Loop statements

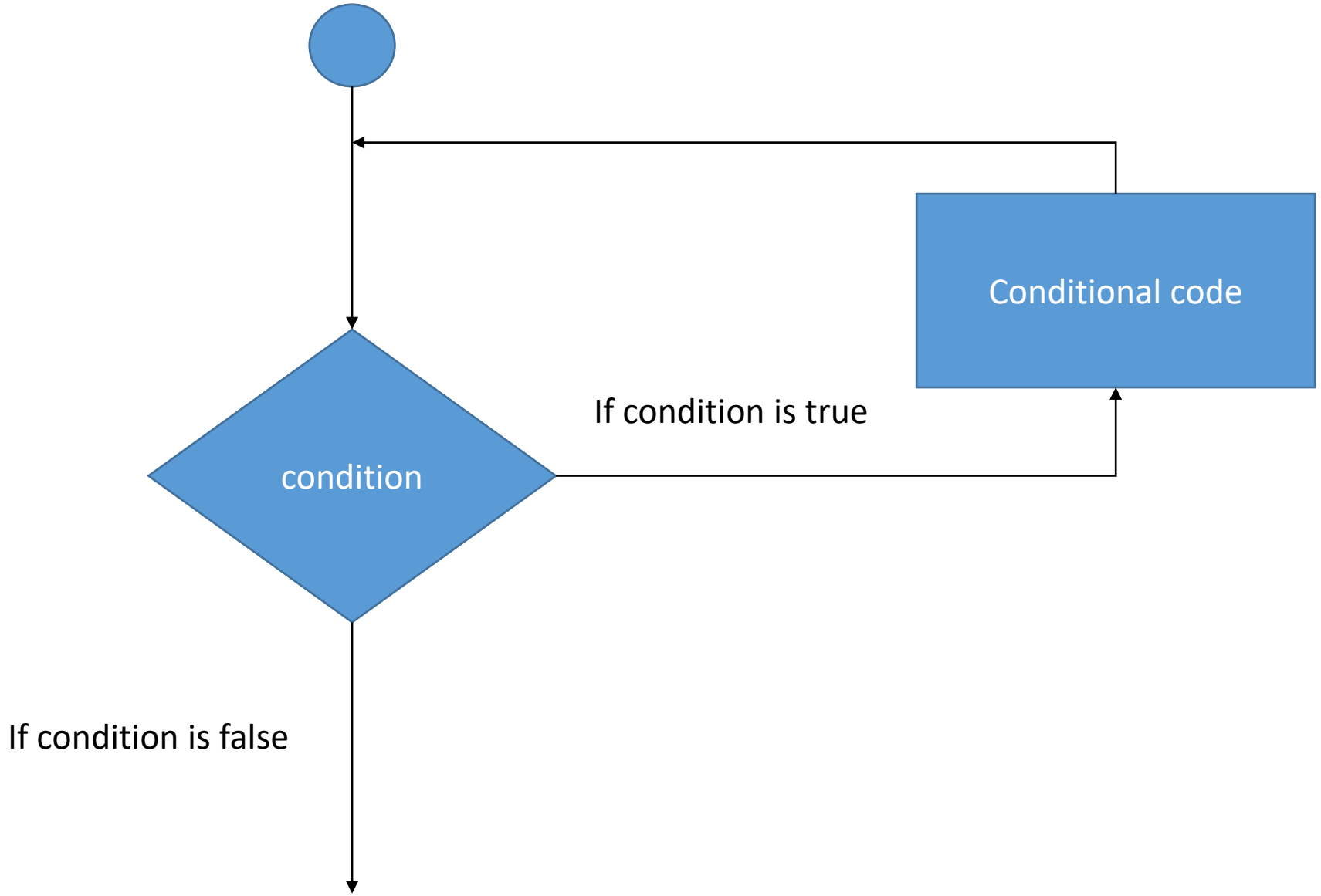
Lecture four

practical

Loop statements

- In general, statements are executed sequentially: the first statement in a function is executed first, followed by the second, and so on.
- There may be a situation when you need to execute a block of code several number of times.
- Programming languages provide various control structures that allow for more complicated execution paths.
- A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement

Flow diagram of loop statement



Loop statements

- Python programming language provides following types of loops to handle looping requirements.

1. while loop: repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2. for loop: executes thatExecutes a sequence of statements multiple times and abbreviates the code thatmanages manages the loop variable.
3. nested loops: you can use one or more loop inside any another while, for loop.

Loop Control Statements

- Loop control statements change execution from its normal sequence.
- When execution leaves scope, all automatic objects that were created in that scope are destroyed.
- Python supports the following control statements.

