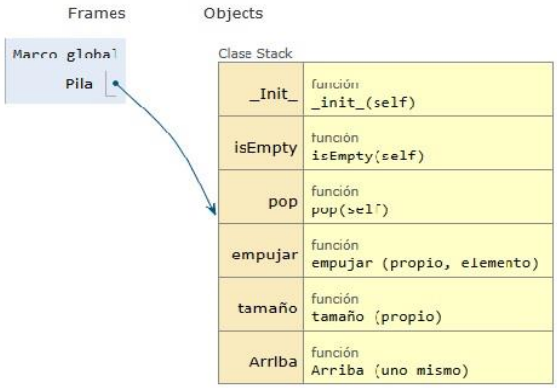
UPTEC “Mariscal Sucre ”
Tarea III Algoritmo
Alumna Marbelys Reyes V-18816574
Profesor José Ortiz
Sección 7122

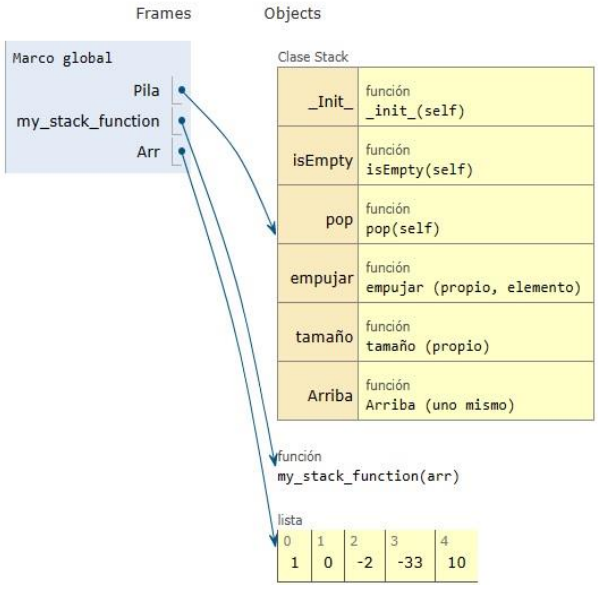
En este código se utiliza una pila implementada mediante el class stack, tiene como función específica, trabajar con pila.

```
11      def pop(self):
12          return self.items.pop()
13
14  def top (self):
15          return self.items[len(self.items)-1]
16
17  Tamaño de definición (propio):
18          return len(self.items)
19
20  def my_stack_function(arr):
21      stack = Stack()
22
23      # Ordenar el arreglo de forma ascendente
24  sorted_arr = ordenado(arr)
25
26  Para num en sorted_arr:
27          stack.push(num)
28
```



El algoritmo toma un arreglo de números enteros y devuelve una pila ordenada. El objetivo es insertar los elementos del arreglo en una pila de forma ordenada.

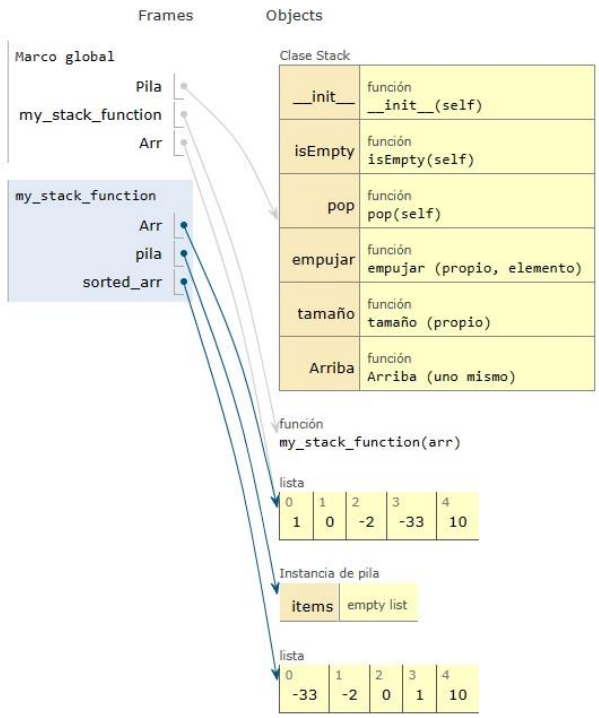
```
23      # Ordenar el arreglo de forma ascendente
24  sorted_arr = ordenado(arr)
25
26  Para num en sorted_arr:
27          stack.push(num)
28
29  Pila de devolución
30
31  # Ejemplo de uso:
32  arr = [1, 0, -2, -33, 10]
33  pila = my_stack_function(arr)
34
35  # Mostrar si la pila está vacía
36  print(stack.isEmpty())  # True
37
38  # Apilar el valor 100
39  stack.push(100)
40
```



Es decir, tengo una lista con elementos desordenados y hay que ordenarlos mediante el algoritmo.

