Activity No. 3	
Control Structures	
Course Code: CPE009	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** 11/21/2019
Section: CPE12FB2	**Date Submitted:** 11/21/2019
Name: Julongbayan, Dan Angelo A.	**Instructor:** Sir Razon

1. Objective(s):

This activity aims to familiarize students in implementing control structures in Python

2. Intended Learning Outcomes (ILOs):

The students should be able to:

Create a Python program that can change its output based on different conditions
Use the different iterative statements in a Python program

3. Discussion:

Control structures allows programmers to change the flow or sequence of the code depending on specific conditions that may come from user input or from other data or variables within the program in runtime execution.

Conditional Statements allow the flow of a program to change if a certain condition was met. Conditional statements need to evaluate Boolean Expressions which returns a True or False value to determine if a condition was met or not.

A **Boolean Expression** can be constructed by using the following Relational and Boolean operators:

Relational Operator	Name
==	Equals
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to
!=	Not equal to

Boolean Operator	Name
and	Logical AND Operator
or	Logical OR Operator
not	Logical NOT Operator

Relational operators are commonly used to compare numerical values other than == which can also be used to compare Strings. In some cases, multiple Boolean Expressions may need be combined using **Boolean Operators** to create programs that can handle more complex logic such as compounding multiple Boolean expressions.

There are three keywords used to implement conditional statements in Python which are: **if**, **elif**, and **else** keywords. Conditional statements in Python use indentations to indicate which codes belong under a given condition. Given a conditional statement that only needs to handle one condition a one-way **if** statement can be used following the syntax:

```
if <Boolean expression>:
    <sequence of statements>
```

To handle two or more conditions, the **elif** statement can be used under the if statement to indicate alternate conditions that will execute on the condition that the Boolean expression of the if statement is false.

The following shows the syntax of an if elif statement to handle two specific conditions:

```
if <Boolean expression>:
    <sequence of statements>
elif <Boolean expression>:
    <sequence of statements>
```

Multiple elif statements can also be added.

```
if <Boolean expression>:
    <sequence of statements>
elif <Boolean expression>:
    <sequence of statements>
elif <Boolean expression>:
    <sequence of statements>
elif <Boolean expression>:
    <sequence of statements>
```

For scenarios where the program will need to handle conditions not specific to the Boolean Expressions written in the program. An else statement can be added to the if statement or if elif statement. The else statement can act as a default when all conditions specified were false. The syntax for an if else statement is shown:

```
if <Boolean expression>:
    <sequence of statements>
else:
    <sequence of statements>
```

To include the else keyword in an if elif statement the syntax is as follows:

```
if <Boolean expression>:
    <sequence of statements>
elif <Boolean expression>:
    <sequence of statements>
...
else:
    <sequence of statements>
```

Iterative Statements allows the flow of a program to repeat certain blocks of code and even the entire program based on certain conditions. Iterative statements are implemented in two ways in Python.

While Statements

The structure of Python's while loop statement is similar to that of other languages. Here is the syntax:

```
while <Boolean expression>:
    <sequence of statements>
```

The while loop will proceed and repeat the code indented under it while the Boolean expression is True. The While loop will terminate or stop repeating once the Boolean Expression is false. Values in a While loop should change so as it only repeats for a finite amount of iterations. While loops whose Boolean Expressions will never be false is called an Infinite Loop which is

will repeatedly loop the code inside it until the program is externally closed. An infinite loop is considered to be a logical error in a program.

For Statements

Python includes a for loop statement for more concise iteration over a sequence of values. The syntax of this statement is:

```
for <variable> in <iterable object>:
    <sequence of statements>
```

A For Loop in Python unlike in other programming languages loops through a sequence or iterable object. The iterable object can be a String, List, Tuple, Dictionary, and a Generator such as the range() function. The range() function is a common generator function used with Python's For Loop which generates numbers from 0 up to the specified value minus 1.

An example of its syntax and use is shown below:

```
for element in range(n):
    print(element)
```

```
for number in range(10):
    print(number)
```

This example code will generate numbers 0 to 9 because by default the range function starts from 0 and ends at n – 1. These codes will be seen and tried in more detail in the laboratory activity.

4. Materials and Equipment:

Desktop Computer with Anaconda Python
Windows Operating System

5. Procedure:

Relational and Boolean Operators

1. Use two variables a and b and assign the following values to the two variables then display the output at each set of values as specified in the table below.

