\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC121 PYTHON Programming**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lesson 03 Selection Control Structures

# **Objectives**

In this lesson, students will learn:

- How to use selection control structures in Python programs

- How to write if statements

- How to use elif headers

- How to write nested if statements

# **3.1 Overview**

We write programs to solve problems. In addition to calculations, some programs also need other types of logical operations. We learned how to write assignment statements to perform calculations. In the coming lessons, we are going to write code to perform logical processing.

The type of logical processing we are learning in this lesson is **selection control structure**. We have a set of statements in the program that the computer may or may not execute. The decision is made during run time. If a pre-defined condition is satisfied, the computer will execute these statements. Otherwise, the computer will not execute these statements and may execute an alternative set of statements. For example, suppose students will get 10 extra points if they answer a bonus question in a test correctly. In a program, we can have statements to award extra credit. The computer will execute these statements only if this condition is satisfied, i.e. if the bonus question is answered correctly. If this condition is not satisfied, the computer will not execute the statements to award extra credit. In Python, we use an if statement to code this type of logical processing.

# **3.2 Writing Conditions**

Before we see how to write if statements, let’s talk about conditions first.

A **condition** is a **Boolean expression**. It is an expression the computer can test whether it is true or false.

Suppose in a computer game a player moves to the next level if score exceeds 5000. Here the condition for moving to the next level is “score exceeds 5000”. This condition is a Boolean expression because it is something either true or false.

The example above compares two values, score and 5000, to see whether the former is greater than the latter. In Python, a few **relational operators** are used to compare values:

|  |  |
| --- | --- |
| Comparison | Python syntax |
| x is equal to y | x == y |
| x is not equal to y | x != y |
| x is greater than y | x > y |
| x is smaller than y | x < y |
| x is greater than or equal to y | x >= y |
| x is smaller than or equal to y | x <= y |

Let’s look at a few examples.

Suppose the Powerball drawn in the lottery last night was the number 8. Anybody who picked this number wins $10. The condition for winning $10 is numberPicked == 8. If this condition is true, the person wins $10. If this condition is false, the person does not win $10.

We can express the same logic in a different way. The condition for not winning $10 is numberPicked != 8. If this condition is true, the person does not win $10. If this condition is false, the person wins $10.

Let’s look at another example. A person can drink alcohol when he reaches age 21. So the condition for drinking alcohol is age >= 21. If this condition is true, the person can drink alcohol. If this condition is false, the person cannot drink alcohol.

We can express the same logic in a different way. The condition for not able to drink alcohol is age < 21. If this condition is true, the person cannot drink alcohol. If this condition is false, the person can drink alcohol.

Here is another example. A patient does not have fever if body temperature is 100 degree or less. So the condition for not having a fever is temperature <= 100. If this condition is true, the patient does not have a fever. If this condition is false, the patient has a fever.

We can express the same logic in a different way. The condition for having a fever is temperature > 100. If this condition is true, the patient has a fever. If this condition is false, the patient does not have a fever.

Now we have a basic understanding of what a condition is and how to write conditions to compare values. We are going to introduce the if statement.

# **3.3 The If Statement**

In Python, the if statement allows the computer to decide whether to execute a set of statements or not while the program is running. Let’s look at an example:

age = float(input('How old are you? '))

if age >= 21:

print('You can drink alcohol legally')

The condition the computer tests is age >= 21. If this condition is true, the computer executes the print statement and displays the text You can drink alcohol legally. If the condition is false, the text will not be displayed.

Here is another example:

temperature = float(input('What is your temperature? '))

if temperature > 100:

print('You have a fever')

print('Please see a doctor')

If the condition temperature > 100 is true, the computer will display the text You have a fever and Please see a doctor. If the condition is false, the text will not be displayed.

We have seen two examples of the if statement. Let’s describe its syntax:

**if** *condition***:**

statement block to execute if the condition is true

The first line is called a **header**. It consists of the keyword if, a testable condition and a colon. After the header line we have one or more statements to execute if and only if the condition is true. These statements must be indented. If there are more than one statement, all statements must be indented the same amount. Python uses indentation to group statements. Consecutive statements with the same amount of indentation are grouped into a **suite** (also called a **block**). The block of statements execute or not as a whole. If the condition is true, the whole block of statements execute. If the condition is false, the whole block of statements do not execute. In the fever example, since the two print statements are indented the same amount, they are grouped into a block. Therefore, either you see both statements You have a fever and Please see a doctor, or you see nothing at all, depending on whether temperature exceeds 100.