Entonces, el código debe ordena los enteros de forma ascendente y Después de recorrer cada número ordenado y arreglado,

```
        return self.items[len(self.items)-1]

Tamaño de definición (propio):
        return len(self.items)

def my_stack_function(arr):
    stack = Stack()

    # Ordenar el arreglo de forma ascendente
sorted_arr = ordenado(arr)

Para num en sorted_arr:
        stack.push(num)

Pila de devolución

# Ejemplo de uso:
arr = [1, 0, -2, -33, 10]
```

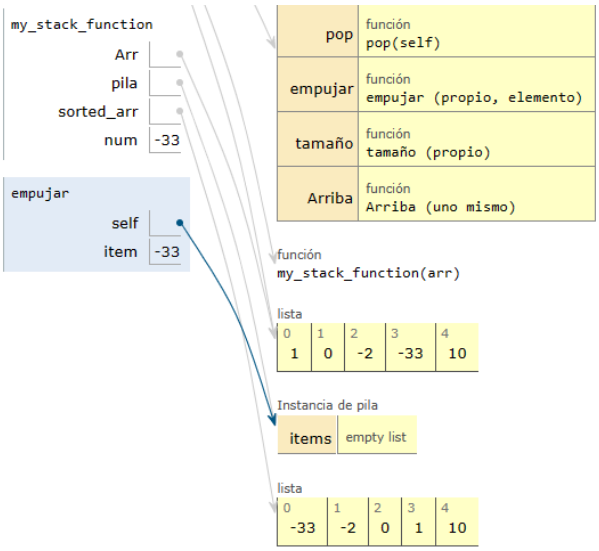


Se inserta la pila utilizando la función push(num), luego se implementa una pila que es necesaria para mantener los elementos ordenados y permitir el acceso a ellos mediante la regla de las pilas, donde el último que entra, es el primero que sale.

Para eso utilizo el push, que es el último elemento, el pop tiene como función eliminar el último elemento agregado.

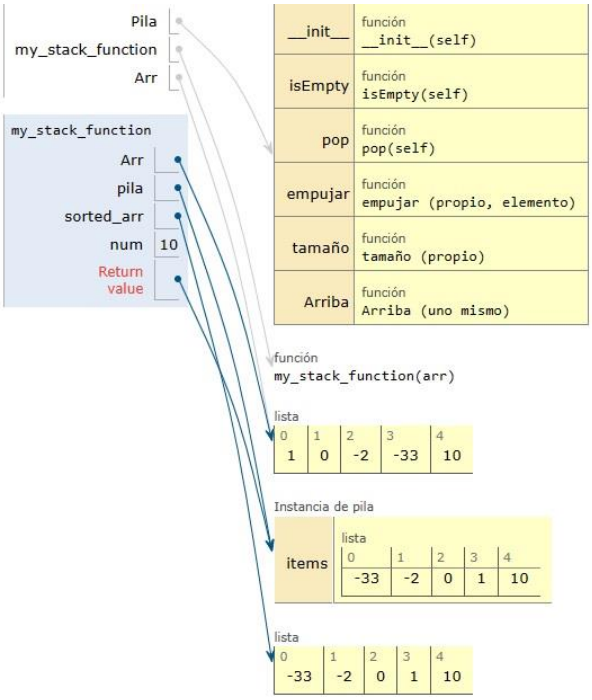
```
def push (self, item):
self.items.append(elemento)

    def pop(self):
        return self.items.pop()

def top (self):
        return self.items[len(self.items)-1]

Tamaño de definición (propio):
        return len(self.items)
```

Editar código



Al tener todas las funciones en el código fuente y el push para ingresar, se coloca un append y el pop lo que hace es retirar el último número que ingresó.

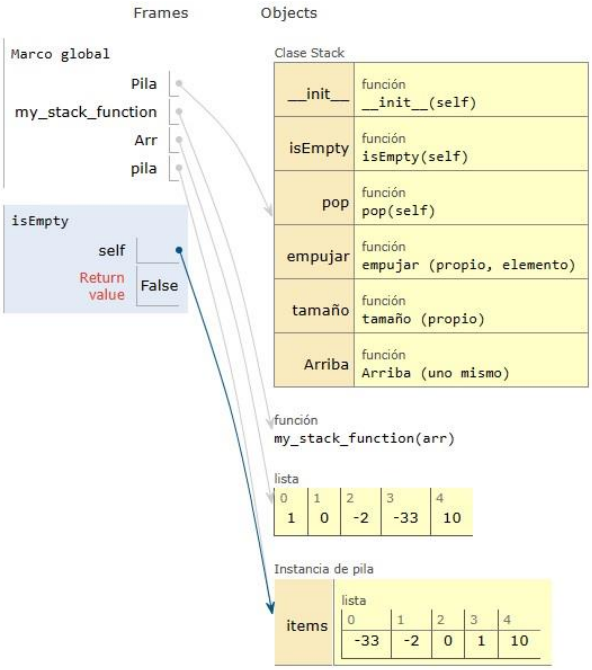
```
29 Pila de devolución
30
31 # Ejemplo de uso:
32 arr = [1, 0, -2, -33, 10]
33 pila = my_stack_function(arr)
34
35 # Mostrar si la pila está vacía
36 print(stack.isEmpty())  # True
37
38 # Apilar el valor 100
39 stack.push(100)
40
41 # Apilar el valor 200
42 stack.push(200)
43
44 # Desapilar el valor 200
```



