Pseudocode for a Stack

Assumes that array storage starts with element zero.

```
CONST SIZE=10  # stack capacity

FUNCTION stackInit() RETURNS INTEGER
  DECLARE stack ARRAY[0:SIZE-1] OF INTEGER
  DECLARE stackPointer:INTEGER
  stackPointer = -1  # empty stack
  RETURN(0)
ENDFUNCTION


FUNCTION push(value:INTEGER) RETURNS INTEGER
  DECLARE status:INTEGER

  IF stackPointer <SIZE-1 THEN
    stackPointer = stackPointer +1
    stack[stackPointer] = value
    status = value # successful push
  ELSE
    OUTPUT “Stack Overflow”    # stack is full already, can’t push a new value
    status = MAX_INTEGER       # large value to indicate error
  ENDIF
  RETURN(status)  # success or error
ENDFUNCTION

FUNCTION pop() RETURNS INTEGER
  DECLARE value:INTEGER
  IF stackPointer >= 0 THEN
    value = stack[stackPointer]
    stackPointer = stackPointer -1
  ELSE
    OUTPUT “stack underflow”      # stack is empty, nothing to pop
    value = MAX_INTEGER # large value to indicate error
  ENDIF
  RETURN(value)
ENDFUNCTION
```

With Python a stack is best implemented as a class so that it can easily be reused.

Typical uses:

- supporting subroutine calls (the return address is held on the stack)
- storing local variables
- the back button on a browser, the undo command on a word processor