What is going to happen if we unindent the second print statement?

temperature = float(input('What is your temperature? '))

if temperature > 100:

print('You have a fever')

print('Please see a doctor') # not indented

Now the second print statement is not in the block anymore. The block only consists of one statement, i.e. the statement that displays You have a fever. If temperature exceeds 100, You have a fever is displayed. If the temperature does not exceed 100, You have a fever is not displayed. What about the second print statement? Since it is not part of the block, it is unrelated to the condition. It is a regular statement that always executes. Look at the following sample test run:

What is your temperature? 98.4

Please see a doctor

You are told to see a doctor even though your temperature does not exceed 100.

We can add a second block to the if statement and tell the computer to execute this new block if the condition is false. This is done by adding the else header to the if statement. The following is an example:

age = float(input(**'How old are you? '**))  
**if** age >= 21:  
 print(**'You can drink alcohol legally'**)  
**else**:  
 print(**'You cannot drink alcohol legally'**)

The following is a sample test run:

How old are you? 17

You cannot drink alcohol legally

The computer executes the block under the else header because the condition is false.

Here is another example:

temperature = float(input(**'What is your temperature? '**))  
**if** temperature > 100:  
 print(**'You have a fever'**)  
 print(**'Please see a doctor'**)  
**else**:  
 print(**'You do not have a fever'**)

The following is a sample test run:

What is your temperature? 98.4

You do not have a fever

The else header must start at the same column as the if header. Otherwise, the Python interpreter will get confused and give you a syntax error:

temperature = float(input(**'What is your temperature? '**))  
**if** temperature > 100:  
 print(**'You have a fever'**)  
 print(**'Please see a doctor'**)  
 **else**: *# indentation error* print(**'You do not have a fever'**)

This program gets the following error message:

File "C:example.py", line 5

else: # indentation error

^

IndentationError: unindent does not match any outer indentation level

Also, do not add condition in the else header:

temperature = float(input(**'What is your temperature? '**))  
**if** temperature > 100:  
 print(**'You have a fever'**)  
 print(**'Please see a doctor'**)  
**else** temperature <= 100: *# should not have condition in else* print(**'You do not have a fever'**)

This program gets the following error message:

File "C:/example.py", line 5

else temperature <= 100:

^

SyntaxError: invalid syntax

Let’s look at another example.

An online music store sells CDs for $7.99 each and $2.99 shipping fee. If you buy 5 or more, you get 10% off plus free shipping. Write a program to calculate total charge.

num\_cd = int(input(**'How many CDs are you buying? '**))  
**if** num\_cd <= 5:  
 total\_charge = 7.99 \* num\_cd + 2.99  
**else**:  
 total\_charge = 7.99 \* num\_cd \* 0.9  
print(**"Please pay this amount: $"**, total\_charge)

First we ask the user how many CDs he is buying. If it is 5 or less, we calculate total charge by multiplying 7.99 to number of CDs and add 2.99 shipping charge to it. Otherwise, we give a 10% discount by multiplying 0.9 to the product of 7.99 times number of CDs. Finally, we use a print statement to display total charge. The following is a sample test run:

How many CDs are you buying? 4

Please pay this amount: $ 34.95

In the next example, we test a string in the condition.

Residents of Pythonville pay $5 to use its swimming pool. Non-residents pay $7.

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**else**:  
 print(**'Please pay $7'**)

The program asks the user to enter y or n to indicate whether he is a resident of Pythonville. The following is a sample test run:

Are you resident of Pythonville? [y/n] y

Please pay $ 5

The program above does not work properly if the user enters uppercase letters. In Python, you can convert a string to all uppercase letters or all lowercase. Let’s add a statement in the program to convert user input into lowercase:

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
resident = resident.lower()  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**else**:  
 print(**'Please pay $7'**)

The syntax resident.lower() returns the all lowercase version of the string stored in resident. The assignment statement stores this all lowercase version in the variable resident to replace the old string. The following is a sample test run:

Are you resident of Pythonville? [y/n] Y

Please pay $ 5

The program works properly now even if an uppercase Y is entered by the user.