1. Break statement: terminates the loop statement and transfers execution to the statement immediately following the loop.
2. continue statement: causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3. pass statement: the pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

while Loop Statement

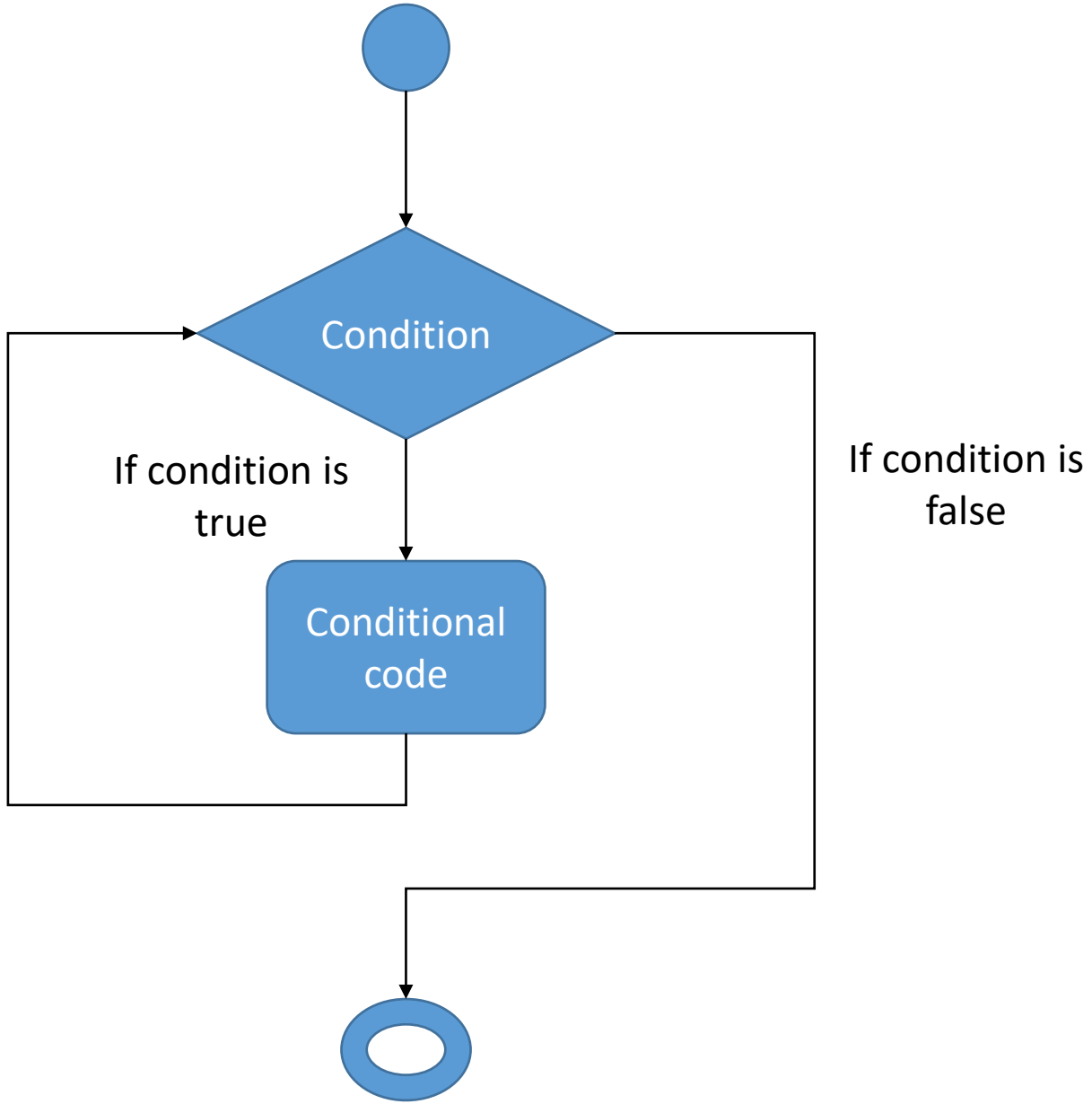
- A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.
- The syntax of a while loop in Python programming language is

```
while expression:
        statement(s)
```

- Here, statement(s) may be a single statement or a block of statements.
- The condition be may any expression, and true is any non-zero value.
- The loop iterates while the condition is true.
- When the condition becomes false, program control passes to the line immediately following the loop.
- In Python, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code.
- Python uses indentation as its method of grouping statements.

Flow diagram of while loop

- Here, key point of the while loop is that the loop might not ever run.
- When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.



while loop example

```
count=0
while (count<9):
   print('The count is: ', count)
   count=count+1
print("Good bye!")
```

- When the above code is executed, it produces the following result –

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

The Infinite Loop

- A loop becomes infinite loop if a condition never becomes FALSE.
- You must use caution when using while loops because of the possibility that this condition never resolves to a FALSE value. This results in a loop that never ends. Such a loop is called an infinite loop.
- An infinite loop might be useful in client/server programming where the server needs to run continuously so that client programs can communicate with it as when required.

Infinite loop example

```
var=1
while var==1:
   num= input("Enter a number :")
   print("You entered: ",num)
print("Good bye!")
```

- When the above code is executed, it produces the following result
- Enter a number :12
- You entered: 12
- Enter a number :4
- You entered: 4
- Enter a number :5
- You entered: 5
- Enter a number :6
- You entered: 6

Using else Statement with While Loop

- Python supports to have an else statement associated with a loop statement.
- If the else statement is used with a while loop, the else statement is executed when the condition becomes false.
- The following example illustrates the combination of an else statement with a while statement that prints a number as long as it is less than 5, otherwise else statement gets executed.

else Statement with While Loop example

```
count=0
while count<5:
    print(count," is less than 5")
    count=count+1
else:
    print(count," is not less than 5")
```

- When the above code is executed, it produces the following result:

```
0  is less than 5
1  is less than 5
2  is less than 5
3  is less than 5
4  is less than 5
5  is not less than 5
```