Tarea 3

Alejandro Martínez López.

Comando	Acción	Ejemplo

__add__(self, value, /)
Return self+value.
	Regresa la concatenación de dos listas.	>>> [1,2,3].__add__([4,5,6])
		[1,2,3,4,5,6]
		
		>>> [1,2,3]+[4,5,6]
		[1,2,3,4,5,6]

__contains__(self, key, /) Return key in self.	Regresa si un elemento (key) pertenece a la lista.	>>>[1,2,3].__contains__(1) True >>> 4 in [1,2,3] False

__delitem__(self, key, /)
Delete self[key].
	Elimina una entrada de la lista.	>>> aux=[1,2,3]
		>>>aux.__delitem__(1)
		>>>print(aux)
		>>> [1,3]

__eq__(self, value, /) Return self==value.	Indica si dos listas son iguales	>>> [1,2,3].__eq__[1,2,3] >> True >>>[1,2,3]==[1.3.2] False
__ge__(self, value, /) Return self>=value.	Determina si el elemento de una lista es mayor o igual que otro elemento de la lista.	>>>aux_1=[1,2,3] >>>aux_2=[4,5,6] >>>aux_1[1]>=aux_2[1] >>>False >>>aux_1=[5,2,3] >>>aux_2=[4,5,6] >>>aux_1[1]>=aux_2[1] >>>True

__getattribute__(self, name, /) Return getattr(self, name)		
__getitem__(...) x.__getitem__(y) <==> x[y]	Obtiene la entrada I-ésima de una lista.	>>>[1,2,5].__getitem__(2) >>5 >>>[1,2,5][2] >>5
__gt__(self, value, /) Return self>value.	Indica si el elemento de una lista es mayor que otro elemento de la lista.	>>>[1,2,3][0].__gt__([3,4,8][1]) >>False >>>[1,2,3][2]>[1,2,3][0] >>>True
__iadd__(self, value, /) Implement self+=value.	Concatena una lista (self) con otra lista (value) y el resultado lo guarda en primer lista (self).	>>>a=[1,2,3] >>>b=[4,5,6] >>>a.__iadd__(b) >>>print(a) >>>[1,2,3,4,5,6] >>>a=[1,2,3] >>>b=[4,5,6] >>>a+=b >>>print(a) >>>[1,2,3,4,5,6]
__imul__(self, value, /) Implement self*=value.	Concatena una lista consigo misma el numero de veces que se le indique, y guarda el resultado en la misma variable.	>>>a=[1,2,3] >>>a.__imul__(3) >>>print(a) >>>[1,2,3,1,2,3,1,2,3] >>>a=[1,2,3] >>>a*=(2) >>>print(a) >>>[1,2,3,1,2,3]
__init__(self, /, *args, **kwargs)	Guarda una lista en otra ya existente. Si el argumento es vacío, le asignara la lista vacía.	>>>a=[1,2] >>>b=[4,5]

Initialize self. See help(type(self)) for accurate signature.		>>>b.__init__(a) >>>print(b) >>>[1,2] >>>b=[4,5] >>>b.__init__() >>>print(b) >>>[ ]
__iter__(self, /) Implement iter(self).	Permite iterar los elementos de una lista sobre una misma variable.	>>>a=[7,8,9] >>>b=iter(a) >>>print(next(b)) >>>7 >>>print(next(b)) >>>8 >>>print(next(b)) >>>9
__le__ (self,value,/) Return self<=value	Regresa True si el valor ingresado es mayor o igual a un elemento de la lista y False en caso contrario.	>>>[1,7].__le__([2]) True >>>[2,3].__le__([1]) False
__len__(self, /) Return len(self)	Regresa la longitud de la lista (el número de elementos de la lista).	>>>len([1,2,5,6,7]) 5 >>>len([2,4,6]) 3
__lt__(self,value,/) Return self < value	Regresa True si el valor ingresado es mayor (estricto) a un elemento de la lista y False en caso contrario.	>>>[3,4,5].__lt__([3]) False >>>[2,4,7].__lt__([4]) True
__mul__(self,value,/) Return self*value	Regresa la concatenación de una lista consigo misma el número del valor. Equvalente a list*value.	>>>[3,4].__mul__(3) [3,4,3,4,3,4] >>>[3,4]*3 [3,4,3,4,3,4]

