ECE12: Introduction to Programming
Lecture 4

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 4: Overview

- String formatting
  - Interactive examples
- Relational and logical operators
  - Comparison of values
- Conditional statements
  - **`if`** statement
- Block indentation

String formatting

- String formatting operator **`%`**
  - **`%`** conversion specifiers in string (left argument) are replaced with formatted values (right argument)
  - Example: **`print “%s is %d years old.” % (“Sophie”,7)`**
  - Conversion specifiers

`%c`	single ASCII character
`%s`	string value (opt.: string length)
`%d`	signed decimal integer (opt. number of digits)
`%u`	unsigned decimal integer (opt. number of digits)
`%o`	unsigned octal integer (opt. number of digits)
`%x,%X`	unsigned hexadecimal integer (0-1a-f, 0-1A-F)
`%f`	floating point number
`%e,%E`	floating point number in scientific notation
`%g,%G`	floating point number using least-significant digits

  - Optional formatting arguments

`-`	left/right justification
`N`	field width (i.e. number of digits/characters)

- String concatenation operator **`+`**
- String multiplication operator **`*`**

Relational Operators

- Relational operators (comparison of values)

`<`	less than
`>`	greater than
`<=`	less than or equal to
`>=`	greater than or equal to
`==`	equal to (remember, **`=`** means assignment!)
`!=`, **`<>`**	not equal to

- Comparison is defined for many types

integer	(e.g. 5 < 6)
floating point	(e.g. 7.0 < 7e1)
string	(e.g. “alpha” < “beta”)

- Result type is boolean, but represented as an integer

false	**`0`**
true	**`1`**

Logical Operators

- Logical operators
  (often used together with relational operators)
  - **`not`** logical negation
  - **`and`** logical and
  - **`or`** logical or

`x`	`y`	`not x`	`x and y`	`x or y`
`0`	`0`	`1`	`0`	`0`
`0`	`1`	`1`	`0`	`1`
`1`	`0`	`0`	`0`	`1`
`1`	`1`	`0`	`1`	`1`

- Argument and result types are boolean, represented as integer (or other type)
  - false **`0`** (or zero **`0.0`**, empty string **`“”`**, ...)
  - true **`1`** (or non-zero, non-empty string, ...)

Conditional Statements

- **`if`** statement
  - Control flow statement for decision making
  - Changes control flow depending on a condition
  - Example:
    - **`if x < 0: print x, “is negative!”`**
    - **`if x > 0: print x, “is positive!”`**
  - **`if`** construct consists of

keyword	**`if`**
condition	expression evaluated to true or false
colon	**`:`**
body	Python statement block

  - the body is executed only if the condition evaluates to true

Comparison of Values

- Example **`compare.py`**

```
# compare.py: compare two values
#
# author: Rainer Doemer
#
# modifications:
# 01/15/04 RD    initial version (based on compute.py)

# input
x = int(raw_input("Please enter 1st integer: "))
y = int(raw_input("Please enter 2nd integer: "))

# compute and output
if x < y:  print x, "is less than", y
if x > y:  print x, "is greater than", y
if x <= y: print x, "is less than or equal to", y
if x >= y: print x, "is greater than or equal to", y
if x == y: print x, "is equal to", y
if x != y: print x, "is not equal to", y
```

- Modifications
  - check if **`x`** is between **`10`** and **`20`**
  - check if **`x`** or **`y`** are odd or even numbers
  - try comparison of strings

Block Indentation

- Python groups statements into blocks by use of indentation
  - Other languages typically use

• parentheses	( )	e.g. Lisp
• braces	{ }	e.g. C, C++, Java
• keywords	begin end	e.g. Pascal

- Example:

```
# some statements...
if x < 0:
    print x, “is negative!”
    # handle negative values of x...
    if x < 100:
        print x, “is too small!”
        # handle the problem
if x > 0:
    # handle positive values of x...
# more statements...
```

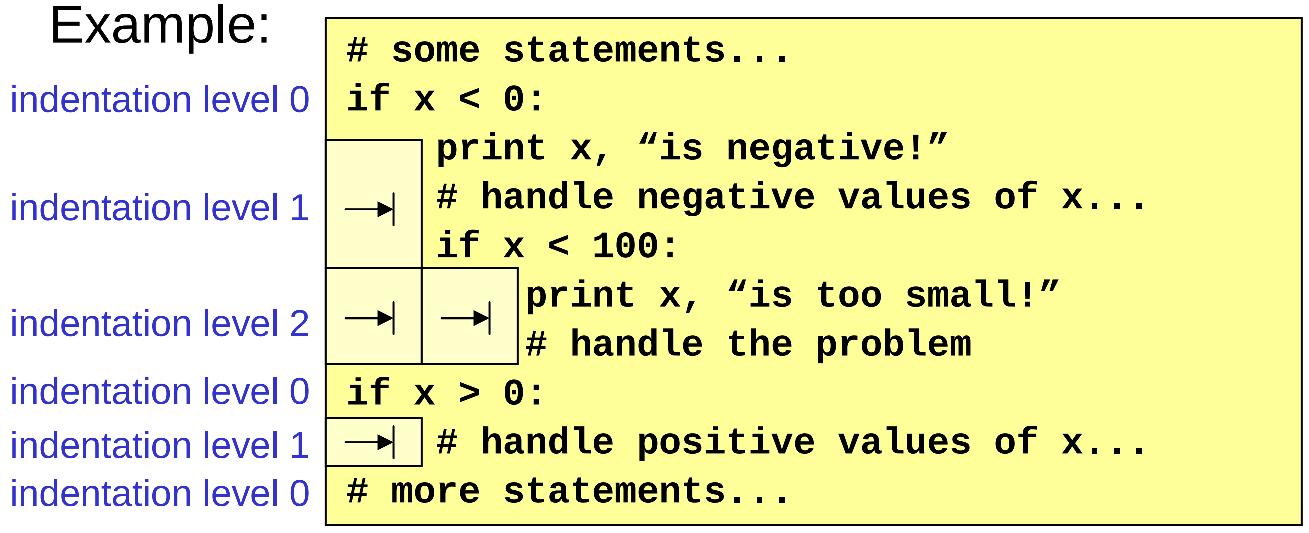
- Indentation increases readability of the code
  - in Python, proper indentation is required
  - in other languages, proper indentation is recommended

Block Indentation

- Python groups statements into blocks by use of indentation
  - Other languages typically use

parentheses	( )	e.g. Lisp
braces	{ }	e.g. C, C++, Java
keywords	begin end	e.g. Pascal

- Example:



```
# some statements...
if x < 0:
    print x, “is negative!”
    # handle negative values of x...
    if x < 100:
        print x, “is too small!”
        # handle the problem
if x > 0:
    # handle positive values of x...
# more statements...
```

- Indentation increases readability of the code
  - in Python, proper indentation is required
  - in other languages, proper indentation is recommended