Exercise 5.1

Rewrite the function print_n from the previous experiment, using iteration instead of recursion.

In [1]:
```
# Answer for Exercise 5.1

def print_n(s, n):
    for i in range(n):
        print(s)

# Example usage
print_n("Hello", 3)
```

```
Hello
Hello
Hello
```

Exercise 5.2

Write a function called eval_loop that iteratively prompts the user, takes the resulting input and evaluates it using eval, and prints the result. It should continue until the user enters 'done', and then return the value of the last expression it evaluated.

In [2]:
```
# Answer for Exercise 5.2

def eval_loop():
    last_result = None
    while True:
        user_input = input("Enter an expression (or 'done' to exit): ")
        if user_input.lower() == 'done':
            return last_result
        try:
            result = eval(user_input)
            print(result)
            last_result = result
        except Exception as e:
            print(f"Error: {e}")

# Example usage
print("Final result:", eval_loop())
```

```
Error: name 'hi' is not defined
Error: name 'hello' is not defined
Final result: None
```

Exercise 5.3

Write a function that takes a string as an argument and displays the letters backward, one per line.

In [3]:
```
# Answer for Exercise 5.3

```

```
def display_backwards(string):
    for char in reversed(string):
        print(char)

# Example usage
display_backwards("Hello")
```

```
o
l
l
e
H
```

Exercise 5.4

A string slice can take a third index that specifies the "step size"; that is, the number of spaces between successive characters. A step size of 2 means every other character; 3 means every third, etc.

In [5]:
```
# Answer for Exercise 5.4

# Example usage of string slicing with step size
string = "bananasandapples"

# Every other character
print(string[::2])

# Every third character
print(string[::3])

# Reverse the string
print(string[::-1])

# Every other character, starting from the second character
print(string[1::2])
```

```
bnnsnape
basdps
selppadnasananab
aaaadpls
```

Exercise 5.5

The following functions are all intended to check whether a string contains any lowercase letters, but at least some of them are wrong. For each function, describe what the function actually does (assuming that the parameter is a string).

In [6]:
```
# Function 1

def any_lowercase1(s):
    for c in s:
        if c.islower():
            return True
        else:
            return False
```

Explanation for Function 1

This function checks only the first character of the string. If the first character is lowercase, it returns True. If the first character is not lowercase, it returns False. It does not check the rest of the string.

In [8]:
```python
# Function 2

def any_lowercase2(s):
    for c in s:
        if 'c'.islower():
            return 'True'
        else:
            return 'False'
```

Explanation for Function 2

This function always returns the string 'True' for any input. The condition 'c'.islower() is always True because it's checking if the literal character 'c' is lowercase, which it always is. It doesn't actually check any characters in the input string.

In [9]:
```python
# Function 3

def any_lowercase3(s):
    for c in s:
        flag = c.islower()
    return flag
```

Explanation for Function 3

This function checks if the last character of the string is lowercase. It iterates through the entire string, updating the flag each time, but only the status of the last character determines the return value.

In [10]:
```python
# Function 4

def any_lowercase4(s):
    flag = False
    for c in s:
        flag = flag or c.islower()
    return flag
```

Explanation for Function 4

This function correctly checks if the string contains any lowercase letters. It initializes a flag to False and then uses the OR operation to update the flag. If any character is lowercase, the flag becomes True and stays True.

In [11]:
```python
# Function 5

def any_lowercase5(s):
    for c in s:
        if not c.islower():
            return False
    return True
```

Explanation for Function 5

This function checks if all characters in the string are lowercase. It returns False as soon as it encounters any non-lowercase character. It only returns True if it has checked all characters and found them all to be lowercase.