__ne__(self,value,/) Return self != value	Regresa True si la lista (valor) es distinta a la lista original y regresa False en caso contrario.	>>> [1,2].__ne__([3,4]) True >>>[1,2].__ne_([1,2]) False
__repr__(self,/) Return repr(self)	Convierte toda la lista en una cadena.	>>>repr([1,2,3]) '[1,2,3]'
__reversed__(self, /) Return a reverse iterador over the list	Regresa el iterador inverso (no se devuelve una lista).	>>>reversed([1,2,3]) <list_reverseiterator object at 0x000001B09C5EA890> >>>list(reversed([1,2,3])) [3,2,1]
__rmul__(self, value, /) Return value*self	Regresa la multiplicación del valor por la izquierda de la lista (a diferencia de mul que lo hace por la derecha)	>>>[1,2,3].__rmul__(3) [1,2,3,1,2,3,1,2,3] >>>3*[1,2,3] [1,2,3,1,2,3,1,2,3]
__setitem__ (self,key,value,/) Set self[key] to value		
__sizeof__(self,/) Return the size of the list in memory, in bytes.	Regresa los bytes de memoria de la lista.	>>>[1,2,3].__sizeof__() 104
append(self,object,/) append object to the end of the list	Agrega un elemento a una lista.	>>>aux = [1,2] >>>aux.append(5) >>>aux [1,2,5]
clear(self,/) Remove all items from list	Elimina todos los elementos de una lista.	>>>aux=[0,1,2] >>>aux.clear() >>>aux []

copy(self, /)
Return a shallow copy of the list.
	Devuelve una copia superficial de la lista.	>>> aux=[1,2,3]
		>>> aux.copy()
		[1, 2, 3]

count(self, value, /)
Return number of occurrences of value.
	Cuenta el número de apariciones de un valor.	>>> aux=[1,2,3,1]
		>>> aux.count(1)
		2

extend(self, iterable, /)
Extend list by appending elements from the iterable.
	Extiende la lista añadiendo elementos del argumento iterable. Itera sobre el argumento y luego añade cada elemento a la lista. El argumento dado debe ser de tipo iterable	>>> aux=[1,2,3]
		>>> aux.extend([4,5])
		>>> aux
		[1, 2, 3, 4, 5]

index(self, value, start=0,
stop=9223372036854775807, /)
Return first index of value.
Raises ValueError if the value is not present.
	Devuelve el primer índice de la lista que contenga el valor asignado.
	Devuelve ValueError si el valor no está en la lista.
		>>> aux=[10,20,33]
		>>> aux.index(20)
		1
		>>> aux.index(40)
		Traceback (most recent call last):
		File "<stdin>", line 1, in <module>
		ValueError: 40 is not in list

insert(self, index, object, /)
Insert object before index.
	Inserta un objeto antes del índice indicado.	>>> aux=[10,30,40]
		>>> aux.insert(1,20)
		>>> aux
		[10, 20, 30, 40]

pop(self, index=-1, /)
Remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.
	Quita y devuelve el elemento que se encuentra en el índice indicado.
	Genera IndexError si la lista está vacía o el índice está fuera del rango.
		>>> aux=[10,20,30]
		>>> aux.pop(2)
		30
		>>> aux
		[10, 20]

remove(self, value, /)
Remove first occurrence of value.
Raises ValueError if the value is not present.
	Remueve la primera aparición del valor que se indica.
	Devuelve ValueError si el valor no está presente.
		>>> aux=[10,20,30,10]
		>>> aux.remove(10)
		>>> aux
		[20, 30, 10]
		
		>>> aux.remove(40)
		Traceback (most recent call last):
		File "<stdin>", line 1, in <module>
		ValueError: list.remove(x): x not in list

reverse(self, /)
Reverse *IN PLACE*.
	Cambia la lista al reverso.	>>> aux=[10,20,30,40]
		>>> aux.reverse()
		>>> aux
		[40, 30, 20, 10]

sort(self, /, *, key=None, reverse=False)
Sort the list in ascending order and return None.
The sort is in-place (i.e. the list itself is modified) and stable (i.e. the order of two equal elements is maintained).
If a key function is given, apply it once to each list item and sort them, ascending or descending, according to their function values.
The reverse flag can be set to sort in descending order.
	Ordene la lista en orden ascendente y devuelva Ninguno.
	Se modifica la lista en sí y se mantiene el orden de dos elementos iguales.
	
	El indicador inverso se puede configurar para ordenar en orden descendente.
		>>> aux=[20,40,10,30,20]
		>>> aux.sort()
		>>> aux
		[10, 20, 20, 30, 40]
		
		>>> aux=[20,40,10,30,20]
		>>> aux.sort(reverse=True)
		>>> aux
		[40, 30, 20, 20, 10]

Class methods defined here: __class_getitem__(...) from builtins.type See PEP 585		
Static methods defined here: __ new__(*args, **kwargs) from builtins.type Create and return a new object. See help(type) for accurate signature.		
Data and other attributes defined here: __hash__ = None		