You can also convert user input into all uppercase if you want to. In that case, you need to test the uppercase Y in the if statement:

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
resident = resident.upper()  
**if** resident == **'Y'**: # Test uppercase Y  
 print(**'Please pay $5'**)  
**else**:  
 print(**'Please pay $7'**)

# **3.4 The NOT Operator**

We can use the not operator to negate the result of a Boolean expression if we want to. Let’s look at an example:

age = float(input(**'How old are you? '**))  
**if not**(age >= 21):  
 print(**'You cannot drink alcohol legally'**)  
**else**:  
 print(**'You can drink alcohol legally'**)

The condition is changed to not(age >= 21), which is the exact opposite of the original condition age >= 21. This new condition is true if age is not greater than or equal to 21, and false if age is. Since the condition is changed to the opposite, we need to swap the statements in the if and else blocks accordingly. If the condition not(age >= 21) is true, the program will display You cannot drink alcohol legally. If the condition is false, the program will display You can drink alcohol legally.

# **3.5 The AND Operator**

Sometimes we want to test two conditions together to make a selection. There are two ways to combine two Boolean expressions into one. The first ways is to use the and operator. Let’s look at an example.

Suppose a community college requires students to finish 68 credit hours and have a cumulative GPA of 2.0 to graduate. If we design an if statement to determine graduation eligibility, we have to test both requirements in the condition.

hrs\_finished = float(input(**"How many credits have you finished?"**))  
gpa = float(input(**"What is your cumulative GPA? "**))  
**if** hrs\_finished >= 68 **and** gpa >= 2.0:  
 print(**"You are eligible to graduate."**)  
**else**:  
 print(**"You are not eligible to graduate."**)

When we use the and operator to combine two Boolean expressions, the combined expression is true only if both expressions are true. If only one of them is true or both of them are false, the combined expression is false.

The following is a sample test run of the program:

How many credits have you finished? 70

What is your cumulative GPA? 3.25

You are eligible to graduate.

In this test case, since both requirements are fulfilled, the combined condition is true.

Here is another test case:

How many credits have you finished? 70

What is your cumulative GPA? 1.95

You are not eligible to graduate.

Although this student has enough credit hours, he is not eligible to graduate since his GPA is too low.

# **3.6 The OR Operator**

There are situations in which satisfaction of only one of two requirements is sufficient to make the whole condition true. The following is an example.

Blood pressure is measured in two numbers: systolic pressure and diastolic pressure. Systolic should not exceed 140 and diastolic pressure should not exceed 90. If any of the two numbers exceeds its limit, the patient has high blood pressure. The following is a Python program to test whether a patient has high blood pressure.

systolic = float(input(**"What is your systolic pressure? "**))  
diastolic = float(input(**"What is your diastolic pressure? "**))  
**if** systolic > 140 **or** diastolic > 90:  
 print(**"Patient has high blood pressure"**)  
**else**:  
 print(**"Patient does not have high blood pressure"**)

When we use the or operator to combine two Boolean expressions, the combined expression is true if at least one expression is true. The combined expression is false only if both expressions are false.

The following is a sample test run of the program:

What is your systolic pressure? 120

What is your diastolic pressure? 95

Patient has high blood pressure

In this test case, since the diastolic pressure exceeds the limit, the combined condition is true even though the systolic pressure is within limit.

Here is another test case:

What is your systolic pressure? 120

What is your diastolic pressure? 80

Patient does not have high blood pressure

The combined condition is false since both numbers are within limits.

Here is another example. Prime numbers are integers divisible only by 1 and itself. 0 and 1 are not considered as prime numbers. The first several prime numbers are 2, 3, 5 and 7. Write a program to do the following. Ask the user to enters a number between 2 and 10. The program will show whether the number is prime or not. The following is the Python program:

num = int(input(**'Pick an integer from 2 to 10: '**))  
**if** num == 2 **or** num == 3 **or** num == 5 **or** num == 7:  
 print(**'It is a prime number'**)  
**else**:   
 print(**'It is not a prime number'**)

The following is a sample test run:

Pick an integer from 2 to 10: 7

It is a prime number

When you use and or or to form a combined Boolean expression, do not shorten each individual expression. The following shows a common error in the prime number program:

num = int(input(**'Pick an integer from 2 to 10: '**))  
**if** num == 2 **or** 3 **or** 5 **or** 7: *# cannot shorten expressions* print(**'It is a prime number'**)  
**else**:  
 print(**'It is not a prime number'**)

