**Q1**

my\_str = "Hello"

my\_tuple = (123, 45, 1, 25)

my\_list = [123, 45, 1, 25]

# For strings

i = 0

while i < len(my\_str):

print(my\_str[i])

i += 1

# For tuples/lists (same syntax)

i = 0

while i < len(my\_tuple):

print(my\_tuple[i])

i += 1

**Q2**

a = [1000, 2000, 3000]

b = a # Shares memory with 'a'

c = a.copy() # New list

b[0] = 3 # Modifies 'a' (since b = a)

c[1] = 5 # Only modifies 'c' (independent copy)

**Q3**

# Case 1: += (in-place)

a = [1, 2, 3]

b = a

a += [2] # Modifies 'a' in-place; affects 'b'

print(b) # Output: [1, 2, 3, 2]

# Case 2: + (rebinding)

a = [1, 2, 3]

b = a

a = a + [2] # Creates new list; 'b' unchanged

print(b) # Output: [1, 2, 3]

**Q4**

def foo(a):

a = [4, 2, 6] # Reassigns local 'a'; no effect on global 'a'

return a

a = [1, 2, 3]

a = foo(a) # Global 'a' now points to returned list

print(a) # Output: [4, 2, 6]