# STACKS

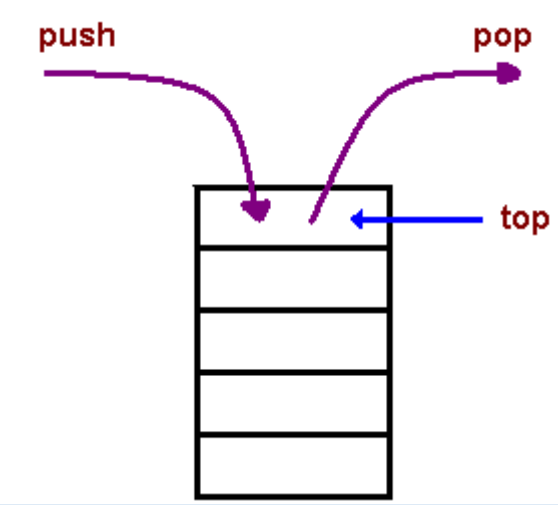
## THE DATA STRUCTURE

**#A stack** is a last in first out dynamic data structure, abbreviated to LIFO. Data is put into and removed from the structure from the same end.

Items are **pushed** (entered onto) or **popped** (retrieved) from the stack. Stacks are a dynamic data structure (they can grow in size/ constantly change). **Stacks can be static OR dynamic.**

**Data cannot be accessed directly** within a stack as all the items have to be removed that come after/pushed in more recently until you can retrieve the desired item. To implement a stack you need a top pointer to indicate the top of the stack in the array.

An empty stack will likely have a **top pointer = 0** and a full stack is one with the **top pointer** at a **position out of** the array's **bounds.**




## OPERATIONS ON THE STACK

| List operation | Description | Example | List contents | Return Value/updates list |
|---|---|---|---|---|
| pop() | Removes and returns the last item in the stack (top item) | a.pop() | [2,3,7,1] | [2,3,7] & returns 1 |
| Push(item) | Adds a new item to the top of the stack (end of list) | a.push(55 | [2,3,7,1] | [2,3,7,1,55] |
| Peek() | Returns the top item from the stack (last item in list) but does not remove it from the stack. | a.peek() | [`blue',`red',`green'] | green |
| isEmpty() | Test to see whether the stack is empty and returns a Boolean value | a.isEmpty() | [`blue',`red',`green'] | False |
| isFull() | Test to see whether the stack is full and returns a Boolean value | a.isFull() | [`blue',`red',`green'] | True |

| | | | | |
|---|---|---|---|---|
| s.size() | Same as .length() function, returns integer value for size of stack. | a.size() | [0,1,2] | 3 |

## APPLICATIONS OF THE STACK DATA STRUCTURE

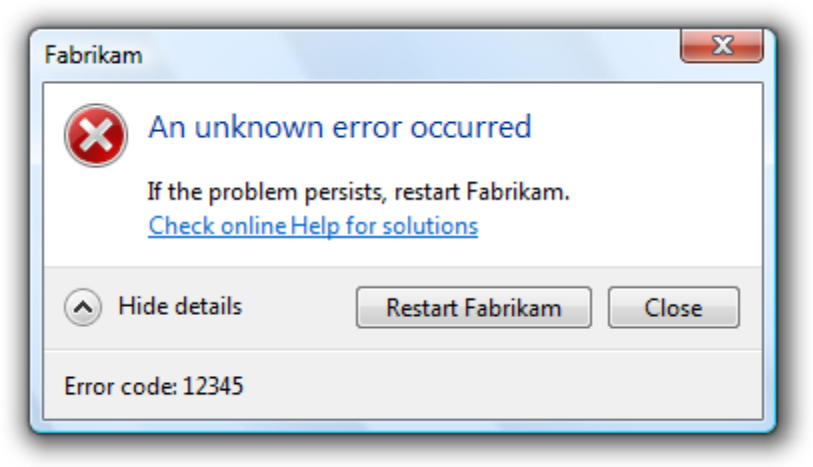
### Interrupts

When running a program, The CPU is fetching, decoding and executing instructions. However, peripherals or software may require attention and so send signals to the CPU along the control bus. These are known as **interrupts.**

Once the CPU has finished executing the current instruction, it will not proceed to the next instruction of the running program (in the program counter), instead it will have to copy all the data/instructions currently held in the CPU's registers and push them onto a stack. It will then jump to the instruction that will run the interrupt service routine which may comprise of numerous instructions. Once the interrupt service routine is complete (and provided there are no other interrupts), all the data is pushed off the stack back into the CPU 's registers and the CPU carries on from where it left of.

