ECE12: Introduction to Programming
Lecture 24

Rainer Dömer

doemer@uci.edu

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

Lecture 24: Overview

- File Processing
  - Module **`sys`**
    - Standard I/O streams
    - Environment functions
    - Command line arguments
  - Python as a scripting language
    - Example **`print.py`**
  - ANSI Color Terminal Control Codes
    - Example **`ansi_color.py`**
    - Example **`pretty_print.py`**

File Processing

- Module **`sys`** (system environment)
  - Standard I/O streams (opened by the environment)
    - **`sys.stdin`** standard input stream (read access)
    - **`sys.stdout`** standard output stream (write access)
    - **`sys.stderr`** standard error stream (write access)
  - Standard functions
    - **`sys.exit(value)`**
      - terminates the program execution
      - returns result **`value`** to the calling shell
        - » return **`value 0`** indicates normal program exit
        - » return **`value>0`** indicates exit due to an error condition
  - Command line arguments
    - **`sys.argv`** list of command line arguments (strings)
      - **`sys.argv[0]`** program name (always given)
      - **`sys.argv[1]`** first argument (if any)
      - **`sys.argv[2]`** second argument (if any), ...

File Processing

- Python as a scripting language
  - Any Python program can be seen as
    - a special Unix *shell* (the Python interpreter)
    - that executes a *shell script* (the Python program)
  - Example:
    - **`sh script.sh`** runs **`script.sh`** in **`sh`**
    - **`python program.py`** runs **`program.py`** in **`python`**
  - Automatic script execution
    - If the *first line* in the script is a valid *pseudo comment*, the Unix operating system automatically calls the specified shell for executing the script
    - Example:
      - **`#!/usr/bin/python`** calls **`python`** directly
      - **`#!/usr/bin/env python`** calls **`env`** which then calls **`python`**
    - The script essentially becomes a *command* !

File Processing

- Example: **`print.py`** (part 1/2)

```
#!/usr/bin/env python
#
# print.py: command to print the contents of a file
# author: Rainer Doemer
# 03/14/04 RD    initial version

import sys

# parse command line arguments
if len(sys.argv) != 2:
    sys.stderr.write("Usage: %s <file_name>\n"    \
                     % sys.argv[0])
    sys.exit(10)
filename = sys.argv[1]

# open the file for reading
try:
    file = open(filename, "r")
except IOError, message:
    print "Cannot open file %s for reading: %s"   \
                     % (filename,message)
    sys.exit(10)
...
```

File Processing

- Example: **`print.py`** (part 2/2)

```
...

# read the file
lines = file.readlines()

# close the file
file.close()

# output the contents page by page
lineno = 0
pageno = 0
pages = len(lines) / 10 + 1
for line in lines:
    if lineno % 10 == 0:
        pageno += 1
        print "\nFile \"%s\" Page %d/%d\n"     \
                % (filename,pageno,pages)
    lineno += 1
    print "%4d" % lineno, line,
```

File Processing

- Example: **`print.py`** (output)

```
% print.py print.py

File "print.py" Page 1/5

   1 #!/usr/bin/env python
   2 #
   3 # print.py: command to print the contents of a file
   4 #
   5 # author: Rainer Doemer
   6 #
   7 # modifications:
   8 # 03/14/04 RD      initial version
   9
  10 # imports

File "print.py" Page 2/5

  11 import sys
...
```

File Processing

- ANSI Color Terminal Control Codes
  - Subset of VT100 terminal escape sequences
    - ESC = "\x1b"

• Black	ESC + "[30m"	• Reset	ESC + "[0m"
• Red	ESC + "[31m"	• Bright	ESC + "[1m"
• Green	ESC + "[32m"	• Dim	ESC + "[2m"
• Yellow	ESC + "[33m"	• Uscore	ESC + "[4m"
• Blue	ESC + "[34m"	• Blink	ESC + "[5m"
• Magenta	ESC + "[35m"	• Reverse	ESC + "[7m"
• Cyan	ESC + "[36m"	• Hidden	ESC + "[8m"
• White	ESC + "[37m"		

