El presente informe tiene como finalidad explicar la ejecución de una función "my\_stack\_function” y como se ejecuta paso por paso, a continuación de dará una breve explicación de las instrucciones presentes en el código:

La clase Stack tiene los siguientes métodos

|  |  |
| --- | --- |
| class Stack:  def \_\_init\_\_(self):  self.items = []  def isEmpty(self):  return self.items == []  def push(self, item):  self.items.append(item)  def pop(self):  return self.items.pop()  def top(self):  return self.items[len(self.items)-1]  def size(self):  return len(self.items) | \_\_init\_\_(self): inicializa la pila con una lista vacía. |
| isEmpty(self): verifica si la pila está vacía, devolviendo True si lo está y False si no. |
| push(self, item): agrega un elemento a la parte superior de la pila. |
| pop(self): elimina y devuelve el elemento en la parte superior de la pila. |
| top(self): devuelve el elemento en la parte superior de la pila sin eliminarlo. |
| size(self): devuelve el tamaño de la pila. |

Antes de comenzar se dará una breve explicación a través de conceptos simples de los métodos o instrucciones de ejecución:

Init: es un método especial que se llama cada vez que se instancia una clase y sirve para inicializar el objeto que se crea. Este método crea atributos que deben tener todos los objetos de clase y por lo tanto contiene los parámetros necesarios para su creación, pero no devuelve nada.

Self: Es una convención que se utiliza como nombre para el primer parámetro de un método en una clase. El objetivo de "Self" es hacer referencia al objeto que se está manipulando cuando se llama al método.

isEmpty: Es una función que devuelve "true" si la expresión evaluada es un valor de celda vacía.

Push: Permite agregar los nuevos elementos que pasan como parámetros, al final del arreglo y luego nos devuelve una nueva longitud de este arreglo.

Pop: Permite eliminar y retornar un elemento de una lista.

Top: Crea una función que al recibir un numero entero verifica que si al sumar las cifras del número y dividirlo por su cantidad de cifras. Si es así, se dice que es un numero "top"

Size: Cuando se aplica el método "size" a un arreglo o a una estructura de datos, retorna al número total de elementos contenido en esa estructura.

Items: Cada elemento es un diccionario, contiene una clave y un valor " es decir, un par de clave-valor" cada par de clave-valor es denominado como elemento (items)

Stack: Permite almacenar objetos y luego recuperarlos en el orden inverso en el cual se insertaron.

Return: indica el final de la función, pero también el valor que devuelve la función.

If: Se utiliza la palabra seguida de la condición que debe cumplirse.

Else: maneja las relaciones en el proceso del lenguaje del código.

While: ejecuta la misma tarea varias veces.

Temp: nos permite crear otra pila temporal donde se almacenan los elementos.

La función “function(arr)” toma un arreglo como entrada y utiliza la pila para ordenar los elementos del arreglo, el algoritmo utilizado es una versión modificada del algoritmo de ordenamiento por inserción.

Una vez explicado cada uno de los métodos presentes en arreglo se procede al explicar la función, en este caso el problema en resolver:

|  |
| --- |
| Crear una función my\_stack\_function la cual recibe un arreglo de números enteros (pueden ser positivos o negativos) y retorna una pila ordenada. |
| class Stack:  def \_\_init\_\_(self):  self.items = []    def isEmpty(self):  return self.items == []    def push(self, item):  self.items.append(item)    def pop(self):  return self.items.pop()    def top(self):  return self.items[len(self.items)-1]    def size(self):  return len(self.items)    def function(arr):  stack = Stack()  for num in arr:  if stack.isEmpty() or num > stack.top():  stack.push(num)  else:  temp\_stack = Stack()  while not stack.isEmpty() and num < stack.top():  temp\_stack.push(stack.pop())  stack.push(num)  while not temp\_stack.isEmpty():  stack.push(temp\_stack.pop())  return stack.items  array = [27, -13,30, -5,90, -98,44, -18]  print ("Entrada:", array)  print ("Resultado:", function(array)) |

Ejecución para la solución

* Se determina a través de un “def” la función del arreglo.
* El stack es la identificación de una pila que debe ser igual a una pila vacía.
* Se utiliza un “for” para determinar la función de los números o elementos que están en el arreglo
* El “if” determina la condicion y el isEmpty el espacio vacío seguido del “num” que debe ser mayor que el Stack. El numero o el elemento es agregado en otra pila.
* Creamos una pila temporal utilizando “temp” para almacenar los números con un “push”
* Utilizando un while, se evalúa la condición y luego ejecuta un bloque de código si la condición es verdadera. El bloque de código se ejecuta repetidamente hasta que la condición ser verdadera o falsa, se evalúa la pila con un “isEmpy” si esta vacía y el numero deber ser menor que el numero dentro de la pila.
* Se evalúa la pila temporal con los datos agregados y la pila donde se elimina el elemento, la pila donde se agregan los datos “Stack.push(num)”
* Nuevamente ejecuta un ciclo while con la pila temporal junto a un “isEmpty” para determinar si es verdadera y si esta vacia, con la función “Stack” junto a un “push” se agregan los elementos en una pila temporal para eliminar y retomar.
* Finalmente se devuelve la lista de elementos en la pila original, que representa el arreglo ordenado.
* Se crea un arreglo llamado “array” y se llama a la “function” pasando el arreglo como argumento. Se imprime utilizando un “Print” el arreglo de entrada y el resultado de la función