**Interrupt service routine (ISR)** = a software routine in an operating system or device driver whose execution is invoked by the reception of an interrupt.

REFER to Mrs Kirkland's notes on "**Operating system managing the CPU"** for more




## IMPLIMENTATION OF THE STACK DATA STRUCTURE

We typically require 2 additional variables when using an array (static) for a stack. One to indicate the top of the stack (the top_pointer) and another to indicate the maximum size of the stack.

## STACK PSEUDOCODE

```
PROCEDURE STACK():

        stack = [“”] * 10
        top_pointer = 0

        continue = “Y”
        WHILE continue == “Y”:
                choice = int(input(“Would you like to push(1), pop(2) or view items (3) from the stack?”))


                IF choice == 1 THEN
                        IF top_pointer == len(stack)  THEN
                                print(“The stack is full”)
                        ELSE:
                                item = str(input(“Enter item to push to stack: ”))
                                stack[top_pointer] = item
                                top_pointer = top_pointer + 1
                        END IF
                ELSE IF choice == 2 THEN
                        IF top_pointer == 0 THEN
                                print(“The stack is empty”)
                        ELSE:
                                item = stack[top_pointer – 1]
                                print(“Item popped: “, item)
                                stack[top_pointer – 1] =  “”
                                top_pointer = top_pointer – 1
                        END IF
                ELSE:
                        FOR i = 0 to len(stack -1):
                                print(“Item: “, stack[i])
                        NEXT i
                END IF

        continue = str(input(“Continue (y/n) ? “)
        continue = continue.upper()
        END WHILE

END PROCEDURE
```

PYTHON ALGORITHM (LIMITTED LENGTH STACK)

```
#LIFO Stack (last in first out) Adam Suttle

def stack_proc():
    stack = [""] * 10                           #Empty stack generated (limitted space of 10 items
    top_pointer = 0                             #Indicates the next empty position in the stack

#-------------------------------------------------------------------------------------------------------------
    response = "Y"
    while response == "Y":
#-------------------------------------------------------------------------------------------------------------

        choice = int(input("Would you like to pop (1) or push (2) or view items(3) from the the stack? "))

        if choice == 1:                             #Pop items from stack

            if top_pointer == 0:                    #No items
                print("The stack is empty")
            else:
                item = stack[top_pointer - 1]       #Removes last item that is not ""
                top_pointer -= 1                    #Decreases pointer by 1
                stack[top_pointer] = ""             #New empty position
                print("Item popped: ", item)

        elif choice == 2:                           #Push item to stack

            if top_pointer == len(stack):           #I.e. the pointer is greater than the last item in the stack
                print("The stack is full")
            else:
                item = str(input("Enter an item to push onto the stack: "))
                stack[top_pointer] = item           #Last "" in stack now holds item
                top_pointer += 1                    #Incremeents the pointer by 1
        else:
            for i in range (0, len(stack)):                 #Displays every item in the stack
                print("Item ", i, ": ", stack[i])

#-------------------------------------------------------------------------------------------------------------
    response = str(input("\nContinue? (y/n)"))
    response = response.upper()
#-------------------------------------------------------------------------------------------------------------

stack_proc()
```

```
#THis is a similar algorithm for a more dynamic approach where the list has unlimittted length

def stack_proc_unlimit():
    stack = []                                  #Empty stack generated (unlimitted length)
    top_pointer = 0                             #Indicates the next empty position in the stack

#-------------------------------------------------------------------------------------------------------------
    response = "Y"
    while response == "Y":
#-------------------------------------------------------------------------------------------------------------

        choice = int(input("Would you like to pop (1) or push (2) or view items(3) from the the stack? "))

        if choice == 1:                                                     #Pop items from stack

            if top_pointer == 0:                                            #No items
                print("The stack is empty")
            else:
                item = stack[top_pointer - 1]
                top_pointer -= 1                                            #Decreases pointer by 1
                stack.pop(top_pointer)                                      #Deletes last element in stack
                print("Item popped: ", item)

        elif choice == 2:                                                   #Push item to stack

            item = str(input("Enter an item to push onto the stack: "))
            stack.append(item)
            #New item is at top of stack
            top_pointer += 1                                                #Incremeents the pointer by 1

        else:
            for i in range (0, len(stack)):                                 #Displays every item in the stack
                print("Item ", i, ": ", stack[i])

#-------------------------------------------------------------------------------------------------------------
    response = str(input("\nContinue? (y/n)"))
    response = response.upper()
#-------------------------------------------------------------------------------------------------------------

#stack_proc_unlimit()
```