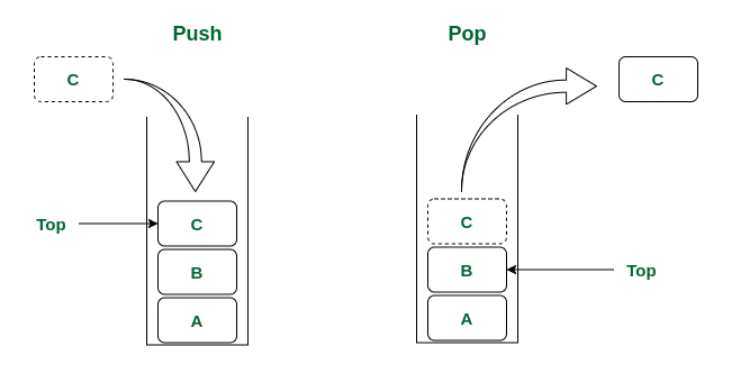
Define Stack

Stack is a data structure in which insert and delete operations are performed at the same end.

It is a collection of objects that works on the LIFO principle. This means that the element which is inserted last will be deleted first.

LIFO stands for Last In First Out.

A user may insert objects into a stack at any time, but may only access or remove the most recently inserted object.



For Example,

1. Web Browser

Internet Web browsers store the addresses of recently visited sites in a stack.

Each time a user visits a new site, that site’s address is “pushed” onto the stack of addresses. The browser then allows the user to “pop” back to previously visited sites using the “back” button.

2. Text Editor

Text editors usually provide an “undo” mechanism that cancels recent editing operations and reverts to the former states of a document.

This undo operation can be accomplished by keeping text changes in a stack.

Characteristics of Stack

Insert operation is referred to as PUSH operation.

Delete operation is referred to as the POP operation.

Insert and Delete operations are performed at only one end, so the stack follows the LIFO principle.

Pointed TOP keeps the track of the top element of the stack. Initially, the value of the top is -1.

Each time a new element is inserted, TOP will be increased by one and if a value is deleted then the value of the top will be decremented by one.

Since the elements or data are stored sequentially in the stack, they are linear data structures.

The most accessible element in the stack is referred to as a TOP element and the least accessible element in the stack is known as the BOTTOM element.

Implementation of Stack using List

Operations on Stack

There are two operations that can be performed on a stack.

1. PUSH

2. POP

1. PUSH

The process of inserting an element into a stack is known as PUSH operation.

In order to insert an element into the stack, we must check whether free space is available in the stack or not.

If space isn’t available then we cannot insert the element. This condition is known as ‘Overflow’.

Here we will implement the stack using list, python will not restricts us to specify a particular size of list. Size of list is variable. So overflow condition won’t occur if we are implementing the stack using list.

But, we can specify the maximum size of the list using seprate variable named maxsize. In this case we use variable called top which holds index of the topmost element of the stack.

If value of top is equal to maxsize – 1 that means the stack is full or we can is say that stack is overflow.

If the stack is not overflow then we can insert an element into the stack.

First we need to increase the value of top by one and then insert an element into the stack.

Algorithm

The algorithm inserts an element on the top of the stack.

Here,

Initially, the value of the top is 0 [-1]

The size keyword represents the length of the stack.

The ele represents the element to be inserted into the stack.

S represents our stack.

Step 1 – [Check for the stack overflow]

If top >= size [size-1] then

Write – Stack Overflow

Return

Step 2 – [Incrementing the value of TOP]

TOP 🡨 TOP + 1

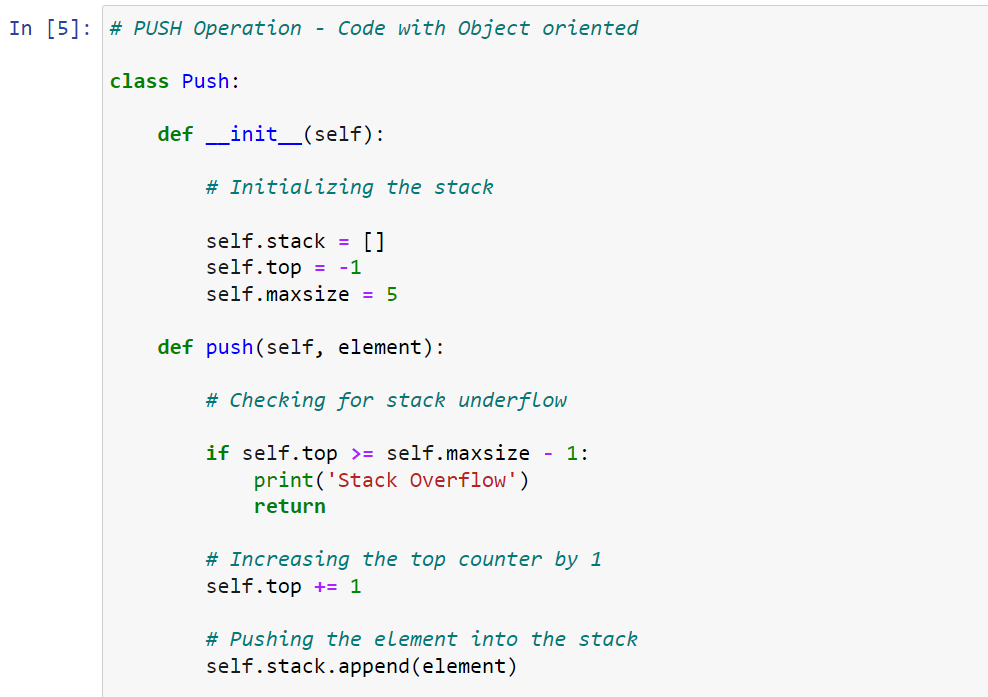
Step 3 – [Insert the element on the top of the stack]

