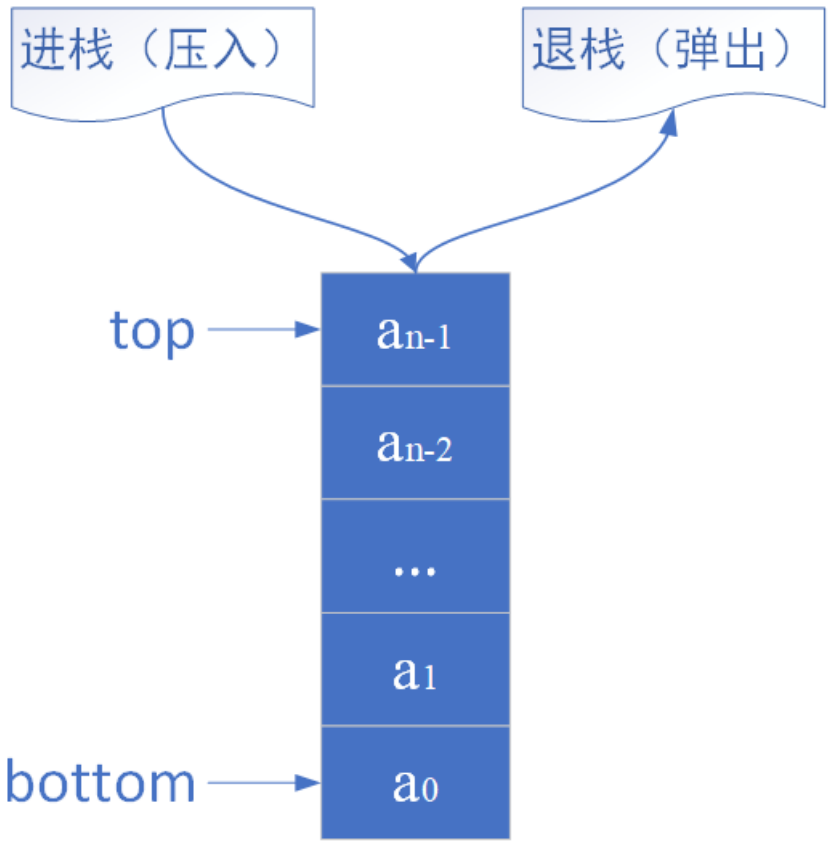
# python 数据结构之栈

## 一、栈简介

栈是一种特殊的线性表，具有后进先出（LIFO, Last In First Out）的特点，并且的栈的所有操作只能在某一端进行，能进行操作的一段的第一个元素称为栈顶（top），另一端的第一个元素成为栈底。不含任何元素的栈称为空栈，栈又称为后进先出的线性表。




顺序栈和顺序表数据成员相同，不同之处在于顺序栈的入栈和出栈操作只允许对当前栈顶进行操作

## 二、栈的实现

栈的五种基本方法：

push(item)：元素进栈。

pop()：元素出栈，删除栈顶元素，并返回该元素。

is_empty()：判断栈是否为空；栈为空，返回True；栈非空，返回False。

peek(item)：访问栈顶元素。

length()：返回栈中元素的个数。

栈的实现可以使用列表或者单链表实现

**基于列表实现栈**

```
'''列表实现栈'''
class Stack(object):
    def __init__(self):
        # 列表实现栈
```

```
        self.stack = []

    def length(self):
        '''获取栈的元素个数'''
        return len(self.stack)

    def is_empty(self):
        '''判断栈是否为空；栈为空，返回True；栈不为空，返回False'''
        return self.length() == 0

    def push(self, data):
        '''元素入栈，在栈顶加入元素'''
        self.stack.append(data)

    def pop(self):
        '''元素出栈，删除栈顶元素，并返回该元素'''
        if not self.is_empty():
            return self.stack.pop()
        else:
            print('StackError: Fail to pop, the stack is empty.')

    def peek(self):
        '''访问栈顶元素'''
        if not self.is_empty():
            return self.stack[-1]
        else:
            print('The stack is empty.')
```

示例：操作栈

```
def main():
    stack = Stack()
    print(stack.is_empty())
    for i in range(5):
        stack.push(i)

    print(stack.is_empty())
    print(stack.peek())
    while stack.length() > 0:
        ele = stack.pop()
        print(ele, end = '\t')
    print('\n')
    print(stack.is_empty())

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

**基于链表实现栈**

```
'''基于链表实现栈类'''
class LNode(object):
    def __init__(self, item, next_ = None):
        self.item = item
        self.next = next_

class LStack(object):
```

```
    def __init__(self):
        self._top = None
    def is_empty(self):
        return self._top is None

    def length(self):
        count = 0
        cur = self._top
        while cur is not None:
            count += 1
            cur = cur.next
        return count

    def push(self, item):
        self._top = LNode(item, self._top)

    def pop(self):
        if self._top is None:
            print('The stack is empty.')
            return
        elem = self._top.item
        self._top = self._top.next
        return elem

    def peek(self):
        '''返回栈顶元素'''
        if self._top is None:
            print('The stack is empty.')
            return
        return self._top.item
```

示例：操作栈

```
def main():
       lstack = LStack()
    print(lstack.is_empty())
    for i in range(5,10):
        lstack.push(i)
    print(lstack.is_empty())
    print("stack length: ", lstack.length())
    while lstack.length() > 0:
        print(lstack.pop(), end='\t')
    print('\n')
    print("stack length: ", lstack.length())
    print(lstack.is_empty())

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