COMP2152 LAB MANUAL

OPEN SOURCE DEVELOPMENT

This booklet will help the reader understand the concepts, principles, and implementation of the Python programming language. By the end of the booklet, the reader will be able to code comfortably in Python.

© 2018 George Brown College

Prepared by Ben Blanc

TABLE OF CONTENTS

CHAPTER 2

CONDITIONS AND OPERATIONS

CONDITIONAL STATEMENTS

A conditional statement is an expression that tests for a particular case. The case that the statement is testing for is up to you, the programmer.

EQUALITY OPERATORS AND TESTS

You can test for equality of any data type in Python.

Symbol	Meaning	Example	Result
==	Equal	(1 == 2)	false
!=	NOT equal	(4 != (19 % 5))	false

```
mB1 = True
mB2 = False
num1 = 10
num2 = 20
num3 = 30
s1 = "Hello"
s2 = "Bye"

print(mB1 == True)
print(mB2 == False)
print(mB2 != False)
print(mB1 == mB2)
print(mB1 != mB2)

print(s1 == s2)
print(s1 == "Hello")
print(s1 != s2)

print(num1 == 10)
print(num1 == 15)
print(num1 == num2)
print(num1 != num2)
print(num1 + num2 == num3)
print(num1 == num2 + num3)
```

CHAPTER 2 AT A GLANCE

In this chapter you will learn about making decisions and testing for conditions such as:

- If Statements
- Ternary Operators

You will also be introduced to relational operators and logical operators and we will add more items to the Order of Operators from Chapter 1.

RELATIONAL OPERATORS AND TESTS

Relations operators test for a specific inequality of two operands.

Symbol	Meaning	Example	Result
>	Greater than	(8 > 5) (1 > 2)	true false
<	Less than	(1 < 2) ('z' < 'a')	true false
>=	Greater than or equal	(100 >= 100) (14 >= 7)	true true
<=	Less than or equal	(100 <= 100) (27 <= 7)	true false

```
num1 = 10
num2 = 20
num3 = 30
c1 = 'T'
c2 = 'W'

print(num1 > 10)
print(num1 > num2)
print(num1 <=20)
print(num1 >=15)


print(c1 <= 'C')
print(c1 > 'R')
print(c1 > c2)

print(c2 > 'T')
print(c2 >= 'X')
print(c2 > c1)


print(num1 < 20)
print(num2 > num1)
print(num2 >=10)
print(num2 <=15)



print(num1 + num2 > num3)
print(num1 - num2 < num3)
print(num2 / num3 < num1)
print(num2 * num3 < num3)
```

LOGICAL OPERATORS AND TESTS

Logical operators are to be combined with series of conditional statements to create compound expressions. There are 3 logical operators supported in conditional statements:

- and
- or
- not

These operators have different placements and behaviors.

AND OPERATOR

The AND operator is used when you want to ensure **all conditions are met**.

For example,

```
mark = 75

print(mark >= 70 and mark <=80)

mark1 = 50

mark2 = 70

mark3 = 90

print(mark1 < mark2 and mark2 < mark3)
```

When putting the AND operator between two conditional expressions, all expressions must evaluate to TRUE for the **entire** conditional statement to be true.

OR OPERATOR

The OR operator is used when you want to ensure **at least one condition is met**.

For example,

```
color = "red"

print(color == "red" or color == "blue" or color == "green")

color = "blue"

print(len( color ) > 5 or color == "red")

color = "orange"

print('a' in color or len( color ) == 3)
```

When putting the OR operator between two conditional expressions, at least one expression must evaluate to TRUE for the **entire** conditional statement to be true.

NOT OPERATOR

The NOT operator is used when you want to negate (or reverse) the result of a conditional expression. It is placed at the beginning or an expression or an operand that results in a Boolean value (which will then be reversed).

Its placement at the beginning of an expression goes as follows:

```
mark = 60

print( not mark == 60)
```

Its placement at the beginning of an operand goes as follows:

```
trueOfFalse = True

print ( not trueOfFalse)
```

IF STATEMENTS

When you want to execute code based on a conditional statement, you would write an **if** statement code block. If statement code blocks can be sub-categorized into one-way, two-way and multi-way sub-categories.

ONE-WAY IF STATEMENTS

One-way **if** statement code blocks have the following syntax

```
if conditional_statement :
    //statements to be executed line 1
    //statements to be executed line 2
```

Fun with Flags

You can code a conditional statement that only has one operand if and only if that operand evaluates to a Boolean value. Take the example below:

```
myBool = True
if myBool :
    print("success")
```

When the value of myBool is TRUE, the output will be "success!". However, if the value of myBool is FALSE, there would be no output.

What is the output when i is 0? When i is 3?

```
i = 0
if i > 2:
    print("hello")
print("there")
```

Same question as above but with this code?

```
i = 0
if i > 2:
    print("hello")
    print("there")
```

TWO-WAY IF STATEMENTS

To make an **if** statement two-way, you would add no more than 1 **else** statement after the **if** statement. The else statement has NO conditional statement. In a two-way if statement, only one statement is executed but not both. If the **if** condition is TRUE, the code statement(s) are executed. If the if condition is FALSE, the **else** code statement(s) are executed.