The Python interpreter does not give you a syntax error for the shortened expressions, but you may get wrong result in some cases. The combined condition is always true with shortened expressions like the ones in this example. That means the program will always say it is a prime number regardless of what number the user enters:

Pick an integer from 2 to 10: 4

It is a prime number

# **3.7 The IN Operator**

The in operator provides a short form for situations like the prime number example. That program can be rewritten like this:

num = int(input(**'Pick an integer from 2 to 10: '**))  
**if** num **in** (2, 3, 5, 7):  
 print(**'It is a prime number'**)  
**else**:  
 print(**'It is not a prime number'**)

The in operator tests whether the value of a variable is in a specified group of values. The whole expression is true if num is in the group of 2, 3, 5 and 7. It is false if num is not in that group. The following is a sample test run:

Pick an integer from 2 to 10: 7

It is a prime number

The following is another test run:

Pick an integer from 2 to 10: 4

It is not a prime number

# **3.8 The elif header**

Let’s revisit the swimming pool program we saw earlier.

Residents of Pythonville pay $5 to use its swimming pool. Non-residents pay $7.

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
resident = resident.lower()  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**else**:  
 print(**'Please pay $7'**)

The condition tests whether resident == 'y'. If it is, the user is charged $5. If it is not, the user is charge $7. That means the program will display Please pay $7 as long as resident is not the letter y.

Are you resident of Pythonville? [y/n] r

Please pay $7

Suppose we want to modify the program so it will charge $7 only if the user has entered n. How do we do it? Can we add the condition resident == 'n' in the else header?

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**else** resident == **'n'**: # Is this allowed?  
 print(**'Please pay $7'**)

This does not work because we are not allowed to add a condition in the else header:

File "C:example.py", line 4

else resident == 'n':

^

SyntaxError: invalid syntax

Then how do we test whether resident is n? The solution is using elif instead of else:

resident = input(**'Are you resident of Pythonville? [y/n] '**)

resident = resident.lower()  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**elif** resident == **'n'**:  
 print(**'Please pay $7'**)

The keyword elif means “else if”. It must be used together with the if header. The computer tests the condition in the if header first. If that condition is true, the block under the if header will execute and the elif header is ignored completely. If the condition in the if header is false, then the computer will test whether the condition in the elif header is true. The following is a sample test run:

Are you resident of Pythonville? [y/n] y

Please pay $5

The user enters y. Since the condition in the if header is true, $5 is charged and the elif header is ignored completely.

Here is another test case:

Are you resident of Pythonville? [y/n] n

Please pay $7

The user enters n. The condition in the if header is false, so the computer tests the condition in the elif header. Since it is true, $7 is charged.

Let’s test one more case:

Are you resident of Pythonville? [y/n] r

This time the user enters r. Since both conditions are false, no output is displayed.

The user may get a little confused in the third test case because the program seems to have no response. We should display an error message if the user enters nither y nor n. We can do that by adding an else header:

resident = input(**'Are you resident of Pythonville? [y/n] '**)  
**if** resident == **'y'**:  
 print(**'Please pay $5'**)  
**elif** resident == **'n'**:  
 print(**'Please pay $7'**)  
**else**:  
 print(**'Invalid input'**)

If the user enters something wrong, an error message will be displayed:

Are you resident of Pythonville? [y/n] Y

Invalid input

Let’s look at another example:

The US clothing industry divides women’s height into three categories: petite (63 in. or shorter), average (from above 63 in. up to 68 in.) and tall (taller than 68 in.). The following Python program determines a woman’s height category.

height = float(input(**'How tall are you in inches? '**))  
**if** height <= 63:  
 print(**'Height category: Petite'**)  
**elif** 63 < height <= 68:  
 print(**'Height category: Average'**)  
**else**: *# user taller than 68 since both conditions are false* print(**'Height category: Tall'**)

The following is a sample test run:

How tall are you in inches? 70

Height category: Tall

If necessary, we can add as many elif headers as we want. The following is an example.

Modify the women height category program by adding two new categories. New definition for each category:

Extra petite: 59 in. and below

Petite: above 59 in. to 63 in.

Average: above 63 in. to 68 in.

Tall: above 68 in. to 71 in.