Y size, lo que hace es regresar la cantidad de elementos que tenga en la pila.

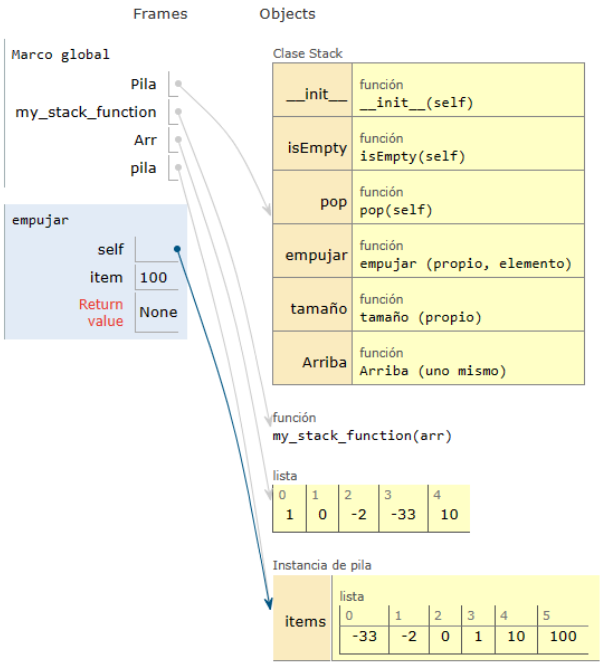
Al tener la pila ordenada, se obtiene el primer número y así la secuencia de la pila, también se visualiza el size del array, y un pop que corresponde a si está vacía.

```
1  clase Pila:
2      def __init__(self):
3          self.items = []
4
5      def isEmpty(self):
6          return self.items == []
7
8  def push (self, item):
9  self.items.append(elemento)
10
11     def pop(self):
12         return self.items.pop()
13
14 def top (self):
15         return self.items[len(self.items)-1]
16
17 Tamaño de definición (propio):
18         return len(self.items)
```



Si está vacía, me va a devolver un false. Se tiene push y le ingresamos valores de la lista y luego, esta el pop que elimina algún valor. Lo que hace es mostrar el valor que se está borrando. Allí muestra si es el valor del array está y top muestra el primer valor entonces, me queda el 100.

```
clase Pila:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

def push (self, item):
self.items.append(elemento)

    def pop(self):
        return self.items.pop()

def top (self):
        return self.items[len(self.items)-1]

Tamaño de definición (propio):
        return len(self.items)
```

