Loops in GoLang

Iteration and Repetition

Looking at various looping constructs in Go as compared to Python and C++

For Loops in Go

```
for i := 0; i < 5; i++ {
    fmt.Println(i)
}
```

- Use `for` loops for fixed iterations.
- Print numbers from 0 to 4.

For Loops in Python

```
for i in range(5):
    print(i)
```

- Python's `for` loop with `range()` for iteration.

For Loops in C++

```
for (int i = 0; i < 5; i++) {
    cout << i << endl;
}
```

- C++ `for` loop for similar behavior.

While Loops in Go

```
count := 0
for count < 5 {
    fmt.Println(count)
    count++
}
```

- Use `for` as a `while` loop in Go.

While Loops in Python

```
count = 0
while count < 5:
    print(count)
    count += 1
```

- Python's `while` loop for condition-based iteration.

While Loops in C++

```
int count = 0;
while (count < 5) {
    cout << count << endl;
    count++;
}
```

- C++ `while` loop for similar behavior.

Looping Over Collections in Go

```
fruits := []string{"apple", "banana", "cherry"}
for _, fruit := range fruits {
    fmt.Println(fruit)
}
```

- Use `for range` to iterate over collections in Go.

Looping Over Collections in Python

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

- Python's `for` loop simplifies collection iteration.

Looping Over Collections in C++

```
vector<string> fruits = {"apple", "banana", "cherry"};
for (const string& fruit : fruits) {
    cout << fruit << endl;
}
```

- C++ `for-each` loop for collection iteration.

Infinite Loops in Go

```
for {
    fmt.Println("This is infinite")
    // Break or return to exit
}
```

- Use `for` without a condition for infinite loops.

Infinite Loops in Python

```
while True:
    print("This is infinite")
    # Break or return to exit
```

- Python's `while True` for infinite loops.

Infinite Loops in C++

```
while (true) {
    cout << "This is infinite" << endl;
    // Break or return to exit
}
```

- C++ `while (true)` for infinite loops.

Summary

- **GoLang**, **Python**, and **C++** offer versatile loop constructs.
- `for`, `while`, and `for-each` variations.
- Be cautious with infinite loops.