

## CS330: Programming Language Project (PLP)

### Assignment 4: Control flow

Use Swift to write:

- a one-condition if/else statement

```
1 var count = 1
2
3 ▼ if (count%3==0) {
4     print("Fizz")}
5 ▼ else {
6     print ("No")}
7 ▲ }
```

[dhcp-92-176:De  
[dhcp-92-176:De  
No  
dhcp-92-176:De

OUTPUT

- a multi-condition if/else statement

```
1 var count = 1
2
3 ▼ if count%3==0 && count%5==0{
4     print("FizzBuzz")}
5 ▼ else {
6     print ("No")}
7
```

No  
[dhcp-  
No  
dhcp-

OUTPUT

- for loop

```
1 let courses = ["MATH-101", "MATH-102", "MATH-103"]
2 ▼ for course in courses {
3     print(course)}
4
```

[dhcp-92-176:De  
MATH-101  
MATH-102  
MATH-103  
dhcp-92-176:De

OUTPUT

```
1 let courses = ["MATH-101":25, "MATH-102":23, "MATH-103":35]
2 ▼ for (course, numofstudents) in courses {
3     print("\(course) has \(numofstudents) students.")}
4
```

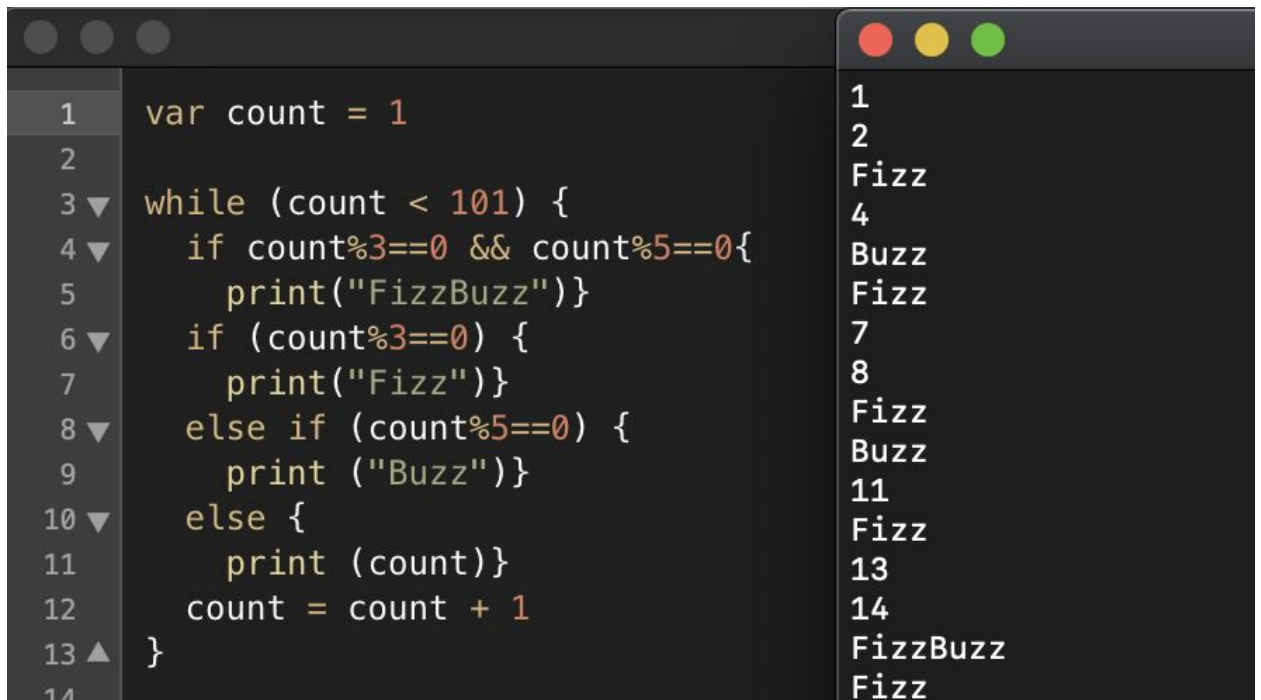
[dhcp-92-176:Desktop melata  
MATH-101 has 25 students.  
MATH-103 has 35 students.  
MATH-102 has 23 students.  
dhcp-92-176:Desktop melata

```
1 ▼ for number in (0...10) {
2     print("\(number)")}
3
4
5
6
7
8
9
10
11
```

