur data sederhana yang mudah dipahami dan digunakan. • Efisien pada saat menambah dan menghapus elemen dengan kompleksitas waktu O(1). • Dapat digunakan untuk membalikkan urutan elemen. • Dapat digunakan untuk mengimplementasikan fungsi undo/redo dalam aplikasi.	• Ukuran terbatas, jika penuh tidak dapat menambahkan elemen lain. • Tidak menyediakan akses cepat ke elemen kecuali elemen paling atas. • Tidak efisien jika digunakan untuk pencarian karena harus mengeluarkan elemen satu persatu.

Seperti kata pepatah banyak jalan menuju roma, begitu juga dengan stack. Stack dapat diimplementasikan dengan 4 cara, yaitu :

a. List
b. Collections.deque
c. queue.LifoQueue
d. Single Linked List

3.1 Stack dengan List

Struktur data bawaan Python, seperti list, dapat berfungsi sebagai tumpukan dengan menggunakan metode append() untuk menambahkan elemen ke tumpukan dan pop() untuk menghapus elemen dalam urutan LIFO. Namun, terdapat beberapa kelemahan pada penggunaan list. Salah satu masalah utamanya adalah bahwa performanya dapat terpengaruh saat tumpukan menjadi besar. Karena item dalam list disimpan berdampingan dalam memori, jika tumpukan melebihi blok memori yang saat ini dialokasikan untuknya, Python harus melakukan alokasi memori tambahan. Hal ini dapat menyebabkan penambahan elemen menggunakan append() memakan waktu lebih lama pada beberapa kasus.

Stack dengan List

```python
# Python program to demonstrate stack implementation using list
stack = []

# append() function to push element in the stack
stack.append('a')
stack.append('i')
stack.append('u')
stack.append('e')
stack.append('o')

print('Initial stack')
print(stack)

# pop() function to pop element from stack in LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)

# uncommenting print(stack.pop()) will cause an IndexError as the stack is now empty
```

```
Initial stack
['a', 'i', 'u', 'e', 'o']

Elements popped from stack:
o
e
u

Stack after elements are popped:
['a', 'i']
```

3.2 Stack dengan Collections.deque

Stack dalam Python bisa direalisasikan menggunakan kelas deque dari modul collections. Deque lebih disukai ketimbang list dalam situasi di mana kita memerlukan operasi append dan pop yang lebih cepat di kedua ujung kontainer, karena deque menawarkan kompleksitas waktu O(1) untuk operasi tersebut, berbeda dengan list yang kompleksitasnya adalah O(n).

Metode yang sama seperti pada list juga diterapkan pada deque, yaitu append() dan pop(). Sedangkan perbedaannya terletak pada pendeklarasian variabel stack, perhatikan pada kotak merah dibawah ini.

Stack dengan collections.deque

```
# Python program to demonstrate stack implementation using collections.deque
from collections import deque

stack = deque()

# append() function to push element in the stack
stack.append('a')
stack.append('i')
stack.append('u')
stack.append('e')
stack.append('o')

print('Initial stack:')
print(stack)

# pop() function to pop element from stack in LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)

# uncommenting print(stack.pop()) will cause an IndexError as the stack is now empty
```

```
Initial stack:
deque(['a', 'i', 'u', 'e', 'o'])

Elements popped from stack:
o
e
u

Stack after elements are popped:
deque(['a', 'i'])
```

3.3 Stack dengan queue.LifoQueue

Modul Antrian juga menyediakan Antrian LIFO (Last In, First Out), yang pada dasarnya berfungsi sebagai Tumpukan. Data dimasukkan ke dalam Antrian menggunakan fungsi put(), dan get() mengambil data dari Antrian.