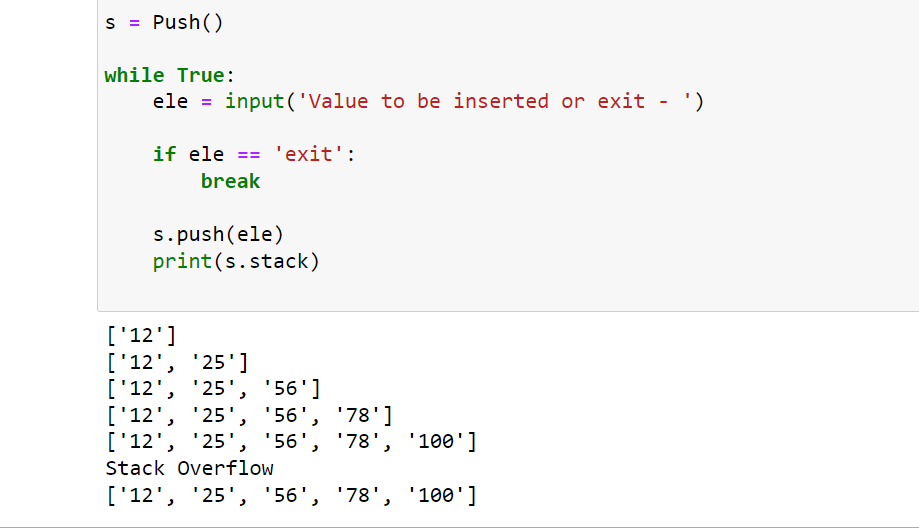
S[TOP] 🡨 ele

Step 4 – [Finished]

Return()

Code





2. POP

Applications of Stack

* Stack is widely used in the concepts of recursion because of its Last In First Out characteristics.
* Stack is used by the compiler for parsing the syntax of the expression.
* Stack is used for reversing the string, in which first all the elements are pushed into the stack and then they are popped from the stack.
* Stack is widely used in the evaluation of polish notation. For example, the stack can be used to convert infix expressions into postfix expressions.
* Programming languages use stack at the time of calling the function to store values of former parameters and return address.
* They are used for matching tags in HTML and XML.
* Stacks are used in web browsers for returning the address of the last visited website by the user.
* They are used in the text editors such as notepad, and Microsoft word for performing the undo operation.
* Stacks are useful for backtracking and parenthesis matching.

Polish Notation

Polish Notation is a method of expressing mathematical, logical, and algebraic equations universally.

Types of Polish Notations

Infix – When the operator is written between two operands, it is known as infix notation. For example, A + B

Prefix – When the operator is written before their operands, it is known as prefix notation. For example, +AB

Postfix – When the operator is written after their operands, it is known as postfix notation.

Proceedings for polish notations conversions

|  |  |
| --- | --- |
| Symbol | Priority |
| ( ) | Highest |
| $ ^ |  |
| \* / % |  |
| + - | Lowest |

Infix to Prefix

Theory

Expression – (a\*b+c) + (d\*e/b)

Infix to Postfix

Theory

Expression – (a\*b+c) + (d\*e/b)

Using Stack with preceding logic

|  |  |  |
| --- | --- | --- |
| Symbol | Stack | Postfix |
| ( | ( |  |
| A | ( | A |
| \* | (\* | A |
| B | (\* | AB |
| + | (+ | AB\* |
| C | (+ | AB\*C |
| ) |  | AB\*C+ |
| + | + | AB\*C+ |
| ( | +( | AB\*C+ |
| D | +( | AB\*C+D |
| \* | +(\* | AB\*C+D |
| E | +(\* | AB\*C+DE |
| / | +(/ | AB\*C+DE\* |
| B | +(/ | AB\*C+DE\*B |
| ) | + | AB\*C+DE\*B/ |
|  |  | AB\*C+DE\*B/+ |

Expression – (a + (b \* c- (m/n $p) \*t) \*s)