[dhcp-92-176:Desktop  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

- *while loop*

OUTPUT<sup>1</sup>

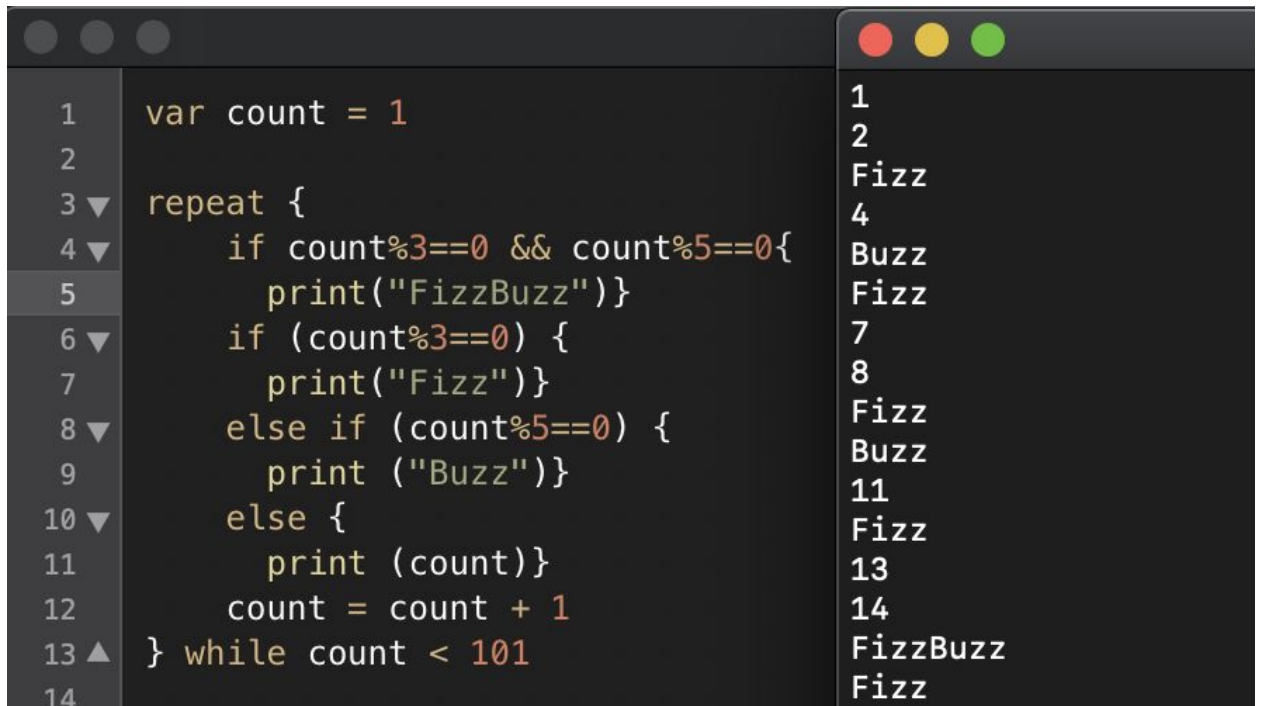


```
1 var count = 1
2
3 while (count < 101) {
4     if count%3==0 && count%5==0{
5         print("FizzBuzz")}
6     if (count%3==0) {
7         print("Fizz")}
8     else if (count%5==0) {
9         print ("Buzz")}
10    else {
11        print (count)}
12    count = count + 1
13 }
14
```

1  
2  
Fizz  
4  
Buzz  
Fizz  
7  
8  
Fizz  
Buzz  
11  
Fizz  
13  
14  
FizzBuzz  
Fizz

- *repeat/while loop*

OUTPUT<sup>2</sup>



```
1 var count = 1
2
3 repeat {
4     if count%3==0 && count%5==0{
5         print("FizzBuzz")}
6     if (count%3==0) {
7         print("Fizz")}
8     else if (count%5==0) {
9         print ("Buzz")}
10    else {
11        print (count)}
12    count = count + 1
13 } while count < 101
14
```

1  
2  
Fizz  
4  
Buzz  
Fizz  
7  
8  
Fizz  
Buzz  
11  
Fizz  
13  
14  
FizzBuzz  
Fizz

<sup>1</sup> Van der Lee, Antoine. "How to Use for Loop, for Each, While, and Repeat in Swift (in-Depth)." *SwiftLee*, 13 Dec. 2019, [www.avanderlee.com/swift/loops-swift/](http://www.avanderlee.com/swift/loops-swift/).

<sup>2</sup> "Swift - Do...while Loop." *Tutorialspoint*, [www.tutorialspoint.com/swift/swift\\_do\\_while\\_loop.htm](http://www.tutorialspoint.com/swift/swift_do_while_loop.htm).

