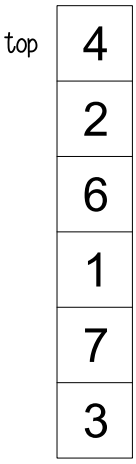
Pila de valores enteros

∗ Escribe una clase `IntegerStack` que represente una pila de valores enteros ∗



Implementa, *al menos*, los siguientes **métodos**:

```
def __init__(self, *, max_size: int = 10):
```

- Crea los atributos: `items` y `max_size`.
- Ten en cuenta que `items` almacenará los elementos de la pila.

```
def push(self, item: int) -> bool:
```

- Añade `item` a la pila `self`.
- Si la pila está llena habrá que retornar `False`. En otro caso retornar `True`.

```
def pop(self) -> int:
```

- Extrae (y devuelve) el elemento que está en el ***top*** de la pila `self`.

```
def top(self) -> int:
```

- Devuelve el elemento que está en el ***top*** de la pila `self`.

```
def is_empty(self) -> bool:
```

- Indica si la pila `self` está vacía.

```
def is_full(self) -> bool:
```

- Indica si la pila `self` está llena.

`def expand(self, factor: int = 2) -> None:`

- Expande el tamaño máximo de la pila `self` en el factor indicado.
- Por ejemplo, si el tamaño máximo es 20 y el factor es 3, el tamaño máximo pasa a ser $20 \times 3 = \mathbf{60}$.

`def dump_to_file(self, path: str) -> None:`

- Vuelca la pila `self` a un fichero con ruta `path`.
- Cada elemento de la pila en una línea diferente.
- El primer elemento del fichero coincide con el elemento del ***top*** de la pila.

`def load_from_file(cls, path: str) -> IntegerStack:`

- Es un **método de clase**.
- Construye (y devuelve) una pila desde el fichero con ruta `path`.
- Cada item de la pila en una línea diferente.
- El primer elemento del fichero coincide con el elemento del ***top*** de la pila.
- Si la pila se llena al ir añadiendo elementos habrá que expandir con los valores por defecto.

`def __len__(self) -> str:`

- Devuelve el número de elementos que hay en la pila `self`.

`def __getitem__(self, index: int) -> int:`

- Devuelve el elemento de la pila `self` que ocupa la posición (*índice*) `index`.

`def __setitem__(self, index: int, item: int) -> None:`

- Asigna el elemento `item` en la posición (*índice*) `index` de la pila `self`.

`def __add__(self, other: IntegerStack) -> IntegerStack:`

- Suma las pilas `self` y `other`.
- La segunda pila `other` va "encima" de la primera `self`.
- El tamaño máximo de la pila resultante es la suma de los tamaños máximos de cada pila.
- Devuelve la pila resultante.

`def __iter__(self) -> IntegerStackIterator:`

- Devuelve un objeto iterador de tipo `IntegerStackIterator`.

∗ Escribe una clase `IntegerStackIterator` que pueda iterar sobre `IntegerStack` ∗

Implementa, *al menos*, los siguientes **métodos**:

`def __init__(self, stack: IntegerStack):`

`def __next__(self) -> int:`