Implementation of Stack

Stack Attributes and Methods

Before we implement our own Stack class, let's review the properties and methods of a Stack.

The stack abstract data type is defined by the following structure and operations. A stack is structured, as described above, as an ordered collection of items where items are added to and removed from the end called the "top." Stacks are ordered LIFO. The stack operations are given below.

- Stack() creates a new stack that is empty. It needs no parameters and returns an empty stack.
- push(item) adds a new item to the top of the stack. It needs the item and returns nothing.
- pop() removes the top item from the stack. It needs no parameters and returns the item. The stack is modified.
- peek() returns the top item from the stack but does not remove it. It needs no parameters. The stack is not modified.
- isEmpty() tests to see whether the stack is empty. It needs no parameters and returns a boolean value.
- size() returns the number of items on the stack. It needs no parameters and returns an integer.

Stack Implementation

In [1]:
```
class Stack:


    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def peek(self):
        return self.items[len(self.items)-1]

    def size(self):
        return len(self.items)
```

Let's try it out!

In [2]:
```
s = Stack()
```

In [3]:
```
print(s.isEmpty())
```

```
True
```

In [4]:
```
s.push(1)
```

In [5]:
```
s.push('two')
```

In [6]:
```
s.peek()
```

Out[6]:
```
'two'
```

In [7]:
```
s.push(True)
```

In [8]:
```
s.size()
```

Out[8]:
```
3
```

In [9]:
```
s.isEmpty()
```

Out[9]:
```
False
```

In [10]:
```
s.pop()
```

Out[10]:
```
True
```

In [11]:
```
s.pop()
```

Out[11]:
```
'two'
```

In [12]:
```
s.size()
```

Out[12]:
```
1
```

In [13]:
```
s.pop()
```

Out[13]:
```
1
```

In [14]:
```
s.isEmpty()
```

Out[14]:
```
True
```

Good Job!