Stacks in Python - A Pythonic Exposition

1. FILO and LIFO Strategy

Stacks operate on the principle of 'last in, first out' (LIFO). Also known as 'first in, last out' (FILO), this principle essentially means that the most recently added elements are the first to be removed. This behavior is similar to a stack of plates: you can only remove the top plate (the last one added), and the bottom plate (the first one added) will be the last one to be removed.

2. The push(element, stack) Function

The push() function is used to add an element to the stack. In Python, this can be easily implemented using the built-in list append() function.

```
def push(element, stack):
    stack.append(element)
```

3. The top(stack) and pop(stack) Functions

The top() function is used to get the top element of the stack without removing it. This can be implemented in Python by accessing the last element of the list. The pop() function, on the other hand, is used to remove and return the top element of the stack. Python's built-in list pop() function can be used for this.

```
def top(stack):
    return stack[-1]
```

```
def pop(stack):
```

```
    return stack.pop()
```

4. Pythonic Implementation of Stacks

In Python, stacks can be implemented using lists. Here's a simple class-based implementation:

```
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, element):
        self.stack.append(element)

    def top(self):
        return self.stack[-1]

    def pop(self):
        return self.stack.pop()
```

5. Differences Between Lists, Linked Lists, and Stacks in Python

Lists: Lists in Python are dynamic arrays that can hold elements of different data types. They are mutable, meaning their elements can be changed. Lists provide constant-time (O(1)) access to elements by index but adding or removing elements from the beginning or middle of the list requires shifting elements and takes linear time (O(n)).

Stacks in Python - A Pythonic Exposition

Linked Lists: Linked lists are collections of nodes where each node contains a value and a pointer to the next node in the list. Linked lists do not provide constant-time access to specific indices; finding an element requires traversing the list from the start to the required index, which takes O(n) time. However, adding or removing elements at the beginning of the list takes constant time.

Stacks: Stacks, as explained above, follow the LIFO principle. Elements are added (pushed) and removed (popped) at the 'top' of the stack. Both these operations take constant time. Accessing elements in the middle of the stack is not directly possible without popping elements off the top first.