

P05 : Stack(lanj.)

Tools: Python dan Visual Studio Code

Pengumpulan Tugas
Simpan file Tugas 1 dengan nama M5-tugas1-stack.py
Simpan file Tugas2 dengan nama M5-tugas2-stack.py
Simpan file laporan dengan nama M5-tugas-stack.pdf
Kumpulkan program File Tugas 1 dan Tugas 2 beserta laporan!
Kumpulkan file-file diatas dalam satu file zip dengan format: NIM_minggu5.zip

Tugas 1: Simpan file dengan nama **M5-tugas1-stack.py**

Susun program untuk menginput sebuah ekspresi aritmetika (tidak lebih dari 20 karakter). Kemudian periksa kurung buka dan kurung tutupnya apakah tepat saling berpasangan. Tampilkan perkataan “BENAR” jika tepat saling berpasangan, atau tampilkan perkataan “SALAH” bila ada yang tidak tepat saling berpasangan.

Contoh:

Input	Tampilan
$A + B * C - D / E$	BENAR
$A + B * (C - D) / E$	BENAR
$(A + B) * (C - D) / E$	BENAR
$(A + B * (C - D) / E)$	BENAR
$(A + B * (C - D) / E$	SALAH
$(A + B) * C - D) / E$	SALAH
$)A + B * (C - D / E$	SALAH
$(A + B (* (C - D) / E$	SALAH
$(A + B) *)C - D) / E$	SALAH

Petunjuk:

Telusuri karakter per karakter isi ekspresi aritmetika sampai ditemui karakter NULL ('\0').

- Jika terbaca karakter kurung buka '(' maka PUSH kurung buka tersebut ke dalam stack.
- Jika terbaca karakter kurung tutup ')' maka periksa isi stack. Jika stack ada isinya, maka keluarkan (POP) satu isi stack, yaitu sebuah kurung buka '('. Tetapi jika isi stack KOSONG berarti kurung buka dan kurung tutup tidak berpasangan, maka tampilkan "SALAH" dan proses selesai.
- Setelah semua karakter telah ditelusuri, sampai ditemui karakter NULL, maka periksa isi stack. Jika isi stack KOSONG, berarti kurung buka dan kurung tutup telah tepat saling berpasangan, tampilkan "BENAR" dan proses selesai. Jika isi stack masih ada, berarti kurung buka lebih banyak daripada kurung tutup, tampilkan "SALAH" dan proses selesai.

Salin dan jelaskan kode program Anda di sini.

```
class Stack:
    def __init__(self):
        # Inisialisasi stack dengan list kosong
        self.items = []

    def push(self, item):
        # Menambahkan item ke dalam stack
        self.items.append(item)

    def pop(self):
        # Menghapus dan mengembalikan item teratas dari stack, jika tidak kosong
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def is_empty(self):
        # Memeriksa apakah stack kosong atau tidak
        return len(self.items) == 0

def check_parentheses(expression):
    stack = Stack() # Membuat objek stack untuk melacak tanda kurung

    for char in expression:
        if char == '(':
            stack.push(char) # Memasukkan tanda kurung buka ke dalam stack
        elif char == ')':
            if stack.is_empty():
                return False
            else:
                stack.pop()
    # Memeriksa apakah stack kosong setelah semua tanda kurung diproses
    return stack.is_empty()

def main():
    expressions = [
        "A + B * C - D / E",
        "A + B * (C - D) / E",
        "(A + B) * (C - D) / E",
        "(A + B * (C - D) / E)",
        "(A + B * (C - D) / E",
        "(A + B) * C - D) / E",
        ")A + B * (C - D / E",
        "(A + B ( * (C - D) / E",
        "(A + B) * )C - D) / E"
    ]

    for expression in expressions:
        print(expression)
        if check_parentheses(expression):
            print("BENAR") # Jika tanda kurung seimbang, cetak "BENAR"
        else:
            print("SALAH") # Jika tanda kurung tidak seimbang, cetak "SALAH"

if __name__ == "__main__":
    main()
```

Tampilkan hasil running dan penjelasannya di sini.

```

C:\Users\dyy.react\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\dyy.react\Documents\Aldy Pro\Education\Poltek - D4 - RPL\TIn
A + B * C - D / E
BENAR
A + B * (C - D) / E
BENAR
(A + B) * (C - D) / E
BENAR
(A + B * (C - D) / E)
BENAR
(A + B * (C - D) / E
SALAH
(A + B) * C - D / E
SALAH
)A + B * (C - D / E
SALAH
(A + B ( * (C - D) / E
SALAH
(A + B) * )C - D) / E
SALAH
Process finished with exit code 0

```

Tugas 2: Simpan file dengan nama **M5-tugas2-stack.py**

Modifikasi program stack sebelumnya sehingga memiliki kemampuan untuk memeriksa kelengkapan pasangan kurung buka ‘(’, ‘[’, ‘{’ dan kurung tutup ‘)’, ‘]’, ‘}’.

