SwiftLang Language Tutorial

1. Introduction

SwiftLang is a small, educational language. It is designed so you can see the classic phases of an interpreter or compiler in action:

- Tokenizer (lexical analysis)
- Parser (syntax)
- Semantic analyzer (basic type and variable checks)
- Interpreter (execution)

This tutorial walks through a complete example program and then gives you an exercise you can try on your own.

2. A First SwiftLang Program

Here is a SwiftLang program that compares two numbers, prints what it's doing, and then reports which value wins (or if they are equal):

```
let a = 10;
let b = 7;

print("Starting comparison...");
print(a);
print(b);

if (a > b) {
   print("a is larger:");
   print(a);
} else {
   if (a == b) {
      print("a and b are equal:");
      print(a);
   } else {
      print("b is larger:");
      print(b);
   }
}
```

Save this as examples/max_example.sl and run it from the project root with:

python -m src.main examples/max_example.sl

You should see output similar to:

```
Running SwiftLang program: examples/max_example.sl
------------------------------------------------
Starting comparison...
10
7
a is larger:
10
------------------------------------------------
Program finished successfully.
```

---

3. Step-by-Step Explanation

Lines 1–2: variable declarations

```
let a = 10;
let b = 7;
```

- let declares a new variable.
- a and b are identifiers (variable names).
- 10 and 7 are integer literals.
- After these lines:
  - a has type integer and value 10.
  - b has type integer and value 7.

Lines 4–6: initial prints

```
print("Starting comparison...");
print(a);
print(b);
```

- print("Starting comparison..."); outputs a simple status message.
- print(a); prints the current value of a.
- print(b); prints the current value of b.

This shows you both inputs before the program decides which one is “bigger”.

Lines 8–10: main if condition

```
if (a > b) {
   print("a is larger:");
   print(a);
```

} else {

- The condition (a > b) compares the two values.
- If it is true, the interpreter executes the first block:
    - It prints the message "a is larger:".
    - Then it prints the actual value of a.

In our example, 10 > 7, so this block runs.

Lines 10–16: equality and b-larger cases

```
} else {
   if (a == b) {
      print("a and b are equal:");
      print(a);
   } else {
      print("b is larger:");
      print(b);
   }
}
```

This is a nested if inside the else:

- If we *didn't* have a > b, we check a == b:
    - If a == b is true, we print that they're equal and show the value.
- Otherwise (the final else), we know that b must be larger:
    - We print "b is larger:" and then the value of b.

So the program covers three cases:

1. a > b → prints "a is larger:" and a.
2. a == b → prints "a and b are equal:" and a.
3. Otherwise → prints "b is larger:" and b.

4. Using a While Loop

Here is another example that uses a while loop to count from 0 up to 2:

```
let i = 0;
```

```
while (i < 3) {
   print(i);
   i = i + 1;
}
```

- The condition i < 3 is checked before each iteration.
- If it is true, the body runs:
    - print(i); outputs the current value.
    - i = i + 1; updates the value of i.
- When i becomes 3, the condition is false and the loop ends.

The output is:

```
0
1
2
```

---

5. Example Exercise

Goal: Write a SwiftLang program that:

1. Declares three integer variables: x, y, and z.
2. Assigns values to them so that x is the smallest, y is in the middle, and z is the largest.
3. Uses if / else and while to add 1 to x repeatedly until x is equal to y.
4. Prints all three values at the end.

Try writing this program in your own .sl file and then run it with:

```
python -m src.main path/to/your_file.sl
```

---

6. Sample Solution (One Possible Answer)

Here is one way to solve the exercise:

```
let x = 1;
let y = 4;
let z = 10;

while (x < y) {
   x = x + 1;
}

print(x);
print(y);
```

print(z);

- After the loop finishes, x will be 4, so the output will be:

4
4
10

You can experiment by changing the starting values or the loop condition to see how the behavior changes. This is a good way to get comfortable with SwiftLang's variables, conditions, and loops.