- *foreach loop*

OUTPUT

```
1 let courses = ["MATH-101", "MATH-102", "MATH-103"]
2 courses.forEach { course in
3     print(course)}
4
```

MATH-101  
MATH-102  
MATH-103  
dhcp-92-176:Desktop

- *a switch-case statement*

OUTPUT<sup>3</sup>

```
1 let coursestaken = 4
2
3 switch coursestaken {
4     case 0:
5         print("You must register for your courses before 9/1.")
6     case 1:
7         print("You must register for three more courses.")
8     case 2:
9         print("You must register for two more courses.")
10    case 3:
11        print("You must register for one more course.")
12    case 4:
13        print("You are all set!")
14    default:
15        print("How many classes have you registered for?")
16}
```

dhcp-92-176:Desktop  
You are all set!  
dhcp-92-176:Desktop

- *a break statement*

OUTPUT<sup>4</sup>

```
1 let coursestaken = 4
2
3 switch coursestaken {
4     case 0:
5         print("You need to register for your courses")
6     case 1:
7         print("You must register for three more courses.")
8     case 2:
9         print("You must register for two more courses.")
10    case 3:
11        print("You must register for one more course.")
12    case 4:
13        break;
14    default:
15        print("How many classes have you registered for?")
16}
```

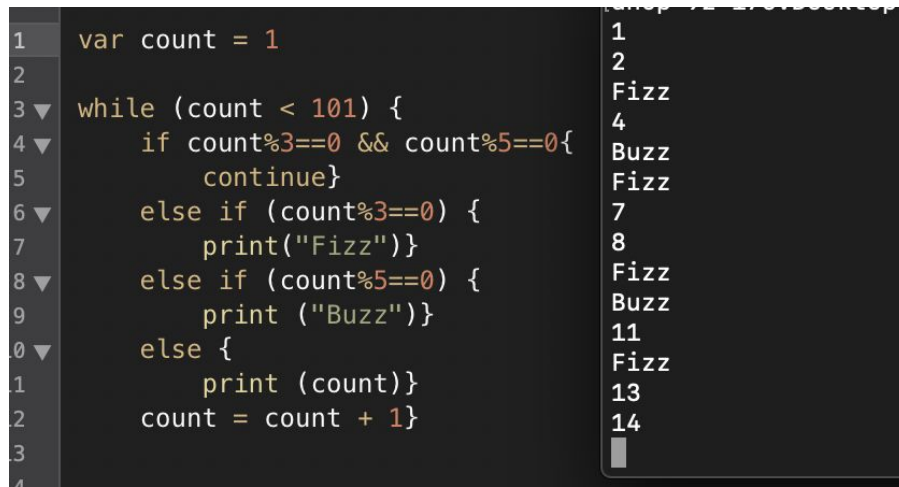
dhcp-92-176:Desktop  
You are all set!  
dhcp-92-176:Desktop

<sup>3</sup> Hudson, Paul. "Switch Case." *Hacking with Swift*, Hacking with Swift, 18 Apr. 2020, [www.hackingwithswift.com/read/0/10/switch-case](http://www.hackingwithswift.com/read/0/10/switch-case).

<sup>4</sup> "Swift's Break and Continue Statements." *AndyBargh.com*, 18 July 2019, [andybargh.com/break-and-continue/](http://andybargh.com/break-and-continue/).

- a *continue* statement

OUTPUT



```
1 var count = 1
2
3 while (count < 101) {
4     if count%3==0 && count%5==0{
5         continue}
6     else if (count%3==0) {
7         print("Fizz")}
8     else if (count%5==0) {
9         print ("Buzz")}
10    else {
11        print (count)}
12    count = count + 1}
13
14
```

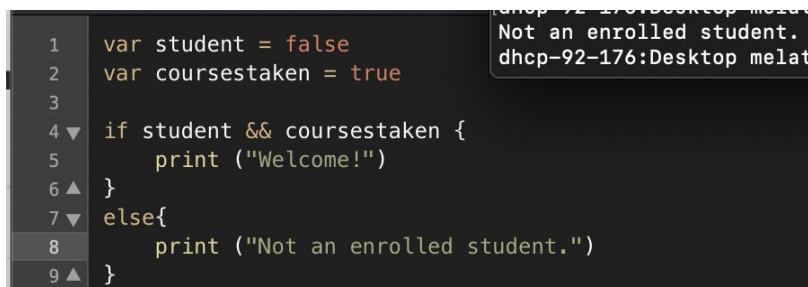
The screenshot shows a Swift code editor on the left and its output on the right. The code is a while loop that prints numbers from 1 to 100. It uses a `continue` statement to skip the `print` statement when the count is divisible by both 3 and 5. The output on the right shows the sequence of numbers printed, with 'Fizz' and 'Buzz' appearing at the appropriate intervals.