Two-way **if** statement code blocks have the following syntax

if conditional_statement :
//statements to be executed line 1
//statements to be executed line 2
else
//statements to be executed line 1
//statements to be executed line 2

Please note that there cannot be an **else** statement without an **if** statement. Also, there cannot be multiple **else** statements. The follow code is NOT valid:

```
i = 2

else : # fragment else
    print("not valid")

else : # multiple else statements
    print("not valid")
```

What is the output when i is -1? When i is 2?

```
i = -1

if i < 0 :
    print("Number is negative")
else :
    print("Number is non-negative")
```

Same question as above but with this code?

```
i = -1

if i < 0 :
    print("Number is negative")
    print("It is winter")
else :
    print("Number is non-negative")
    print("It is summer")
```

MULTI-WAY IF STATEMENTS

To make an **if** statement multi-way, you would add the following statement(s) after the **if** statement:

- One or many **elif** statements
- no more than 1 **else** statement

In a multi-way if statement, only one statement is executed. If the **if** condition is TRUE, the code statement(s) are executed. If any of the **elif** conditions are TRUE, the code statements are executed. Lastly, if there is an **else** statement, its code statement(s) are executed.

Multi-way **if** statement code blocks have the following syntax

```
if conditional_statement :
        //statements to be executed line 1
        //statements to be executed line 2
elif conditional_statement :
        //statements to be executed line 1
        //statements to be executed line 2
```

OR

```
if conditional_statement :
        //statements to be executed line 1
elif conditional_statement :
        //statements to be executed line 2
elif conditional_statement :
        //statements to be executed line 3
        //statements to be executed line 4
```

OR

```
if conditional_statement :
        //statements to be executed line 1
        //statements to be executed line 2
elif conditional_statement :
        //statements to be executed line 1
        //statements to be executed line 2
elif conditional_statement :
        //statements to be executed line 3
        //statements to be executed line 4
else:
        //statements to be executed line 5
```

Can you determine the output of this code?

```
i = 4

if i == 0 :
    print ("first")
elif i > 0 and i < 4 :
    print ("second")
elif i > 3 and i != 5 :
    print ("third")
else :
    print("fourth")
```

What if i = 5?

Or i = 1?

Is there output in this code?

```
i = 0

if i > 0 :
    print ("first")
elif i < 0 :
    print ("second")
```

Please note that there cannot be an **elif** statement(s) without an **if** statement. Also, the restrictions of Two-Way If Statements apply also there cannot be multiple **if else** statements.

NESTED IF STATEMENTS

Nested **if** statements are **if** statements within **if** statements. It is best practice to ensure your indentation shows the nested relation of one if statement within another if statement.

Take the code below that calculated an employee's bonus amount.

```
hourlyEmployee = True;
hours = 50
bonus = 0
yearsEmployed = 5

if hourlyEmployee == True :
    if hours > 40 :
        bonus = 500
    else :
        bonus = 100
else :
    if yearsEmployed > 10 :
        bonus = 300
    else :
        bonus = 200

print (" hourlyEmployee:" , hourlyEmployee, "\n hours: ",
    hours, "\n bonus: ", bonus, "\n yearsEmployed:", yearsEmployed)
```

The first level if statement determines if the employee is hourly or not. From the first level of if statement, there is a nested if statements that will calculate the employee bonus.

Comprehensive If Statement Example

Here is now you would put together some of the concepts we've covered so far.

```
number = int(input("Enter number: "))

if number > 0 :
    print("Your number is greater than 0")
elif number == 0 :
    print("Your number is 0")
else :
    print("Your number is less than 0")
```

The code above attempts to get a number that the user has inputted and typecasts the value into an integer data type. The program then determines if the number if greater than, less than or equal to zero.

COMPACTED IF STATEMENTS (TERNARY OPERATOR)

The Compacted If Statement (Ternary Operator in other programming operators) is a compact way to have a two-way **if** statement. It takes the following structure:

true_statement if condition else false_statement

When using the ternary operator, the true_statement is stated first, then the conditional_expression is evaluated. If the conditional_expression is FALSE, the false_statement is evaluated.

A popular use for the ternary operator is to condense a two-way statement to one line instead of multiple lines.

```
age = 11
print ('adult' if age <18 else 'minor')
```

What is the value of grade?

```
examScore = 90
grade = 'A' if examScore > 89 else 'C'
print(grade)
```

What is the value of charges?

```
timeAtSite = 3.5
charges = 100.00 if timeAtSite < 2.0 \
    else timeAtSite * 50.00;
print(charges)
```

ORDER OF OPERATIONS

Now that we have introduced more operators, we can display the full order of operations table.

Order	Operator	Direction
1	**	Left to right
2	* / // %	Left to right
3	+ -	Left to right
4	< > <= >=	Left to right
5	== !=	Left to right
6	not	Left to right
7	and	Left to right
8	or	Left to right

Can you following the Order of Operations? Review the code and the steps

Take the following code:

```
value1 = 10
value2 = 20
value3 = 30
value4 = 40
value5 = 50


if value1 > value2 or value3 == 10 and value4 + 5 < value5 :
    print ("The expression evaluates to TRUE")

else :
    print ("The expression evaluates to FALSE")
```

The expression will be evaluated in the following order

1) value4 + 5
    a. = 40 + 5
        i. = 45

2) value1 > value2
    a. = 10 > 20
        i. = false

3) value4 + 5 < value5
    a. = 45 < 50
        i. = true

4) value3 == 10
    a. = 30 ==10
        i. = false

5) (value3 == 10) && ((value4 + 5) < value5)
    a. = false && true
        i. = false

6) (value1 > value2) || (value3 == 10) && ((value4 + 5) < value5)
    a. = false || false
        i. = false

Since the expression evaluated to FALSE, the ELSE output is executed.