Values of a and b	Codes to display per set
1, 2	print(a == b)
5, 0	print(a > b)
-1, -1	print(a >= b)
2, 2.0	print(a < b)
"same", 'same'	print(a <= b)
(1,2,3), (1,2,4)	print(a != b)

2. Use three variables a=1, b=2 and c=3 and then run the following print() functions below.
    a. print(a<b and c==3)
    b. print(a+b==2 and c<=3)
    c. print(a<b or c==3)
    d. print(a+b==2 or c<=3)

3. Write your observations per condition and experiment with the operators by changing the values of a, b, and/or c.

Conditional statements

Suppose that we have a sensor that reads the temperature inside a large machine. We want to create a program that will print a message on the control screen to inform the operator of the status of the machine.

Scenario 1: If the temperature is greater than 100°C display the status code 10 and the message "Too hot – turn off the machine"

1. Declare a variable named **statusCode** and assign it an initial value of 0
2. Declare a variable named **machineTemp** and assign it an initial value of 0.0
3. Receive a temperature input from the user and assign the value to the variable **machineTemp.**
   Note: The received value must be converted to float.
4. Type the code displayed below:

```
if machineTemp > 100:
    statusCode = 10
    print(f"Status Code: {statusCode} Too hot - turn off the machine.")
```

   Note: Observe the different way of displaying the value unlike the last activity. This is known as an f-string which is a new way to do string formatting for Python 3.7 and above.
5. Complete and run the program. Remember to note your observations.

Scenario 2:

a. If the temperature is greater than 100°C display the status code 10 and the message "Too hot – turn off the machine"
b. If the temperature is not greater than 100°C display the status code 2 and the message "Normal reading – No action required"

1. Modify the code snippet from step#4 and add a new condition that handles scenario 2b.
   Hint: This scenario acts as a default condition when scenario a is not true.

2. Complete and run the program. Remember to note your observations.

Scenario 3:

a. If the temperature is greater than 100°C, display the status code 10 and the message "Too hot – turn off the machine"
b. If the temperature is not greater than 100°C, display the status code 2 and the message "Normal reading – No action required"
c. If the temperature is in the range of 90°C and 100°C, display the status code 5 and the message "Critical temperature – Monitor the machine, do not leave the area"

1. Modify the code snippet from step#4 and #6 and add a new condition that handles scenario 3c.
   Hint: This will require compounded or join multiple Boolean expressions.

2. Complete and run the program. Remember to note your observations.

Iterative statements

Suppose that you are developing a program for a security door that implements a lock using an RFID card system. A Card will be tapped and scanned at the door and only registered cards will be able to unlock the door. We want to create a program that will display a message on the small monitor placed above the RFID scanner then unlock the door based on the data read from the RFID card.

Scenario 1: The program should only continuously accept and read cards. As of now there is only one registered card. The program should not terminate unless a special card with the code “sh00” is read by the system.

1. Declare two variables named **scannedCode,** and **registeredCode** and assign both an initial value of “”
2. The program should ask for the **registeredCode** only once in the program.
   Note: The **registeredCode** can be any string value (combinations of numbers and letters).
3. Type the following code below:

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
```

4. Complete and run the program. Remember to note your observations.

Scenario 2: The program should only repeatedly accept invalid cards up to three times. Afterwards it should shutdown automatically and be restarted by Authorized personnel. As of now there is only one registered card. The program should still terminate using special card with the code “sh00”.

1. You may copy or modify the previous code block. Declare a new variable named **counter** and assign a value of 3.
2. The program should still ask for the **registeredCode** only once in the program.
   Note: The **registeredCode** can be any string value (combinations of numbers and letters).
3. Modify the code in step#3 so that the code may look similar to the one below:

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

4. Complete and run the program. Remember to note your observations.

Scenario 3: The program should only repeatedly accept invalid cards up to three times. Afterwards it should shutdown automatically and be restarted by Authorized personnel. As of now there are three registered cards. The program does not ask anymore the registered codes but instead the codes are initialized in the program. The program should still terminate using special card with the code “sh00”.

1. You may copy or modify the previous code block. Change the name of registeredCode to registeredCodes and assign the values as follows:

```
registeredCodes = ["code1","code2","code3"]
```

   You may modify the code1, code2, code3 with values that you want.

2. Modify the code in the previous steps to look similar to the image below

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode in registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

Note: Observe the change in comparison of the if statement.

3. Complete and run the program. Remember to note your observations.

6. Supplementary Activity:

Tasks

1. **Write a program that would construct a table of values (first 10 values from the origin) using a for loop and range() function for the following Mathematical functions:**
   - **a.** Parabola: $y = x^2$
   - **b.** Square root function: $y = \sqrt{x}$
   - **c.** Absolute-value function: $y = |x|$

     Note: for c. 5 values should come from the negative side and 5 values should come from the positive side (Ex. -5 -4 -3 -2 -1 and 1 2 3 4 5). Using/importing a library is not allowed.

   Recall a table of values:

$y = |x|$

x	y
-2	2
-1	1
0	0
1	1
2	2

   You may refer to the documentation about the range function here: https://docs.python.org/3/tutorial/controlflow.html

2. **Write a program that determines the mean or average value of a given set of numbers.**
   Hint: Do not use any function that was not discussed or given in the laboratory activity.

7. Conclusion:

This lab activity has showed me just how powerful conditional statements are paired with relational or boolean operators and especially if its under a loop. It has also taught me additional syntax that can be useful in later activities specifically the use of f before the quotation marks for the print function to enable the output of variables as long as it is enclosed in curly braces.

8. Assessment Rubric:

Screenshots:

Nov 21, 2019 Lab Activity 3 - Control Structures

Relational and Boolean Operators

1. Use two variables a and b and assign the following values to the two variables then display the output at each set of values as specified in the table below.

Values of a and b	Codes to display per set
1, 2	print(a == b)
5, 0	print(a > b)
-1, -1	print(a >= b)
2, 2.0	print(a < b)
"same", 'same'	print(a <= b)
(1,2,3), (1,2,4)	print(a != b)

In [2]:
```
a, b = 1, 2
print(a == b)
```
```
False
```

In [3]:
```
a, b = 5, 0
print(a > b)
```
```
True
```

In [4]:
```
a, b = -1, -1
print(a >= b)
```
```
True
```

In [5]:
```
a, b = 2, 2.0
print(a < b)
```
```
False
```

In [6]:
```
a, b = "same", 'same'
print(a <= b)
```
```
True
```

In [7]:
```
a, b = (1,2,3), (1,2,4)
print(a != b)
```
```
True
```

2. Use three variables a = 1, b = 2, and c = 3 and then run the following print functions below.

   a. print(a<b and c==3)
   b. print(a+b==2 and c<=3)
   c. print(a<b or c==3)
   d. print(a+b==2 or c<=3)

In [8]:
```
a, b, c = 1, 2, 3
```

In [9]:
```
print(a<b and c==3)
```
```
True
```

In [10]:
```
print(a+b==2 and c<=3)
```
```
False
```

In [11]:
```
print(a<b or c==3)
```
```
True
```

In [12]:
```
print(a+b==2 or c<=3)
```
```
True
```

3. Write your observations per condition and experiment with the operators by changing the values of a, b, and/or c.

I observed that aside from integers and float values, you can also check the logical relations between character/string data types and ordered pairs.

Conditional Statements

Suppose that we have a sensor that reads the temperature inside a large machine. We want to create a program that will print a message on the control screen to inform the operator of the status of the machine.

Scenario 1:

If the temperature is greater than 100°C display the status code 10 and the message "Too hot – turn off the machine"

1. Declare a variable named **statusCode** and assign it an initial value of 0

In [13]:
```
statusCode = 0
```

2. Declare a variable named **machineTemp** and assign it an initial value of 0.0

In [14]:
```
machineTemp = 0.0
```

3. Receive a temperature input from the user and assign the value to the variable **machineTemp.**
Note: The received value must be converted to float.

In [21]:
```
machineTemp = float(input('Please input temperature: '))
```

```
Please input temperature: 101
```

4. Type the code displayed below:

```
if machineTemp > 100:
    statusCode = 10
    print(f"Status Code: {statusCode} Too hot - turn off the machine.")
```

Note: Observe the different way of displaying the value unlike the last activity. This is known as an f-string which is a new way to do string formatting for Python 3.7 and above.

In [22]:
```
if machineTemp > 100:
    statusCode = 10
    print(f"Status Code: {statusCode} Too hot - turn off the machine.")
```

```
Status Code: 10 Too hot - turn off the machine.
```

5. Complete and run the program. Remember to note your observations.

I observed the use of curly braces in the print function to print the values of a variable only if "F" is included before the quotation marks. I also noted the use of the relational operator > in the boolean expression used in the if condition, inputting any value from 100 and below does not satisfy the condition.

Scenario 2:

a. If the temperature is greater than 100°C display the status code 10 and the message "Too hot – turn off the machine"
b. If the temperature is not greater than 100°C display the status code 2 and the message "Normal reading – No action required"

1. Modify the code snippet from step#4 and add a new condition that handles scenario 2b. Hint: This scenario acts as a default condition when scenario a is not true.

In [24]:
```
machineTemp = float(input('Please input temperature: '))
```