**1. What types of conditional statements are available in your language? (if/else, if/then/else, if/elseif/else). Does it allow for statements other than “if” (for example, Perl has an “unless” statement, which does the opposite of “if”!)**

Swift only has if/else conditional statements.<sup>5</sup>

**2. Does your language use short-circuit evaluation? If so, make sure that your code includes an example.**

Swift does use short-circuit evaluation, which is “when a statement stops evaluation as soon as an outcome is determined.”<sup>6</sup> An example is:



```
1 var student = false
2 var coursestaken = true
3
4 if student && coursestaken {
5     print ("Welcome!")
6 }
7 else{
8     print ("Not an enrolled student.")
9 }
```

The screenshot shows a Swift code editor on the left and its output on the right. The code is an if statement that checks if a student is enrolled and if a course has been taken. Since the first condition (`student`) is false, the second condition (`coursestaken`) is not evaluated. The output on the right shows "Not an enrolled student."

Since the first value is false, it doesn’t even check the second value. It automatically moves on to the else statement.<sup>7</sup>

<sup>5</sup> Inc., Apple. “Swift.org.” *Statements - The Swift Programming Language (Swift 5.3)*, docs.swift.org/swift-book/ReferenceManual/Statements.html.

<sup>6</sup> Goldsby, Chris. “Swift: Short-Circuit Evaluations.” *Medium*, Medium, 26 Mar. 2016, medium.com/@cgoldsby/swift-short-circuit-evaluation-28995775fe21.

<sup>7</sup> Inc., Apple. “Swift.org.” *Basic Operators - The Swift Programming Language (Swift 5.3)*, docs.swift.org/swift-book/LanguageGuide/BasicOperators.html.

### 3. How does your programming language deal with the “dangling else” problem?

The dangling else problem occurs when an if statement is followed by another if statement and an else statement, and it becomes unclear whether the else should be nested under the first if statement or not. For example:

```
if (statement):  
    (do this)  
if (statement2):  
    (do this)  
else (statement3):  
    (do this)
```

In this case, the lack of indentation could have been intentional or it could have been a mistake and the programmer may have actually meant:

```
if (statement):  
    (do this)  
        if (statement2):  
            (do this)  
        else (statement3):  
            (do this)
```

OR

```
if (statement):  
    (do this)  
        if (statement2):  
            (do this)  
else (statement3):  
    (do this)8
```

However, Swift **requires** the use of parentheses {} after each if statement, which fixes this issue since you can enclose the else in the first if statement's parentheses or not depending on how you want the code to run, and indentation won't matter.<sup>9</sup>

---

<sup>8</sup> “Dangling Else, Yet Again.” Dr. Dobb's, 22 Sept. 2011, [www.drdobbs.com/cpp/dangling-else-yet-again/231602010](http://www.drdobbs.com/cpp/dangling-else-yet-again/231602010).

<sup>9</sup> “Swift Fixes C's 'Issues.'” *The Craft of Coding*, 26 May 2017, [craftofcoding.wordpress.com/2017/05/26/swift-fixes-cs-issues/](http://craftofcoding.wordpress.com/2017/05/26/swift-fixes-cs-issues/).

**4. Does your language include multiple types of loops (while, do/while, for, foreach)? If so, what are they and how do they differ from each other?**

Yes, Swift includes all the mentioned loops, with the exception that the do/while loop is called the repeat/while loop instead. I've included examples above, and the main difference lies in the formatting of each, but the bulk of the code remains the same, especially with the for and foreach loops and the while and repeat/while loops respectively. You can pick the most suitable type of loop depending on what exactly you're trying to accomplish with your code.

**5. Can you use break or continue statements (or something similar) to exit loops?**

As shown in the examples above, a break statement can be used to exit a loop, but a continue statement does not exit the loop.

**6. If your language supports switch or case statements, do you have to use "break" to get out of them? Can you use "continue" to have all of them evaluated?**

In the case of a switch statement, as long as you set a default value that the program can output if none of the cases match, it will run the code then exit the program.

Continue statements are only allowed in loops and not in switch statements. As soon as there is a matching case in a switch statement, it only executes the code for that match then exits.

**7. Is there anything special in terms of control flow that your language does that isn't addressed in this assignment? If so, what is it and how does it work?**

The previous question asks if we can use continue to evaluate all cases in a switch statement and while we can't do that, we can use another statement called a fallthrough to do something similar. Fallthrough stops the program from exiting as soon as it executes the code under a match, but, unfortunately, instead of evaluating the next case, it just automatically executes the code for it.

---

```
let coursestaken = 0
switch coursestaken {
  case 0:
    print("No")
    fallthrough
  case 1:
    print("You must register for three more courses.")
  case 2:
    print("You must register for two more courses.")
  case 3:
    print("You must register for one more course.")
  case 4:
    break;
  default:
    print("Done.")]
}

print("You are all set!")
```

```
dhcp-92-176:Desktop melatassefa$ swift 112
No
You must register for three more courses.
You are all set!
dhcp-92-176:Desktop melatassefa$
```

It evaluated the first case, found it was true, and executed the code which told it to print out "No". Then it fell through to the next case and printed out "You must register for three more courses." even though the case didn't match.