StackBasics.py

```python
"""
A stack is a data structure that follows the LIFO principle - Last In, First Out.
Think of a stack like a stack of plates: you put plates on top (push), and you take plates from the top (pop).
You can only access the top plate, not the ones below.
"""
# Basic Stack Operations
# Push: Add an element to the top of the stack.
# Pop: Remove the top element from the stack.
# Peek/Top: Look at the top element without removing it.
# Is Empty: Check if the stack has any elements.

##We will use the first python inbuilt method for stacks: List
# This is because it supports .append() pus() and pop() methods

stack = []   # Create an empty list to represent the stack

stack.append(10)   # Add 10 to the top of the stack
stack.append(20) # Add 20 to the top of the stack
stack.append(30) # Add 30 to the top of the stack

print("Stack after pushes:", stack)  # Expected: [10, 20, 30]

# Peek at the top element (last element in list)
top_element = stack[-1]  # Access last element without removing it
print("Top element is:", top_element)  # Expected: 30

# Check if stack is empty
if len(stack) == 0:
    print("Stack is empty")
else:
    print("Stack is not empty")  # Expected here

 # The second method, is via custom classes, Here we implement a Stack class with all key operations.
class SimpleStack:
    def __init__(self):
        # Initialize an empty list to hold stack elements
        self.items  = []

    def is_empty(self):
        #Return True if the stack has no elements, otherwise return false.
        return len(self.items) == 0

    #add item to the top of the stack
    def push(self, item):
        self.items.append(item)


    # remove an item from the top and return it
    def pop(self):
        # if stack is empty return an error to avoid an invalid operation
        if self.is_empty():
            raise Exception("Cannot pop an empty stack")
        return self.items.pop()

    # PEEK: return the top item without removing it.
    def peek(self):
        #Raise an error is stack is Empty
        if self.is_empty():
            raise Exception("STACK IS EMPTY")
        #The top most eleemnt is the last to be pushed,
        # since actual size of list is greater than position by one,
        # minus one returns last element
        return self.items[-1]

    #SIZE: Return the number of all the items in the stack
    def size(self):
        return len(self.items)

    def print_stack(self):
```

```
        # Print all items in the stack from bottom to top
        print("Stack from bottom to top:", self.items)
        return

#Main meal
if __name__ == "__main__":
    #instantiate the class stack by creating an object for it.
    stack1 = SimpleStack()

    #Then, push some elements
    stack1.push(1000)
    stack1.push(2000)
    stack1.push(3000)

    #print the elements
    stack1.print_stack()

    # Peek top element
    print("Top element:", stack1.peek())  # Expected: 300

    # Pop elements
    print("Popped:", stack1.pop())  # Expected: 300
    stack1.print_stack()  # Expected: [100, 200]

    # Check if empty
    print("Is stack empty?", stack1.is_empty())  # Expected: False

    # Pop all to empty
    stack1.pop()
    stack1.pop()
    print("Is stack empty after popping all?", stack1.is_empty())

```