CMPT 120: Introduction to Computing Science and Programming 1

Control Structures: Loops

Reminders

One-Stop Access To Course Information

- **Course website**: **One-stop access** to all course information.

  http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html

- Course Outline	**- Learning Outcomes**	**- Grading Scheme**
- Exam Schedule	**- Office Hours**	**- Lab/Tutorial Info**
- Python Info	**- Textbook links**	**- Assignments**
- CourSys/Canvas link	**- and more…**	

- **Canvas**: Discussions forum - https://canvas.sfu.ca/courses/39187
- **CourSys**: Assignments submission, grades - www.coursys.sfu.ca

How to Learn in This Course?

A **Attend** Lectures & Labs

R **Read** / review Textbook/Slides/Notes

R **Reflect** and ask Questions

O **Organize** – your learning activities on weekly basis,

and finally…

W **Write** Code, Write Code, and Write Code.

Deliverables

1. Deliverables are due by the given date and time.
2. For the course, we are using IDLE to write and run our Python code.
3. You can use the CSIL lab computers outside your lab hours.
4. Plan ahead your assignments and other deliverables. Computer crash, network problems etc. are not acceptable excuses for delays in deliverables.
5. You may use online Python interpreters for running and testing your codes, such as:

   https://repl.it/languages/Python3

Labs

1. Each lab has an assigned TA.
2. Attend your assigned lab and show your work to your TA for the participation marks.
3. Class enrolments and lab swaps are closed now.

Course Topics

1. General introduction
2. Algorithms, flow charts and pseudocode
3. Procedural programming in Python
4. Data types and Control Structures
5. Fundamental algorithms
6. Binary encodings
7. Basics of computability and complexity
8. Basics of Recursion
9. Subject to time availability:
   - Basics of Data File management

Today's Topics

1. Introduction to Loops: Repetition Structures
   a. The `for` Loop: a Count-Controlled Loop
   b. continue & break
   c. The `while` Loop: a Condition-Controlled Loop
2. In-Class Coding Practice
3. Sentinels
4. Input Validation Loops
5. Nested Loops

1

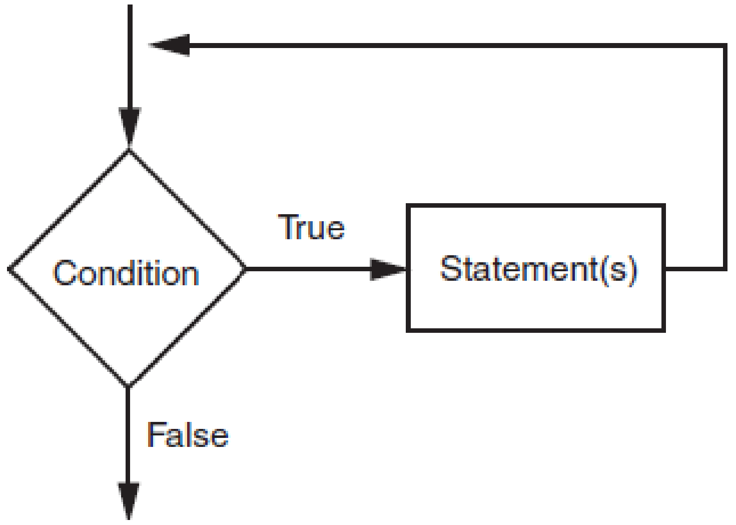
Introduction to Loops: Repetition Structures - while

The `while` Loop: Condition-Controlled Loop

- **Condition-Controlled loop**: An **indefinite** loop that iterates an **unspecified** number of times.
  - General format:

```
while condition:
    statements
```

- The loop executes while the **condition** is **true**.
- Based on the result of the **condition**, statements inside the loop may get executed:
  - **zero** time, or
  - **one** time, or
  - **any** number of times.
- We refer to the first line as the **while clause**.



