Stack(tumpukan) Stacks adalah satu struktur data dimana penambahan dan penghapusan data, hanya dapat dilakukan pada satu ujung yang sama atau yang biasa dilakukan dengan istilah top.

1.1 Operasi pada stack

1. stack() - inisialiasai stack yang kosong
2. push(data) - penambahan data baru pada posisi top dari stack
3. pop() - penhapusan data yang terdapat di posisi top dari stack
4. peek() - informasi data yang terletak pada posisi top
5. isEmpty() - untuk memeriksa apakah stack dalam keadaan kosong
6. size() - informasi jumlah data yang tedapat pada stack

In [40]:
```
def stack():
    s = []
def push(s, data):
    s.append(data)
def pop(s):
    return s.pop()
def peek(s):
    return s[len(s)-1]
def isEmpty(s):
    return s==[]
def size(s):
    return len(s)
```

In [41]:
```
st = stack()
isEmpty(st)
```

Out[41]: True

In [42]:
```
push(st,1)
push(st,2)
push(st,3)
pop(st)
push(st,4)
pop(st)
print(st)
```

```
[1, 2]
```

latihan - 1

buatlah fucntion untuk 'reverse word' dengan menggunakan konsep stacks, misalkan kata 'faqih' menjadi 'hiqfa'!

In [25]:
```python
def stack():
  return []

# Operasi push (menambahkan elemen ke dalam stack)


def stack_push(stack, element):
  stack.append(element)

# Operasi pop (mengambil elemen terakhir yang dimasukkan ke dalam stack)


def stack_pop(stack):
  if not stack:
    return None
  else:
    return stack.pop()

# Operasi peek (mengambil elemen teratas pada stack)


def stack_peek(stack):
  if not stack:
    return None
  else:
    return stack[-1]

# Operasi is_empty (mengecek apakah stack kosong atau tidak)


def stack_is_empty(stack):
  return len(stack) == 0
```

In [26]:
```python
def reverse_word(string):
  # Inisialisasi stack dengan fungsi stack()
  s = stack()

  # Menambahkan setiap karakter ke dalam stack
  for char in string:
    stack_push(s, char)

  # Mengambil setiap karakter dari stack untuk membentuk string terbalik
  reversed_string = ""
  while not stack_is_empty(s):
    char = stack_pop(s)
    reversed_string += char

  return reversed_string


print(reverse_word('faqih'))
```

```
hiqaf
```

1.2 Delimiter matching

mengecek apakah kurung buka dan kurung tutup '(){}[]' itu sesuai dengan persamaan matematika menggunakan fungsi paranthesesCheck()

- parantheses sering digunkana untuk urutan penyelesain dalam persamaan matematika seperti x = 5 x (4 + 5) / ((3 + 2) x (10 - 8))

In [ ]:
```
def stack():
    s = []
    return s
def push(s, data):
    s.append(data)
def pop(s):
    return s.pop()
def peek(s):
    return s[len(s)-1]
def isEmpty(s):
    return s == []
def sixe(s):
    return len(s)

# function
def paranthesesCheck(strMath):
    operandStack = stack()
    lenMath = len(strMath)
    openOperand = '({['
    closeOperand = ')}]'
    print(f'len math = {lenMath}')

    i = 0
    matched = True
    while i < lenMath:
        # print(f'{i} = {strMath[i]}')
        if strMath[i] in openOperand:
            push(operandStack, strMath[i])
            # print(operandStack)
        elif strMath[i] in closeOperand:
            if not isEmpty(operandStack):
                top = pop(operandStack)
                # print(f'top = {top}')
                # print(operandStack)
                if openOperand.index(top)==closeOperand.index(strMath[i]):
                    matched = matched and True
                else:
                    matched = matched and False
                    print('kurung buka dan kurung tutup tidak benar atau tidak
cocok ')
            else:
                matched = matched and False
                print('jumlah kurung tutup lebih banyak')
        i += i
        # print(matched)

    if not isEmpty(operandStack):
        matched = False
        print('jumlah kurung buka lebih banyak')
    return matched

persamaan_1 = '5 x 10 + (4 + 5) / ((3 + 2) x (10 = 9 ))'
persamaan_2 = '5 x 10 + (4 + 5) / ((3 + 2) x (10 = 9 )))'
persamaan_3 = '5 x 10 + (4 + 5) / ((3 + 2) x (10 = 9 )'
persamaan_4 = '5 x 10 + (4 + 5] / ((3 + 2) x (10 = 9 ))'
```

```
print(paranthesesCheck(persamaan_1))
print(paranthesesCheck(persamaan_2))
print(paranthesesCheck(persamaan_3))
print(paranthesesCheck(persamaan_4))
```