Contoh:

Input	Tampilan
$A + [B * (C - D) - E] / F$	BENAR
$[A + B * (C - D)] - E / F$	BENAR
$[A + B * (C - D)] - E / F$	SALAH
$\{A + B * (C - D)\} - E / F$	SALAH

Salin dan jelaskan kode program Anda di sini.

```
class Stack:
    def __init__(self):
        # Inisialisasi stack dengan list kosong
        self.items = []

    def push(self, item):
        # Menambahkan item ke dalam stack
        self.items.append(item)

    def pop(self):
        # Menghapus dan mengembalikan item teratas dari stack, jika tidak kosong
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def is_empty(self):
        # Memeriksa apakah stack kosong atau tidak
        return len(self.items) == 0

    def peek(self):
        # Melihat item teratas dari stack tanpa menghapusnya
        if not self.is_empty():
            return self.items[-1]
        else:
            return None

def check_parentheses(expression):
    stack = Stack() # Membuat objek stack untuk melacak tanda kurung
    opening_brackets = "([{" # Tanda kurung buka
    closing_brackets = ")]}" # Tanda kurung tutup

    for char in expression:
        if char in opening_brackets:
            stack.push(char) # Memasukkan tanda kurung buka ke dalam stack
        elif char in closing_brackets:
            if stack.is_empty():
                return False # Tanda kurung tutup tidak memiliki pasangan buka
            else:
                top = stack.pop() # Mengambil tanda kurung buka teratas dari stack
                # Memeriksa apakah tanda kurung buka memiliki pasangan yang sesuai
                if opening_brackets.index(top) != closing_brackets.index(char):
                    return False

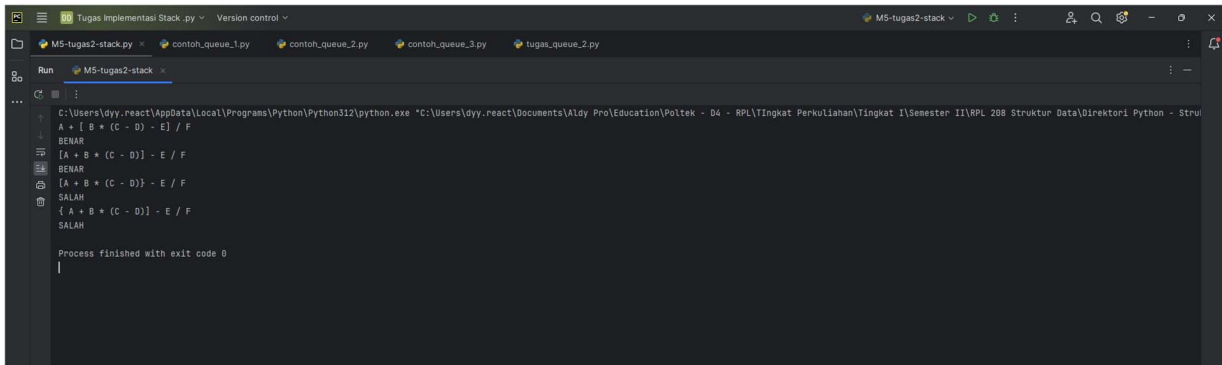
    # Memeriksa apakah stack kosong setelah semua tanda kurung diproses
    return stack.is_empty()

def main():
    expressions = [
        "A + [ B * (C - D) - E] / F",
        "[A + B * (C - D)] - E / F",
        "[A + B * (C - D)] - E / F",
        "{ A + B * (C - D) ] - E / F"
    ]

    for expression in expressions:
        print(expression)
        if check_parentheses(expression):
            print("BENAR") # Jika tanda kurung seimbang, cetak "BENAR"
        else:
            print("SALAH") # Jika tanda kurung tidak seimbang, cetak "SALAH"

if __name__ == "__main__":
    main() # Memanggil fungsi main jika kode dieksekusi sebagai program utama
```

Tampilkan hasil running dan penjelasannya di sini.



The screenshot shows a Python IDE with a file explorer on the left containing files: `MS-tugas2-stack.py`, `contoh_queue_1.py`, `contoh_queue_2.py`, `contoh_queue_3.py`, and `tugas_queue_2.py`. The main editor displays the code for `MS-tugas2-stack`. The code defines a stack class with methods `push`, `pop`, `peek`, `is_empty`, and `size`. It then tests the stack with a sequence of operations: push 10, push 20, push 30, push 40, push 50, pop, pop, pop, pop, pop, and peek. The output shows the stack is empty after five pops, and the final peek returns 10. The process finished with exit code 0.

```
C:\Users\dyv.react\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\dyv.react\Documents\Aldy Pro\Education\Poltek - D4 - RPL\Tingkat Perkuliahan\Tingkat II\Semester II\RPL 208 Struktun Data\Direktori Python - Stru
A + [ B * (C - D) - E ] / F
BENAR
[A + B * (C - D)] - E / F
BENAR
[A + B * (C - D)] - E / F
SALAH
{ A + B * (C - D) } - E / F
SALAH

Process finished with exit code 0
|
```