Control Structures: Loops
Reminders

Liaqat Ali, Summer 2018.
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](http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html)

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

• **Canvas**: Discussions forum - [https://canvas.sfu.ca/courses/39187](https://canvas.sfu.ca/courses/39187)

• **CourSys**: Assignments submission, grades - [www.coursys.sfu.ca](http://www.coursys.sfu.ca)

Liaqat Ali, Summer 2018.
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

Liaqat Ali, Summer 2018.
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

Liaqat Ali, Summer 2018.
Today’s Topics

1. Introduction to Loops: Repetition Structures
   a. The for Loop: a Count-Controlled Loop
   b. Break & continue
   c. The while Loop: a Condition-Controlled Loop

2. In-Class Coding Practice

3. Sentinels

4. Input Validation Loops

5. Nested Loops
Introduction to Loops: Repetition Structures
Introduction to Loops: Repetition Structures

• When we write programs, often we need to write code that performs the same task multiple times.

• print('t--------------------------')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t!                        !')
• print('t--------------------------')
Introduction to Loops: Repetition Structures

• One option is to duplicate the instructions. But, it has disadvantages:
  • ______________________________.
  • ______________________________.
  • ______________________________.
  • ______________________________.

• Programming languages provide ways to efficiently handle code duplications.

• We can call these ways as “Repetition Structures”.

• **Repetition structure**: A repetition structures makes computer repeat the code (included inside the structure) as many times as required.
  1. **count-controlled** loops (**for** loop i.e., repeat 5 times, 10 times, 100 times etc.)
  2. **condition-controlled** loops (**while** loop, repeat as long as some condition is true.)
Count-Controlled Loop (Definite Loop): for Loop

- **Count-Controlled loop**: A definite loop iterates a specific number of times.
- We use a `for` statement to write count-controlled loop.
  - Python `for loop` is designed to work with **sequence of data items**.
    - The for loop repeats or iterates once for each item in the sequence.

- **General format**:

  ```python
  for variable in range/list [val1, val2, etc]:
  statements
  ```

  - We refer to the first line as the _________________
  - Inside brackets a sequence of values, separated by comma, appear.
Count-Controlled Loop: Example

- Say, we want to print each name from the following list:

```
friends_list = ['Nick', 'Sharmin', 'Akash', 'Albert', 'Akshay', 'Yue', 'Vanessa', 'Justin', 'Jasmine']
```

```
print(friends_list[0])
print(friends_list[1])
print(friends_list[2])
```

- The name is a variable.
- You can use n, nm, x, y…
- The name will take the first value from the list (‘Nick’).
- Computer will run the print(name) statement for ‘Nick’.

---

Liaqat Ali, Summer 2018.
We can use data values in the for-clause as well.

```python
for name in ['Nick', 'Sharmin', 'Akash', 'Albert']:
    print(name)
```

Or,

```python
for num in [1, 2, 3, 4, 5]:
    print(num)
```
Count-Controlled Loop: Example 3

friend_list = ['Daewon', 'Harleen', 'Da Som', 'Tsz', 'Zaid', 'Yue', 'Adrian', 'Thomas', 'Wenshu']
In-Class Coding Practice
Count-Controlled Loop: Practice

1. Write a python program to print numbers from 1 and 10.
   ```python
   for num in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
       print(num)
   ```

2. Write a python program to print even numbers from 1 and 10.
   ```python
   for num in [2, 4, 6, 8, 10]:
       print(num)
   ```

3. Write a python program to print even numbers from 1 and 10.
   ```python
   for num in [1, 3, 5, 7, 9]:
       print(num)
   ```
Count-Controlled Loop: \texttt{range}

- We can use the \texttt{range} function in for clause to specify a range.
  - The \texttt{range()} can take up to three values as argument.
    - A single value means ‘repeat number of times’.
    - Two arguments specify a \texttt{from} to \texttt{range}.
      - \texttt{# num takes a value from range (target variable)}
    - Three arguments specify a \texttt{from} to \texttt{range} and step value.
Count-Controlled Loop: break & continue

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

   ```python
   for num in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
       if num == 6:
           break
   print(num)
   ```

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

   ```python
   for num in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
       if num == 6:
           continue
   print(num)
   ```
Class Participation: Printing Tables

• Write a Python program to print a multiplication table using a `for` 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.

<table>
<thead>
<tr>
<th>6 x 1</th>
<th>= 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 x 2</td>
<td>= 12</td>
</tr>
<tr>
<td>6 x 3</td>
<td>= 18</td>
</tr>
<tr>
<td>6 x 4</td>
<td>= 24</td>
</tr>
<tr>
<td>6 x 5</td>
<td>= 30</td>
</tr>
<tr>
<td>6 x 6</td>
<td>= 36</td>
</tr>
<tr>
<td>6 x 7</td>
<td>= 42</td>
</tr>
<tr>
<td>6 x 8</td>
<td>= 48</td>
</tr>
<tr>
<td>6 x 9</td>
<td>= 54</td>
</tr>
<tr>
<td>6 x 10</td>
<td>= 60</td>
</tr>
</tbody>
</table>

Questions?