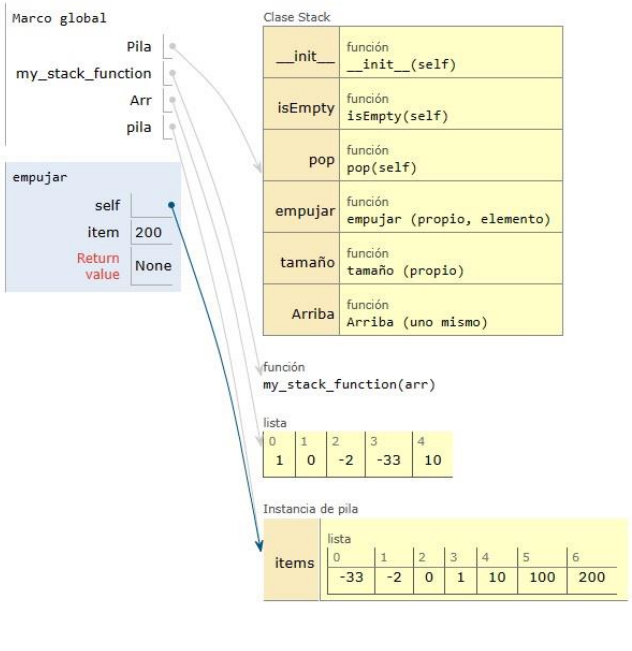


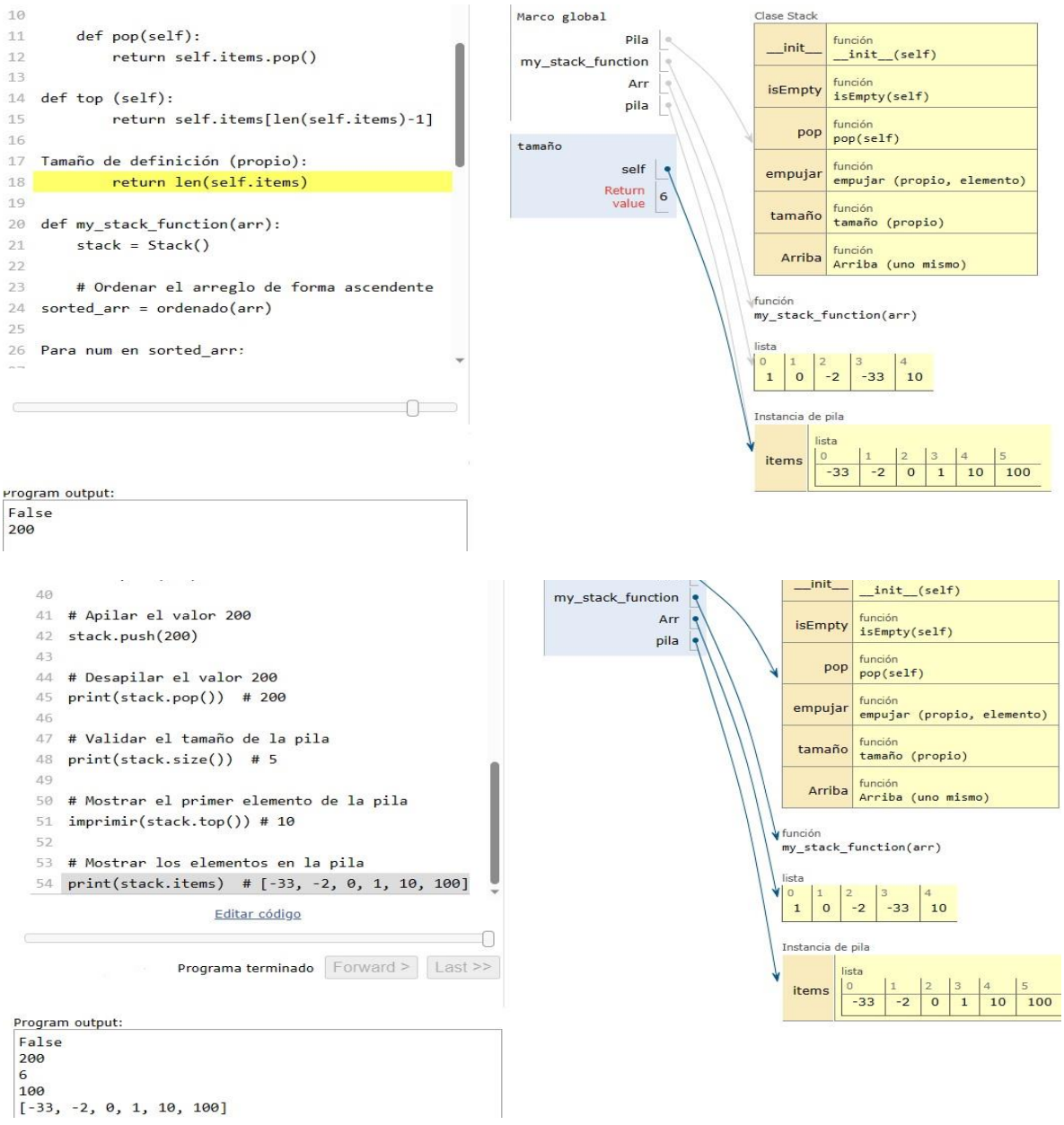
Aquí me queda el 100, pero el 200 que también lo agregamos aquí, que fue el último que agregamos, realmente fue el primero que salió. Entonces, así es como va avanzando el código.

```
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

def push (self, item):
self.items.append(elemento)

    def pop(self):
        return self.items.pop()

def top (self):
        return self.items[len(self.items)-1]

Tamaño de definición (propio):
        return len(self.items)
```

Program output:

```
False
```





La pila se implementa utilizando la clase "Stack" y tiene métodos como "push" para agregar elementos, "pop" para eliminar el último elemento agregado, "top" para mostrar el primer elemento de la pila y "size" para obtener la cantidad de elementos en la pila. El objetivo es tomar un arreglo de números enteros desordenados y devolver una pila ordenada. El código recorre los números ordenados y los inserta en la pila utilizando el método "push". Además, se utilizan otros métodos para mostrar información sobre la pila, como si está vacía o no.