TOTAL Python

lists

Lists are ordered sequences of objects. These objects can be data of any type: strings, integers, floats, booleans, lists, among others. They are mutable data types.

mutable ✓ ordered ✓ allow duplicates ✓

```
list_1 = ["C", "C++", "Python", "Java"]
list_2 = ["PHP", "SQL", "Visual Basic"]
```

indexing: we can access the elements of a list through their indices [start:end:step]

```
print(list_1[1:3])
>> ["C++", "Python"]
```

item count: through property len( )

```
print(len(list_1))
>> 4
```

concatenation: we add the elements of several lists with the + symbol

```
print(list_1 + list_2)
>> ['C', 'C++', 'Python', 'Java', 'PHP', 'SQL', 'Visual Basic']
```

lists

TOTAL Python

```
list_1 = ["C", "C++", "Python", "Java"]
list_2 = ["PHP", "SQL", "Visual Basic"]
list_3 = ["d", "a", "c", "b", "e"]
list_4 = [5, 4, 7, 1, 9]
```

append( ) function: add an element to a list *in place*

```
list_1.append("R")
print(list_1)
>> ["C", "C++", "Python", "Java", "R"]
```

pop( ) function: removes an element from the list given its index, and returns the value removed

```
print(list_1.pop(4))
>> "R"
```

sort( ) function: sort list items *in place*

```
list_3.sort()
print(list_3)
>> ['a', 'b', 'c', 'd', 'e']
```

reverse( ) function: reverses the order of elements *in place*

```
list_4.reverse()
print(list_4)
>> [9, 1, 7, 4, 5]
```

↳ `reverse` is not the opposite of `sort`