class Stack:
    """A last-in-first-out (LIFO) stack of items."
    Stores data in last-in, first-out order. When removing an item from the
    stack, the most recently-added item is the one that is removed.
    """

def __init__(self) -> None:
    """Initialize a new empty stack."
    """

def is_empty(self) -> bool:
    """Return whether this stack contains no items.
    """
    >>> s = Stack()
    >>> s.is_empty()
    True
    >>> s.push('hello')
    >>> s.is_empty()
    False
    """

def push(self, item: Any) -> None:
    """Add a new element to the top of this stack.
    """

def pop(self) -> Any:
    """Remove and return the element at the top of this stack.
    """
    >>> s = Stack()
    >>> s.push('hello')
    >>> s.push('goodbye')
    >>> s.pop()
    'goodbye'
    """

We are writing client code and need a function (outside the class) to determine the number of items on a stack.

1. Is the following a good solution? Explain.

```python
def size(s: Stack) -> int:
    """Return the number of items in s.
    """
    >>> s = Stack()
    >>> size(s)
    0
    >>> s.push('hi')
    >>> s.push('more')
    >>> s.push('stuff')
    >>> size(s)
    3
    """
    count = 0
    for _ in s:
        count += 1
    return count
```
2. Is the following a good solution? Explain.

def size(s: Stack) -> int:
    """Return the number of items in s."
    count = 0
    while not s.is_empty():
        s.pop()
        count += 1
    return count

Does more than documented (mutates s)

3. Is the following a good solution? Explain.

def size(s: Stack) -> int:
    """Return the number of items in s."
    return len(s._items)

Don't access private attributes

4. Is the following a good solution? Explain.

def size(s: Stack) -> int:
    """Return the number of items in s."
    s_copy = s
    count = 0
    while not s_copy.is_empty():
        s_copy.pop()
        count += 1
    return count

Creates an alias!

5. Given what you've learned, implement the function yourself on a separate sheet of paper.