Introduction to: Computers & Programming: Loops in Python

Adam Meyers
New York University
Outline

• What is a Loop?
• While Loops
• For Loops
• Examples
• Nested Loops
What is a Loop?

- Loops are control structures
  - A block of code repeats
  - The extent of the repetition is usually limited in some way
- Two kinds of Loops in Python
  - **while** loops
    - The evaluation of a boolean expression determines when the repetition stops
    - Changes in values of variables lead to different evaluations of the boolean expression on each repetition
    - When the expression is evaluated as **False**, the loop halts
    - If the expression can never evaluate as **False**, the loop is endless
  - **for** loops
    - The length of a “sequence” determines how many times the body executes
      - A sequence is an object that is made up of other objects (arranged in an order), e.g., a string is a sequence of characters: “duck” = “d”, “u”, “c”, “k”.
    - The loop uses one member of the sequence at a time, ending with the last one
An Endless Loop: keeping time

• Example

```python
def endless_timer ():
    import time
    now = 0
    while (True):
        time.sleep(1)
        now = now + 1
        print(now)
```

• This loop will keep counting seconds until stopped with a Control-C
What is a `while` Loop?

- A while loop consists of:
  - The word `while`
  - A boolean expression (`True` on the last slide)
  - A colon :
  - The body: an indented block of instructions

- The body of the loop repeats
  - until the boolean expression is False

- The loop on the previous slide is endless
  - because boolean expression is never False.
  - Any program can be stopped using Control-C
What is a *while* Loop? 2

- A loop that iterates a limited number of times

  ```python
def seconds_stop_watch (total_seconds):
    import time
    now = 0
    while (now < total_seconds):
      time.sleep(1)
      now = now + 1
      print(now)
```

- If we call `seconds_stop_watch` with 5 as an argument
  - The variable `now` is initialized to 0
  - The loop iterates 5 times
  - Each time: a second passes, 1 is added to now and now is printed
  - In this way, 1 to 5 is printed over 5 seconds

- How many times would a loop beginning *while (False):* repeat?
Loops for getting the Right Input

• Suppose you want to make sure that the user responds 'yes' or 'no'
  output = 'initial_input' # initialize the variable output
  while(not (output == 'yes') or (output == 'no')):
    if output != 'initial_input':
      print('Invalid Input!')
    output=(input("Please respond: 'yes' or 'no'"))

• Suppose you want to make sure the user enters a one digit integer.
  output = 'initial_input' # initialize variable output
  while (not (output in '0123456789')):
    if output != 'initial_input':
      print('Invalid Input!')
    output = input('choose an integer. ')
A sample *for* loop

• This function simulates a 60 second timer
  
  ```python
  def one_minute_timer ():
      print(0)
      for second in range(60):
          time.sleep(1)
          print(second + 1)
  ```

• The function prints 0, then enters a *for* loop
  – The loop iterates through a list of numbers from 0 to 59
    • The variable *second* is assigned that number as a value
    • The system waits one second
    • The system prints *second* + 1
The range function

- `range` takes three arguments:
  - `START`: An optional 1st argument
  - `MAXIMUM`: A required 2nd argument
  - `INCREMENT`: An optional 3rd argument
- When left out, `START` defaults to 0
- When left out, `INCREMENT` defaults to 1
- creates a sequence of numbers from `START` to `MAXIMUM-1` such that consecutive items in the sequence differ by `INCREMENT`
- Examples:
  - `range(5)` → `[0,1,2,3,4]`
  - `range(2,5)` → `[2,3,4]`
  - `range(10,2,-2)` → `[10,8,6,4]`
The *for* loop

- The first line – *for* variable *in* sequence:
  - *for* and *in* are keywords
  - variable can be any legal variable name
  - sequence is an ordered set of items
    - Python sequences includes data types like: *range*, *list*, *string*, …

- The body of the loop repeats once for each item in the sequence
- On each iteration, the variable is bound to the next item in the sequence

- Examples:
  - *for* character *in* 'multi-character':
    
    ```python
    print(character)
    ```
  - *for* number *in* range(5):
    
    ```python
    print(number)
    ```
Looping Through a String

• Using a **for** loop

```python
def for_string_loop (string):
    for letter in string:
        print(letter)

-- for-string-loop('Downward')
```

• Using a **while** loop

```python
def while_string_loop (string):
    position = 0  ## counter
    while(position < len(string)):
        print(string[position])
        position = 1 + position
```
Lengths and elements of Sequences

• The function *len* returns a sequence's length
  – The number of characters – `len('Downward')`
  – The number of integers in a range – `len(range(60))`
  – Etc.

• Elements in a range can be identified by their position, beginning with 0 and ending in one less than the length.
  – 'Downward'[0], range(5,10)[0]
  – 'Downward'[7], range(5,10)[4]
  – 'Downward'[8], range(5,10)[5] --- these are errors
for loops vs. while loops

• With some code modification, it is always possible to replace a for loop with a while loop, but not the other way around

• for loops are used for situations where you know the number of iterations ahead of time
  – e.g., looping through sequences

• There is no significant efficiency difference

• The difference relates to ease in which humans can read/write code
Simple Examples of Nested Loops

• What do you expect to print out from the following loop within a loop?
  – for number in [1,2,3,4]:
    for letter in 'abcd':
      print(number,letter)

• Function based on above example
  – def print_two_item_combinations(seq1,seq2):
    for item1 in seq1:
      for item2 in seq2:
        print(item1,item2)
A Few More Details About “print”

- print takes 2 optional arguments
- optional arguments have default values
- def testPrintDefaults(val1,val2=' ',val3='\n'):
  print('Using print defaults:',end='')
  print('A',val1,'B',val2,val3)
  print('Done')
  print('Using nondefaults sep and end:',end='')
  print('A',val1,'B',sep=val2,end=val3)
  print('Done')
More Examples in Separate Slides

- We will probably only have time to cover some of these in class.
Summary

- Loops provide a way to repeat blocks of instructions
- While loops are the most general
  - They require a condition for exiting the loop
    - If the condition is never true, the loop is endless
- For loops provide a simple way of repeating a block
  - once for each element in a sequence
  - or a fixed number of times
- A For loop can always be replaced by an equivalent While loop
- It is often useful to have nested loops (loops within loops)
Homework 1 (due Oct 5, 2015)

• Do E-learning Module 5
  – [http://cs.nyu.edu/elearning/CSCI_UA_0002/module05.php](http://cs.nyu.edu/elearning/CSCI_UA_0002/module05.php)

• Do Quiz 5 in NYUClasses

• Write function that will make a rectangle consisting of any character:
  – Make_character_rectangle(height, width, char)
    • Should make height rows of width instances of char

• Write a function that makes a parallelogram of characters:
  – Make_character_parallelogram(height, width, char)
    • The loop should add height-N spaces to the beginning of each line, where N starts out at 1 (the first line) and ends with height (the last line). Thus the parallelogram leans right, but touches the left margin.

• Write a timer that prints out every one tenth of a second
  – It should use the format: Hours:Minutes:Seconds.fraction
  – For example, 00:00:00.0, 00:00:00.1, 00:00:00.2, etc.
Homework 2 (Due Oct 13, 2015)

• Read Chapter 5
• Do E-learning Module 6
  – http://cs.nyu.edu/elearning/CSCI_UA_0002/module06.php
• Do Quiz 6 in NYUClassses