Extra tall: above 71 in.

The following is the Python program:

height = float(input(**'How tall are you in inches? '**))  
**if** height <= 59:  
 print(**'Height category: Extra Petite'**)  
**elif** 59 < height <= 63:  
 print(**'Height category: Petite'**)  
**elif** 63 < height <= 68:  
 print(**'Height category: Average'**)  
**elif** 68 < height <= 71:  
 print(**'Height category: Tall'**)  
**else**: *# user taller than 71 since all conditions are false* print(**'Height category: Extra Tall'**)

Please make up some test cases to test this program yourself.

# **3.9 Nested If Statements**

We can put more if statements inside any if, elif or else blocks of another if statement. The if statements added inside another if statement are called nested if statements. Let’s look at some examples.

Earlier we wrote a program to tell a student whether he is eligible to graduate. Now we want to improve the program by telling the student which requirement(s) he does not meet if he is not eligible. The flowing is the original program:

hrs\_finished = float(input(**"How many credits have you finished?"**))  
gpa = float(input(**"What is your cumulative GPA? "**))  
**if** hrs\_finished >= 68 **and** gpa >= 2.0:  
 print(**"You are eligible to graduate."**)  
**else**:  
 print(**"You are not eligible to graduate."**)

How do we give explanations to the user if he is not eligible? If the user is not eligible to graduate, the program executes the else block. That means we should add code to the else block to test the two requirements again separately and display message whenever a requirement is not fulfilled. Two if statements are added to the else block:

hrs\_finished = float(input(**"How many credits have you finished?"**))  
gpa = float(input(**"What is your cumulative GPA? "**))  
**if** hrs\_finished >= 68 **and** gpa >= 2.0:  
 print(**"You are eligible to graduate."**)  
**else**:  
 print(**"You are not eligible to graduate."**)  
 **if** hrs\_finished < 68:  
 print(**"You need 68 credit hours or more."**)  
 **if** gpa < 2.0:  
 print(**"You need 2.0 or higher GPA"**)

Notice that the two new if statements are indented the same amount as the print statement in the else block because all three statements belong to the else block. The following is a sample test run:

How many credits have you finished? 57

What is your cumulative GPA? 3.2

You are not eligible to graduate.

You need 68 credit hours or more.

The following is another test run:

How many credits have you finished? 72

What is your cumulative GPA? 1.87

You are not eligible to graduate.

You need 2.0 or higher GPA

The next program is about college tuition. North Carolina residents pay $50 per credit hour at Pythonville Community College, while non-residents pay $300 per credit hour. There is also a tuition cap that limits how much a student pays at the maximum. For residents, the cap is $600. For non-residents, it is $4000. Write a program to calculate tuition.

Before we write the Python code, let’s analyze the problem and design an algorithm first. We need to know two pieces of information: whether the user is a resident and how many credit hours he is taking. We first test whether the user is a resident, and write two blocks of code to calculate tuition using different rates and caps. The following algorithm is designed:

1. Enter residency and credit hours

2. If user is resident:

Calculate tuition = 50 \* credit hours

If tuition > 600:

Tuition = 600

Else:

Calculate tuition = 300 \* credit hours

If tuition > 4000:

Tuition = 4000

3. Display tuition

The following is the Python code:

resident = input(**"Are you NC resident?[y/n] "**)  
hours = float(input(**"How many credit hours are you taking? "**))  
**if** resident == **'y'**:  
 tuition = 50 \* hours  
 **if** tuition > 600:  
 tuition = 600  
**else**:  
 tuition = 300 \* hours  
 **if** tuition > 4000:  
 tuition = 4000  
print(**"Please pay $"**, tuition)

We have an if statement at the top level to test residency. Inside the if and else blocks of this if statement, we use the correct rate to calculate tuition and a nested if statement to apply the cap if the tuition is too high. The following is a sample test run:

Are you NC resident?[y/n] n

How many credit hours are you taking? 15

Please pay $ 4000

Please make up a few more test cases to test the program yourself.

# **3.10 Further Reading**

Please read chapter 3 sections 3.1 to 3.3 of the textbook. Section 3.1 introduces a few types of control structures. Section 3.2 talks about Boolean expressions, including different types of operators used in Boolean expressions. Sections 3.3 introduces the if statement, including the if, elif and else headers, blocks, and nested if statements.