```
Please input temperature: 99
```

In [25]:
```
if machineTemp > 100:
    statusCode = 10
    print(f"Status Code: {statusCode} Too hot - turn off the machine.")
else:
    statusCode = 2
    print(f'Status Code: {statusCode} Normal reading - No action required')
```

```
Status Code: 2 Normal reading - No action required
```

2. Complete and run the program. Remember to note your observations.

The only change in the code is the use of else, which gives a default response to the if condition if it isn't met.

Scenario 3:

a. If the temperature is greater than 100°C, display the status code 10 and the message "Too hot – turn off the machine"
b. If the temperature is not greater than 100°C, display the status code 2 and the message "Normal reading – No action required"
c. If the temperature is in the range of 90°C and 100°C, display the status code 5 and the message "Critical temperature – Monitor the machine, do not leave the area"

1. Modify the code snippet from step#4 and #6 and add a new condition that handles scenario 3c. Hint: This will require compounded or join multiple Boolean expressions.

In [36]:
```
machineTemp = float(input('Please input temperature: '))
```

```
Please input temperature: 90
```

In [37]:
```
if machineTemp > 100:
    statusCode = 10
    print(f"Status Code: {statusCode} Too hot - turn off the machine.")
elif machineTemp >= 90 and machineTemp <= 100:
    statusCode = 5
    print(f'Status Code: {statusCode} Critical temperature - Monitor the machine, do not leave the area')
else:
    statusCode = 2
    print(f'Status Code: {statusCode} Normal reading - No action required')
```

```
Status Code: 5 Critical temperature - Monitor the machine, do not leave the area
```

2. Complete and run the program. Remember to note your observations.

In this last scenario, the use of the elif(else-if) statements acquired the goal. What it does is it checks for another if condition after the one before it returns false, many else-if statements can be used after an if-statement. The use of else instead of another elif for status code 2 notes that it will execute that after all other if/elif statements fail to satisfy the condition

Iterative Statements

Suppose that you are developing a program for a security door that implements a lock using an RFID card system. A Card will be tapped and scanned at the door and only registered cards will be able to unlock the door. We want to create a program that will display a message on the small monitor placed above the RFID scanner then unlock the door based on the data read from the RFID card.

Scenario 1:

The program should only continuously accept and read cards. As of now there is only one registered card.
The program should not terminate unless a special card with the code "sh00" is read by the system.

1. Declare two variables named scannedCode, and registeredCode and assign both an initial value of ""

In [1]:
```
scannedCode, registeredCode = "", ""
```

2. The program should ask for the registeredCode only once in the program. Note: The registeredCode can be any string value (combinations of numbers and letters).

In [3]:
```
registeredCode = (input('Enter registered Code: '))
```

```
Enter registered Code: 3bb
```

3. Type the following code below:

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
```

In [4]:
```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
```

```
Enter the scanned code: 00c
Unauthorized.
Enter the scanned code: 00d
Unauthorized.
Enter the scanned code: 3bb
Authorized. Opening door.
Enter the scanned code: sh00
Shutdown card invoked.
```

4. Complete and run the program. Remember to note your observations.

I observed the use of the == relational operator which checks if one variable is equal to another variable. You can also use literals but usually atleast one is a variable.

Scenario 2:

The program should only repeatedly accept invalid cards up to three times. Afterwards it should shutdown automatically and be restarted by Authorized personnel. As of now there is only one registered card. The program should still terminate using special card with the code "sh00".

1. You may copy or modify the previous code block. Declare a new variable named **counter** and assign a value of 3.
2. The program should still ask for the **registeredCode** only once in the program.
   Note: The **registeredCode** can be any string value (combinations of numbers and letters).
3. Modify the code in step#3 so that the code may look similar to the one below:

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

4. Complete and run the program. Remember to note your observations.

In [10]:
```
counter = 3
scannedCode, registeredCode= '', ''
registeredCode = input('Enter the registered code: ')
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode==registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