Beberapa fungsi tersedia dalam modul ini diantaranya sebagai berikut :

a. maxsize: Menunjukkan jumlah maksimum item yang diizinkan dalam antrian.
b. empty(): Mengembalikan True jika antrian kosong, dan False sebaliknya.

c. full(): Mengembalikan True jika antrian berisi jumlah maksimum item yang ditentukan oleh maxsize. Jika maxsize disetel ke 0 (nilai default), full() tidak pernah mengembalikan True.
d. get(): Menghapus dan mengembalikan sebuah item dari antrian. Jika antrian kosong, ia menunggu hingga sebuah item tersedia.
e. get_nowait(): Mengembalikan sebuah item jika tersedia secara langsung, jika tidak, menaikkan QueueEmpty.
f. put(item): Menambahkan sebuah item ke dalam antrian. Jika antrian penuh, ia menunggu hingga sebuah slot tersedia sebelum menambahkan item tersebut.
g. put_nowait(item): Menambahkan sebuah item ke dalam antrian tanpa menunggu. Jika tidak ada slot kosong yang tersedia secara langsung, ia menaikkan QueueFull.
h. qsize(): Mengembalikan jumlah item yang saat ini ada dalam antrian

Stack dengan queue.LifoQueue

```
[10] # Python program to demonstrate stack implementation using queue module
     from queue import LifoQueue

     # Initializing a stack
     stack = LifoQueue(maxsize=5)

     # qsize() show the number of elements in the stack
     print(stack.qsize())

     # put() function to push element in the stack
     stack.put('a')
     stack.put('i')
     stack.put('u')
     stack.put('e')
     stack.put('o')

     print("Full: ", stack.full())
     print("Size: ", stack.qsize())

     # get() function to pop element from stack in LIFO order
     print('\nElements popped from the stack')
     print(stack.get())
     print(stack.get())
     print(stack.get())

     print("\nEmpty: ", stack.empty())
```

```
0
Full:  True
Size:  5

Elements popped from the stack
o
e
u

Empty:  False
```

3.4 Stack dengan Single Linked List

Terdapat dua metode dalam linked list, yaitu addHead(item) dan removeHead(), yang beroperasi dalam waktu tetap. Kedua metode tersebut sesuai untuk mengimplementasikan tumpukan.

Berikut adalah beberapa fungsi yang tersedia:

a. getSize() – Mendapatkan jumlah item dalam tumpukan.
b. isEmpty() – Mengembalikan True jika tumpukan kosong, dan False jika tidak.
c. peek() – Mengembalikan item teratas dalam tumpukan. Jika tumpukan kosong, akan muncul pengecualian.
d. push(value) – Menambahkan sebuah nilai ke bagian kepala tumpukan.
e. pop() – Menghapus dan mengembalikan sebuah nilai dari bagian kepala tumpukan. Jika tumpukan kosong, akan muncul pengecualian.

Stack dengan Single Linked List

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


class Stack:

    # Initializing a stack, Use a dummy node, which is easier for handling edge cases.
    def __init__(self):
        self.head = Node("head")
        self.size = 0

    # String representation of the stack
    def __str__(self):
        cur = self.head.next
        out = ""
        while cur:
            out += str(cur.value) + "->"
            cur = cur.next
        return out[:-2]

    # Get the current size of the stack
    def getSize(self):
        return self.size

    # Check if the stack is empty
    def isEmpty(self):
        return self.size == 0

    # Get the top item of the stack
    def peek(self):
        # Sanitary check to see if we are peeking an empty stack.
        if self.isEmpty():
            raise Exception("Peeking from an empty stack")
        return self.head.next.value

    # Push a value into the stack.
    def push(self, value):
        node = Node(value)
        node.next = self.head.next
        self.head.next = node
        self.size += 1

    # Remove a value from the stack and return.
    def pop(self):
        if self.isEmpty():
            raise Exception("Popping from an empty stack")
        remove = self.head.next
        self.head.next = self.head.next.next
        self.size -= 1
        return remove.value


# Driver Code
if __name__ == "__main__":
    stack = Stack()
    for i in range(1, 11):
        stack.push(i)
    print(f"Stack: {stack}")

    for _ in range(1, 6):
        remove = stack.pop()
        print(f"Pop: {remove}")
    print(f"Stack: {stack}")
```

```
Stack: 10->9->8->7->6->5->4->3->2->1
Pop: 10
Pop: 9
Pop: 8
Pop: 7
Pop: 6
Stack: 5->4->3->2->1
```

TUGAS 3

Rangkumlah tentang penerapan queue dalam python !

Tugas dikumpulkan pada elita maksimal 1 jam sebelum pertemuan selanjutnya.