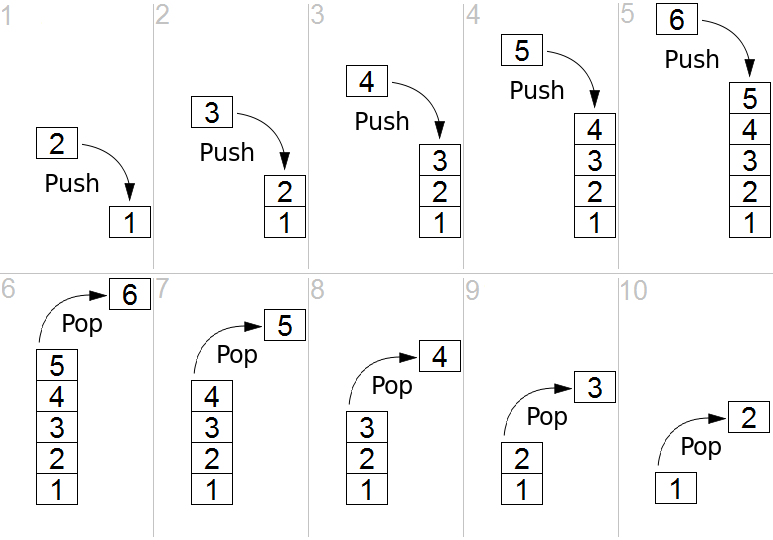
**Stacks**

A stack is a simple data structure best thought of as a pile of papers or books. It is sometimes described as a Last In First Out or LIFO structure. It comes equipped with only two operations:

push(object) will put the given data object on top of the current stack of objects.

pop() will remove the top object from the stack and return it for use in the main program.

A simple pictorial representation is



A stack can be made in Python using a list and the list.append and list.pop operations.

A more efficient version can be constructed as a stack class using a linked structure similar to the linked list object. We will look at this idea in more detail, together with the queue data structure, later.

Stacks are commonly used in programs that evaluate expressions. One example is to evaluate a mathematical calculation involving many brackets. You can traverse the expression, pushing each open bracket onto the stack. When you encounter a close bracket you know that it's time to work out the part between the most recent open bracket and the close bracket so you can pop that part of the expression off the stack and work it out.

Another example is in processing or *parsing* a markup language such as html to display in a browser window. Every time you encounter a tag (e.g. <list>) you can push it onto the stack and change the processing to that required for the new environment. When you encounter the end tag (in this case </list>) you can pop the stack and return to the previous state.

**Exercises:**

1. A stack called s is initially empty. Draw pictures of the stack as a tower with the most recent item added on the top after each of the following sequences of operations are applied. (One picture for part a), one for b) etc.)
   1. s.push("Fred"); s.push("Brian"); s.push("Sue"); s.pop(); s.push("Brian")
   2. (continued on from part a)) s.push("Sue"); s.push("Sally"); s.pop(); s.push("Sally")
   3. (continued from b)) s.pop(); s.pop(); s.pop()
2. A stack called s initially contains "a", "b" and "c", with "a" on the bottom and "c" on the top. If we apply print(s.pop()) the computer will first remove the top item from the stack and then print out the resulting item, in this case it will print c and the stack will only have two items left. What is the output of each of the following sequences of operations?
   1. print(s.pop()); s.pop(); print(s.pop())
   2. (using the original 3 item stack) s.pop(); print(s.pop()); print(s.pop())
   3. (using the original stack) s.pop(); s.pop(); print(s.pop())

**(turn over)**

1. A program uses two stacks, s1 contains "a", "b", "c" as in question 2, s2 is initially empty.
   1. What is the output of the following sequence of operations?

s2.push(s1.pop()); print(s1.pop()); print(s2.pop()); print(s1.pop())

* 1. Using the same two stacks as at the start, and a combination of the four operations s1.push(s2.pop()), s2.push(s1.pop()), print(s1.pop()), print(s2.pop()) **and no others**, can you write instructions to print out a then b then c?
  2. Parts a) and b) have printed out two of the six arrangements of the three letters abc. Write sets of instructions to print out as many of the other arrangements as you can, each time starting with the original two stacks. Are any arrangements impossible?
  3. Suppose s1 contains the letters of the word stack in order with s on the bottom and s2 is empty. There are 120 different arrangements of these letters. Is it possible to print out **any** given arrangement or are some impossible? You should describe an algorithm that would take an arrangement as an input and use the two stacks to print it out, or you should give an arrangement that can't be printed and a convincing reason why not. ("I don't know how" is not very convincing!)

1. A common programming project is to implement the "Towers of Hanoi" puzzle using stacks. There's a nice version in Python, using PyGame for the graphics, at the website <http://www.tuxradar.com/content/code-project-tower-hanoi-python>

Notice that, because the stacks are only small they have been created using lists and list.append rather than creating a special linked object with nodes.

1. Another common use of stacks is in generating and traversing mazes. Look up and read the articles on Wikipedia about the *Maze Generation Algorithm* and the *Maze Solving Algorithm.*