Stack and Queue

- We use stack and queue ADTs to determine the visiting order of items in the structure.
  - We can only append (**push**) items to the tail of a stack (or top of a stack) and only remove (**pop**) item from the tail of a stack (or top of a stack). We say stack is a "last-in, first-out" (LIFO) structure.
  - We can only append (**enqueue**) items to the tail of a queue and only remove (**dequeue**) item from the head of a queue. We say queue is a "first-in, first-out" (FIFO) structure.

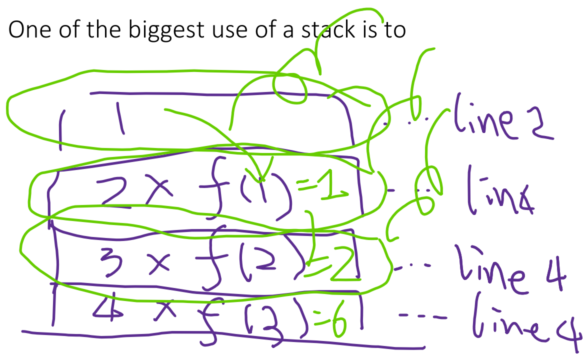
Stack

- As an ADT, a stack has at least the following methods:
  - Push(item)
  - Pop ()
  - Peek (): return the current top item without popping it out.

- Other than determining the order, stacks are also used in a lot of cases. One of the biggest use of a stack is to handle recursions.

```
                 4
def factorial (n: int):
1   if n <= 1:
2       return 1
3   else:
4      return n * factorial(n-1)
```

$n! = n \times (n-1)!$



1. Use a stack to eliminate the recursion in the method factorial.

2. Given an expression that contains multiple "(" and ")", determine whether the parentheses are valid. For example, "$(1 * (2 + 1)) * (2 + 3)$" is valid, but "$(1 * (2 + (3 - 2)) * (4$" is not valid.
   - For a left parenthesis, there must be a right parenthesis comes later. From this observation, here is our algorithm:
     - Whenever we see a left parenthesis, push it to a stack.
     - Whenever we see a right parenthesis, we try to pop out a left parenthesis.
     - If we successfully paired up all left and right parentheses, it is valid.
   - This example shows that stack can be used to pair things up.