

PARAMETERS, INDEFINITE LOOPS, AND LOOP PATTERNS

CSSE 120—Rose Hulman Institute of Technology

Review: Python parameter passing

- Formal parameters only receive the **values** of the actual parameters
- Assigning a new value to a formal parameter does not affect the actual parameter
- Python passes actual parameters *by value*
- Can functions in Python **mutate** parameters?

Functions mutating parameters

- Can we write a function that exchanges the values of its two parameters?
- In Eclipse checkout the project named `Session11` from your SVN repository
- Study the code in the module `mutatingParameters.py` but don't run it
 - ▣ Together, observe what happens as we trace its execution in the debugger

Modifying Parameters

- How do functions send information back?
 - Return statements
 - *Mutating* parameters
 - Value of actual parameter must be a mutable object
 - *State* of the mutable object is changed
 - The actual parameter itself is NOT changed since it refers to the same object
 - Parameter is still passed by value

Recap: Two main types of loops

□ Definite Loop

- We know at the beginning of the loop how many times its body will execute
- Implemented in Python as a **for** loop.

□ Indefinite loop

- The body executes as long as some condition is true.
- Implemented in Python as a **while** loop.
- Can be an infinite loop if the condition never becomes False.

□ Python's **for line in file:** construct

- indefinite loop that looks syntactically like a definite loop!

Some indefinite loop patterns

- Interactive loops
- Sentinel loops
- File loops
- post-test loops
- "loop and a half"

Interactive: Make the user count

- Open module `averageUserCount.py` and execute it together
- When does the loop terminate?
- Is this the best way to make the user enter input?
 - Why?
 - Why not?

Interactive: Ask user if there is more

- Open module `averageMoreData.py` and execute it together
- User no longer has to count, but still has a big burden

Sentinel loop

- Open module `averageSentinel.py` and study the code then execute it together
- User signals end of data by a special "sentinel" value
- Note that the sentinel value is not used in calculations

Non-numeric Sentinel

- What if negative numbers are legitimate values?
- Open module `averageOtherSentinel.py` and study the code
 - ▣ Execute it together
 - ▣ What is the sentinel?
- **Again note:** sentinel value is not used in calculations.

File loop

- Open module `averageFile.py` and execute together with input file `numbers.txt`
- Uses a **for** loop as we have seen before
- Also note the conditional execution of `main()`

Interactive loop with graphics

- Display a window that contains a circle and a message saying "Click inside Circle".
- Whenever the user clicks outside the circle, display "You missed!".
- If the user clicks inside the circle, display "Bull's eye!". Then pause and close the window.
- Implement together in module `clickInsideCircle.py`

Individual Exercise on Using loops

- Define function **listAndMax()** in module **listMax.py** that
 - Prompts the user to enter numbers, one at a time
 - Uses a blank line (<ENTER>) as sentinel to terminate input
 - Accumulates the numbers in a list
 - Uses a loop to calculate the maximum value of the numbers
 - Returns two values:
 - the list of numbers entered in the order they were entered
 - the maximum value
- Define function **main()** in module **listMax.py** that
 - Calls listAndMax()
 - Prints the list of numbers entered
 - Prints the maximum value of the list of numbers

Start homework

- When you are through with your individual exercise commit your solutions to your svn repository
- Start working on homework 1 1