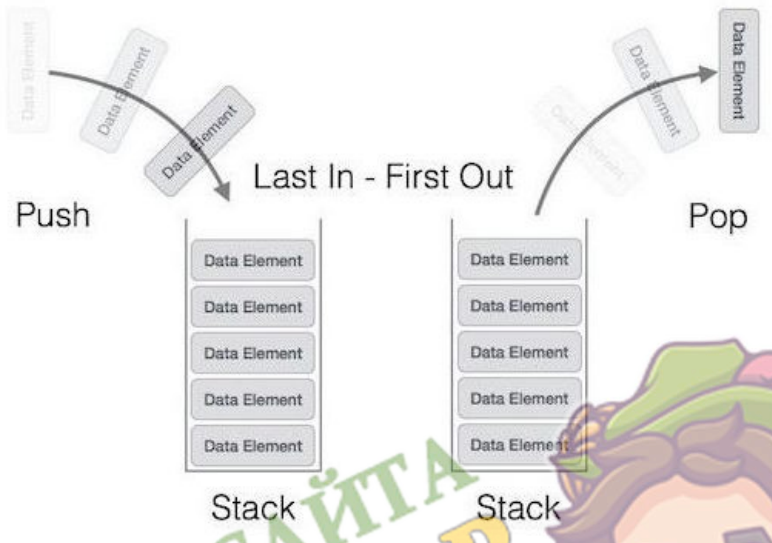
В этом задании вам предстоит реализовать свой стек(stack) - это упорядоченная коллекция элементов, организованная по принципу *LIFO* (англ. last in — first out, «последним пришёл — первым вышел»).



Ваша задача реализовать класс Stack, у которого есть:

- метод __init__ создаёт новый пустой стек. Параметры данный метод не принимает. Создает атрибут экземпляра values, где будут в дальнейшем хранятся элементы стека в виде списка (list), изначально при инициализации задайте значение атрибуту values равное пустому списку;
- `метод push(item)` добавляет новый элемент на вершину стека, метод ничего не возвращает.
- `метод pop()` удаляет верхний элемент из стека. Параметры не требуются, метод возвращает элемент. Стек изменяется. Если пытаемся удалить элемент из пустого списка, необходимо вывести сообщение "Empty Stack";
- `метод peek()` возвращает верхний элемент стека, но не удаляет его. Параметры не требуются, стек не модифицируется. Если элементов в стеке нет, распечатайте сообщение "Empty Stack", верните None после этого;
- `метод is_empty()` проверяет стек на пустоту. Параметры не требуются, возвращает булево значение.
- `метод size()` возвращает количество элементов в стеке. Параметры не требуются, тип результата - целое число.

```
s = Stack()
s.peek()  # распечатает 'Empty Stack'
print(s.is_empty())  # распечатает True
s.push('cat')  # кладем элемент 'cat' на вершину стека
s.push('dog')  # кладем элемент 'dog' на вершину стека
print(s.peek())  # распечатает 'dog'
s.push(True)  # кладем элемент True на вершину стека
print(s.size())  # распечатает 3
print(s.is_empty())  # распечатает False
s.push(777)  # кладем элемент 777 на вершину стека
print(s.pop())  # удаляем элемент 777 с вершины стека и печатаем его
print(s.pop())  # удаляем элемент True с вершины стека и печатаем его
print(s.size())  # распечатает 2
```