Looping Structures: Common Pitfalls in Loops; for, range

25 Sep 2009
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Quiz 1: 10min, 20pts

- WADES! (explain in English phrases) [5]
- What is pseudocode? [3]
- "Computers are t_____, computing scientists are t_____." [2]
  - In your own words, what does this mean for you as a programmer? [2]
- Name at least 4 types of documentation your Python programs need (internal or external). [4]
- Contrast static vs. dynamic typing: [4]
  - What are good and bad aspects of each?

Quiz 1: answers #1-2

- WADES! [5]
  - Write everything down
  - Apprehend the problem
  - Design a solution
  - Execute the solution (i.e., code)
  - Scrutinize the results (check w/user)
- What is pseudocode? [3]
  - A design of a solution showing program flow and data structures, but without nitty-gritty syntax details

Quiz 1: answers #3-4

- "Computers are tools, computing scientists are toolsmiths." [4]
  - Servant leadership – not to please self or make us look smart, but to build tools to assist others – is the user happy?
- Name at least 4 types of documentation your Python programs need (internal or external). [4]
  - Internal: comments, docstrings, good identifiers, online help/prompts, ...
  - External: write-up, design, pseudocode, programmer's diary, user manual, etc.

Quiz 1: answers #5

- Contrast static vs. dynamic typing: [4]
  - What are good and bad aspects of each?
  - Static: type of each variable is fixed
    - Must declare type and initialize variable before using it
  - Dynamic: type can change
    - Python: just assign to variable: x = 5 and it will spring into existence
    - Easier to use, but easier to have bugs: usually don't want to change type

Outline for today

- while loops: continue, break, else
- Common pitfalls with loops
- for loops
- range()

while loops: continue

- You can prematurely go to the next iteration of a while loop by using continue:
  - **counter = 0**
  - **while counter < 5:**
    - **counter += 1**
    - **if counter == 3:**
      - **continue**
    - **print counter,**
- Output:
  - **1 2 4 5**

while loops: break

- You can quit a while loop early by using break:

```
counter = 0
while counter < 5:
    counter += 1
    if counter == 3:
        break
    print counter,
```

- Output:

```
1 2
```

while loops: else

- The optional else clause of a while loop is executed when the loop condition is False:
  - **counter = 0**
  - **while counter < 5:**
    - **counter += 1**
    - **print counter,**
  - **else:**
    - **print “Loop is done!”**
- Output:
  - **1 2 3 4 5 Loop is done!**

while loops: break skips else

- If the loop is exited via break, the else clause is not performed:

```
counter = 0
while counter < 5:
    counter += 1
    if counter == 3:
        break
    print counter,
else:
    print "Loop is done!"
```

- Output: 1 2

Common errors with loops

- Print squares from 1^2 up to 10^2:
  - **counter = 0**
  - **while counter < 10:**
    - **print counter*counter,**
- What's wrong with this loop?
  - counter is never incremented!
- → Always make sure progress is being made in the loop!

Common errors with loops

- Count from 1 up to 10 by twos:
  - **counter = 1**
  - **while counter != 10:**
    - **print counter,**
    - **counter += 2**
- What's wrong with this loop? How to fix it?
  - **counter = 1**
  - **while counter < 10:**
    - **print counter,**
    - **counter += 2**

Common errors with loops

- Count from 1.1 up to 2.0 in increments of 0.1:
  - **counter = 1.1**
  - **while counter != 2.0:**
    - **print counter,**
    - **counter += 0.1**
- Seems like it should work, but it might not due to inaccuracies in floating-point arithmetic
  - **counter = 1.1**
  - **while counter < 2.0:**
    - **print counter,**
    - **counter += 0.1**

TODO

- Read ch3
- HW2 posted, due Mon (ch2,3)
- Lab2 posted, due next Wed/Thu
  - Uses selection(if) and/or looping
  - Short writeup ok