```
Enter the registered code: 5cc
Enter the scanned code: 01
Unauthorized.
Enter the scanned code: 02
Unauthorized.
Enter the scanned code: 5cc
Authorized. Opening door.
Enter the scanned code: 03
Unauthorized.
Door indefinitely sealed. Call an Authorized Personnel.
```

Scenario 3:

The program should only repeatedly accept invalid cards up to three times. Afterwards it should shutdown automatically and be restarted by Authorized personnel. As of now there are three registered cards. The program does not ask anymore the registered codes but instead the codes are initialized in the program. The program should still terminate using special card with the code "sh00".

1. You may copy or modify the previous code block. Change the name of registeredCode to registeredCodes and assign the values as follows:

```
registeredCodes = ["code1","code2","code3"]
```

You may modify the code1, code2, code3 with values that you want.

In [3]:
```
scannedCode, registeredCodes = "", ["code1", "code2", "code3"]
counter = 3
```

2. Modify the code in the previous steps to look similar to the image below

```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode in registeredCode:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

In [4]:
```
while scannedCode != "sh00":
    scannedCode = input("Enter the scanned code: ")
    if scannedCode in registeredCodes:
        print("Authorized. Opening door.")
    elif scannedCode == "sh00":
        print("Shutdown card invoked.")
    else:
        print("Unauthorized.")
        counter-=1
        if counter==0:
            print("Door indefinitely sealed. Call an Authorized Personnel.")
            break
```

```
Enter the scanned code: code1
Authorized. Opening door.
Enter the scanned code: code2
Authorized. Opening door.
Enter the scanned code: code3
Authorized. Opening door.
Enter the scanned code: code4
Unauthorized.
Enter the scanned code: code5
Unauthorized.
Enter the scanned code: code6
Unauthorized.
Door indefinitely sealed. Call an Authorized Personnel.
```

3. Complete and run the program. Remember to note your observations.

I observed the use of an 'in' statement to check if the scannedCode matches with any of the codes in the 'registeredCodes' array. There is also a nested if inside the default else statement that checks if the counter has reached the value of 0(which initially starts at 3) before locking the doors and stopping the system from checking more RFIDs

Supplementary Activity:

1. **Write a program that would construct a table of values (first 10 values from the origin) using a for loop and range() function for the following Mathematical functions:**
   - **a.** Parabola: $y \quad x^2$
   - **b.** Square root function: $y = \sqrt{x}$
   - **c.** Absolute-value function: $y - |x|$

     Note: for c. 5 values should come from the negative side and 5 values should come from the positive side (Ex. -5 -4 -3 -2 -1 and 1 2 3 4 5). Using/importing a library is not allowed.

   Recall a table of values:

$$y = |x|$$

x	y
-2	2
-1	1
0	0
1	1
2	2

Parabola:

$$y = x^2$$

In [11]:
```
print(" x\t \t y")
for x in range(0,10):
    y = x**2
    print(f"{x}\t|\t{y}")
```

```
 x              y
0       |       0
1       |       1
2       |       4
3       |       9
4       |       16
5       |       25
6       |       36
7       |       49
8       |       64
9       |       81
```

Square root function:

$$y = \sqrt{x}$$

In [12]:
```
print(" x\t \ty")
for x in range(0,10):
    y = x**(1/2)
    print(f"{x}\t|\t{y}")
```

```
 x              y
0       |       0.0
1       |       1.0
2       |       1.4142135623730951
3       |       1.7320508075688772
4       |       2.0
5       |       2.23606797749979
6       |       2.449489742783178
7       |       2.6457513110645907
8       |       2.8284271247461903
9       |       3.0
```

Absolute-value function:

$$y = | x |$$

In [15]:
```
print(" x\t \ty")
for x in range(-5,5+1):
    y = (x**2)**(1/2)
    print(f"{x}\t|\t{y}")
```

```
 x              y
-5      |       5.0
-4      |       4.0
-3      |       3.0
-2      |       2.0
-1      |       1.0
0       |       0.0
1       |       1.0
2       |       2.0
3       |       3.0
4       |       4.0
5       |       5.0
```

2. **Write a program that determines the mean or average value of a given set of numbers.**
   Hint: Do not use any function that was not discussed or given in the laboratory activity.

In [8]:
```
count, number = 0, 0
prompt = 0
print("Input a negative value to stop input")
while prompt >= 0:
    prompt = float(input("Enter a number: "))
    if prompt < 0:
        print("Stopping input...")
        break
    else:
        count += 1
        number += prompt


mean = number / count
print("\nMean/Average: ", mean)
```

```
Input a negative value to stop input
Enter a number: 96
Enter a number: 75
Enter a number: 86
Enter a number: 94
Enter a number: 76
Enter a number: 85
Enter a number: 90
Enter a number: 91
Enter a number: -1
Stopping input...

Mean/Average:  86.625
```