Using Stack with preceding

|  |  |  |
| --- | --- | --- |
| Symbol | Stack | Postfix |
| ( | ( |  |
| A | ( | a |
| + | (+ | a |
| ( | (+( | Ab |
| B | (+( | Ab |
| \* | (+(\* | Ab |
| C | (+(\* | Abc |
| - | (+(- | Abc\* |
| ( | (+(-( | Abc\* |
| M | (+(-( | Abc\*m |
| / | (+(-(/ | Abc\* |
| N | (+(-(/ | Abc\*mn |
| $ | (+(-(/ | Abc\*mn$ |
| P | (+(-(/ | Abc\*mn$p |
| ) | (+(- | Abc\*mn$p/ |
| \* | (+(- | Abc\*mn$p/\* |
| T | (+(- | Abc\*mn$p/\*t |
| ) | (+ | Abc\*mn$p/\*t- |
| \* | (+ | Abc\*mn$p/\*t-\* |
| S | (+ | Abc\*mn$p/\*t-\*s |
| ) |  | Abc\*mn$p/\*t-\*s+ |
|  |  |  |

Evaluate the postfix expression

1. Initialize empty stack

2. Scan expression from left to right

3. If operand then push element into stack

4. If scanned character is operator then perform this – last element <operator> second last element and push the result of this into the stack.

5. Repeat steps 1 to 4 until all the characters are scanned

Evaluate 234\*+5-

|  |  |  |
| --- | --- | --- |
| 2 | Push | 2 |
| 3 | Push | 2 3 |
| 4 | Push | 2 3 4 |
| \* | Multiply | 2 12 |
| + | Add | 14 |
| 5 | Push | 14 5 |
| - | Minus | 9 |
|  |  |  |

If you evaluate any postfix expression, it will be converted to infix

Evaluate ABC+\*D\*

|  |  |  |
| --- | --- | --- |
| A | Push | a |
| B | Push | A b |
| C | Push | A b c |
| + | Add | A (b + c) |
| \* | Multiply | A \* (b+c) |
| D | Push | (A \* (b+c)) D |
| \* | Multiply | (A \* (b + c)) \* D |
|  |  |  |

Recursive Functions

A recursive function is a function that calls itself in its execution.

Following are the problems that can be solved through recursive functions

1. Factorial

2. Greatest Common Divisor [GCD]

3. Fibonacci Series

1. Factorial

The factorial of a positive integer n is denoted by n!

Factorial is defined as the product of the integers from 1 to n. If = 0, then n! is 1.

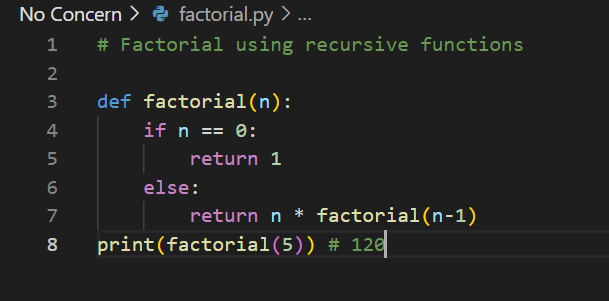
Algorithm

Step 1 – n 🡨 integer to get the factorial of

Step 2 – if n == 0 then return 1

Step 3 – if n > = then return n \* factorial(n-1)

Code



2. Greatest Common Divisor [GCD]

GCD of two integers is the largest number that divides both the numbers.

Algorithm

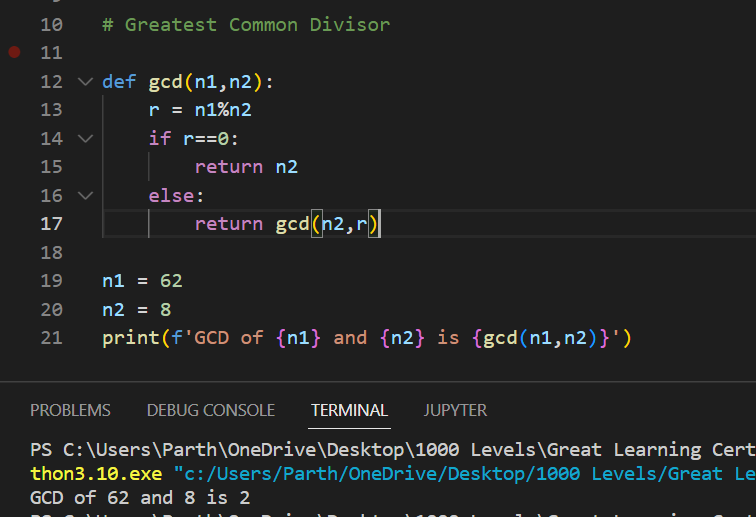
Step 1 – Assign values to n1 and n2

Step 2 – r = n1 Mod n2

Step 3 – if r == 0 then return n2

Step 4 – id r > = then return gcd(n2, r)

Code



3. Fibonacci Series

More on Stack

S.top( ): Return a reference to the top element of stack S, without removing it; an error occurs if the stack is empty.

S.is\_empty( ): Return True if stack S does not contain any elements.

len(S): Return the number of elements in stack S

List as Stack

Although a programmer could directly use the list class in place of a formal stack class, lists also include behaviors (e.g., adding or removing elements from arbitrary positions) that would break the abstraction that the stack ADT (Abstract Data Type) represents.

Also, the terminology used by the list class does not precisely align with traditional nomenclature for a stack ADT, in particular the distinction between append and push. Instead, we demonstrate how to use a list for internal storage while providing a public interface consistent with a stack.

