```
items = ["A", "B", "C"]
items[0] = "D"
```

After this code executes, what are the values in the list `items`?

A. `"A", "B", "C"`
B. `"A", "B", "C", "D"`
C. `"D", "A", "B", "C"`
D. `"D", "B", "C"`
E. `"D"`

Answer: D
The second statement reassigns the value at index 0. It does not change the length of the list. Note that you can't perform the same operation on strings. Because strings are immutable, strings do not support item assignment (i.e., assigning to specific indices/characters).

```
a = [2, 4, 6, 8]
a.remove(4)
a.pop(2)
```

What is the value of a after the above code executes

A. [2, 4]
B. [6, 8]
C. [2, 6]
D. [2, 8]

Answer: C
After a.remove(4), a is [2, 6, 8], after a.pop(2), the item at index 2 is removed, so a is [2,6]

```
a = [2, 4, 6, 8]
a.pop(2)
a.remove(4)
```

What is the value of a after the above code executes

A. [2, 4]
B. [6, 8]
C. [2, 6]
D. [2, 8]

Answer: D
After a.pop(2), the item at index 2 is removed, so a is [2,4,8] and a.remove(4), a is [2,8]

```
for i in list(range(2,4)):
    print(i)
```

What does this code print?

A. 2 3	B. 2 3 4
C. [2, 3]	D. [2, 3, 4]

Answer: A
`list(range(2,4))` creates a list [2, 3]. When we use that as the loop sequence, each iteration will set i to the next value in the list, i.e., 2 and then 3, printing out those scalar values.