ECE12: Introduction to Programming
Lecture 3

Rainer Dömer

doemer@uci.edu

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

Lecture 3: Overview

- Our first Python program
  - Hello World!
- Our second Python program
  - Input, computation, output
- Objects and variables
- Arithmetic operations
  - expression evaluation order
- String formatting

Our first Python Program

- Program file **`hello.py`**
  - **`# comment`** (until end of the line)
  - **`print`** function: formatted output (to stdout)

```
# hello.py: our first Python program
#
# author: Rainer Doemer
#
# modifications:
# 01/13/04 RD  initial version

print "Hello World!"
```

- Execute the program
  - run Python interpreter in batch mode
  - **`python hello.py`**
  - **`Hello World!`**
- Program modification
  - multiple statements...
  - text formatting using escape sequences...

Our first Python Program

- Text formatting using escape sequences

– **`\n`**	new line
– **`\t`**	horizontal tab
– **`\r`**	carriage return
– **`\b`**	back space
– **`\a`**	alert / bell
– **`\\`**	backslash character
– **`\”`**	double quote character
– **`\’`**	single quote character

Our second Python Program

- Program file **`compute.py`**

```
# compute.py: compute with two numbers
# author: Rainer Doemer
#
# modifications:
# 01/13/04 RD     initial version

# input
x = int(raw_input("Please enter a number:\n"))
y = int(raw_input("Please enter another number:\n"))

# compute
sum = x + y
product = x * y

# output
print "The sum is", sum
print "The product is", product
```

- Input, compute, output
- Step 1:
  - **`raw_input()`** function: reads a string as input (from stdin)
    - optional argument: prompt for input (string)
  - **`int()`** type conversion function: converts string to integer
  - variables **`x`** and **`y`** store the input data

Our second Python Program

- Program file **`compute.py`**

```
# compute.py: compute with two numbers
# author: Rainer Doemer
#
# modifications:
# 01/13/04 RD    initial version

# input
x = int(raw_input("Please enter a number:\n"))
y = int(raw_input("Please enter another number:\n"))

# compute
sum = x + y
product = x * y

# output
print "The sum is", sum
print "The product is", product
```

- Input, compute, output
- Step 2:
  - computation by use of assignment expression
  - variable **`sum`** receives result of addition operation
  - variable **`product`** receives result of multiplication operation

Our second Python Program

- Program file **`compute.py`**

```
# compute.py: compute with two numbers
# author: Rainer Doemer
#
# modifications:
# 01/13/04 RD     initial version

# input
x = int(raw_input("Please enter a number:\n"))
y = int(raw_input("Please enter another number:\n"))

# compute
sum = x + y
product = x * y

# output
print "The sum is", sum
print "The product is", product
```

- Input, compute, output

- Step 3:
  - **`print`** function outputs the result of the computation
  - formatted output (to stdout)

Objects and Variables

- Objects are used to store data
- Every object has

– a type	(e.g. integer, floating point, string)
– a value	(e.g. 42, 3.1415, “text”)
– a size	(number of bytes in the memory)
– a location	(address in the memory, aka. identity)

- Objects are either

– mutable	(object value can be changed)
– immutable	(object value cannot be changed)

- Variables
  - serve as identifiers for objects
  - are bound to objects
  - give objects a name

Arithmetic Operations

- Evaluation order of expressions
  - left to right (except for exponentiation!)
  - by operator precedence:

• unary plus, minus	+, -
• exponentiation	**
• multiplication, division, modulo	*, /, %
• addition, subtraction	+, -
• shift left, shift right	<<, >>
• bitwise and	&
• bitwise xor	^
• comparison	<, <=, ==, >=, >, !=, <>
• logical not	not
• logical and	and
• logical or	or

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)