The while Loop: Condition-Controlled Loop

- The loop begins:
  1. while it is set as True. For example:
  2. or, the result of the condition (Boolean expression) is true.
     - ***We would Use the condition method most often.***
- For a loop **to stop executing**, something MUST happen inside the loop to **makes the condition false**.
  - Else, the loop would run indefinitely.

Example 1:

```
while True:
   print('Welcome! ')
```

Example 2:

```
carryOn = 'Y'
while carryOn = 'Y':
   print('Welcome! ')
```

Count-Controlled Loop: How to Control Execution

1. You may define a variable to control the starting and ending points of the while loop. (Choose any variable name.)
2. Assign the variable a value .
   - The value should set the while condition true, initially. For example:

     index = 0 ,or
     carryOn = 'Y' **, or**
     keepGoing = True etc.

3. **Use the variable to define the while condition, so that it may become true to enter the loop.**

   while index < 11: ,or
   while carryOn == 'Y': ,or
   while keepGoing:

4. **Inside the loop, add code to change the value of the variable to make the condition false at some point.**
   - **For example: when the count is reached, or**
   - **when the user enter X to exit.**
   - **You might need the if statement.**

Condition-Controlled Loop: Example 1

```
num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in num_list:
    print(num)
```

```
num_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
num = 0
while num < len(num_list):
    print(num_list[num])
     num = num + 1
```

Or,

```
num = 0
while num < 11:
    print(num)
    num = num + 1
```

Condition-Controlled Loop: Example

```
frnd_list = ['Tiffany','Jiawei','Wenzhao', 'Ping-Chieh','Mitchell','Cole ‘]

for friend in frnd_list:
    invite = "Hi " + friend + ".  You are invited!"
    print(invite)
```

```
Hi Tiffany. You are invited!
Hi Jiawei. You are invited!
Hi Wenzhao. You are invited!
Hi Ping-Chieh. You are invited!
Hi Mitchell. You are invited!
Hi Cole . You are invited!
```

```
frnd_list = ['Tiffany','Jiawei','Wenzhao', 'Ping-Chieh','Mitchell','Cole ‘]

index = 0

while index < len(frnd_list):
    invite = "Hi " + frnd_list[index] + ".  You are invited!"
    print(invite)

    index = index + 1
```

2

In-Class Coding Practice

Count-Controlled Loop: Practice

1. Write a python program to print even numbers from 1 and 10.

```
num = 2

while num <= 10:
    print(num)

    num = num + 2          # num += 2
```

1. Write a python program to print odd numbers from 1 and 10.

```
num = 1

while num <= 10:
    print(num)

    num += 2
```

Count-Controlled Loop: break & continue

1. With the break statement we can stop the loop before it has looped through all the items.

```
num = 1
while num < 11:
   if num == 6:
      break
   print(num)
   num = num + 1
```

```
1
2
3
4
5
```

2. With the continue statement we can stop the current iteration of the loop, and continue with the next.

```
num = 1
while num < 11:
   if num == 6:
      num = num + 1
      continue
   print(num)
   num = num + 1
```

```
1
2
3
4
5
7
8
9
10
```

Augmented Assignment Operators (Shorthand Operators)

- In many assignment statements, the variable on the **left side** of the = operator also appears on the **right side** of the = operator.

  num = num + 1

- **Augmented Assignment Operators**: Special set of shorthand operators designed to use in assignment statements where a variable appears on the both sides of the equal sign.

Augmented Assignment Operators (Shorthand Operators)

Shorthand Op.	Usage	Equivalent
+=	num += 1	num = num + 1
-=	num -= 3	num = num – 3
*=	num *= 2	num = num * 2
/=	num /= 4	num = num / 4
%=	num %= a	num = num % a

Class Participation: Printing Tables

- Write a Python program to print a multiplication table using a **while** loop and upload on Canvas by tonight 11:59pm.
- Take input a number from the user.
- Use the for loop to print a multiplication table of the number user entered.
- If user enters 6, then the program output should be as shown on the right.
- You may add comments and appropriate headings.

```
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

Questions?