File Processing

- Example: **`ansi_color.py`** (part 1/2)

```
# ansi_color.py: Escape sequences for ANSI Color Terminal
# author: Rainer Doemer
# 03/16/04 RD    initial version

# definitions of control codes
ESC      = "\x1b"

Reset    = ESC + "[0m"
Bright   = ESC + "[1m"
Dim      = ESC + "[2m"
Uscore   = ESC + "[4m"
Blink    = ESC + "[5m"
Reverse  = ESC + "[7m"
Hidden   = ESC + "[8m"

Black    = ESC + "[30m"
Red      = ESC + "[31m"
Green    = ESC + "[32m"
Yellow   = ESC + "[33m"
Blue     = ESC + "[34m"
Magenta  = ESC + "[35m"
Cyan     = ESC + "[36m"
White    = ESC + "[37m"
...
```

File Processing

- Example: **`ansi_color.py`** (part 2/2)

```
...

# driver program for testing
if __name__ == "__main__":
    print Reset, "Reset", Reset
    print Bright, "Bright", Reset
    print Dim, "Dim", Reset
    print Uscore, "Uscore", Reset
    print Blink, "Blink", Reset
    print Reverse, "Reverse", Reset
    print Hidden, "Hidden", Reset
    print Black, "Black", Reset
    print Red, "Red", Reset
    print Green, "Green", Reset
    print Yellow, "Yellow", Reset
    print Blue, "Blue", Reset
    print Magenta, "Magenta", Reset
    print Cyan, "Cyan", Reset
    print White, "White", Reset
```

File Processing

- Example: **`ansi_color.py`** (output)

```
% python ansi_color.py
 Reset
 Bright
 Dim
 Uscore
 Blink
 Reverse

 Black
 Red
 Green
 Yellow
 Blue
 Magenta
 Cyan
 White
```

File Processing

- Example: **`pretty_print.py`** (part 1/4)

```
#!/usr/bin/env python
#
# pretty_print.py: command to print programs in pretty format
#
# author: Rainer Doemer
#
# modifications:
# 03/16/04 RD    initial version (based on print.py)

# imports
import sys
import re
import ansi_color

# settings
LinesPerPage = 20

...
```

File Processing

- Example: **`pretty_print.py`** (part 2/4)

```
...
# function definitions
def ColorCode(code):
    (code,n) = re.subn(r"^\s*(#.*)",                          \
        ansi_color.Green + r"\1" + ansi_color.Reset, code)
    if n == 0:
        code = re.sub(r"(r?\"([^\\]|\\.)*?\")",               \
        ansi_color.Magenta + r"\1" + ansi_color.Reset, code)
        code = re.sub(r"\b(and|assert|break|class|continue|def|del|"
                r"elif|else|except|for|from|if|import|in|not|"
                r"or|pass|print|raise|return|try|while)\b", \
                ansi_color.Red + r"\1" + ansi_color.Reset, code)
    return code

# check command line arguments
if len(sys.argv) < 2:
    sys.stderr.write("Usage: %s <file_name>[.py]...\n" % sys.argv[0])
    sys.exit(10)

...
```

File Processing

- Example: **`pretty_print.py`** (part 3/4)

```
...
# process each specified file
for filename in sys.argv[1:]:

    # read the file
    if not filename.endswith(".py"):
        filename += ".py"
    try:
        file = open(filename, "r")
    except IOError, message:
        print "Cannot open file %s: %s" % (filename,message)
        print "Skipping..."
        continue
    lines = file.readlines()
    file.close()

...
```

File Processing

- Example: **`pretty_print.py`** (part 4/4)

```
...

# output the contents page by page
    lineno = 0
    pageno = 0
    pages = len(lines) / LinesPerPage + 1
    for line in lines:
        if lineno % LinesPerPage == 0:
            pageno += 1
            print "\nFile \"%s\", page %d/%d\n“              \
                        %(filename,pageno,pages)
        lineno += 1
        prefix = "%s%4d%s" %                                 \
                 (ansi_color.Blue,lineno,ansi_color.Reset)
        code = line.rstrip()
        code = code.expandtabs()
        code = ColorCode(code)
        print prefix, code
```

File Processing

- Example: **`pretty_print.py`** (output)

```
% pretty_print.py pretty_print.py

File "pretty_print.py", page 1/4

   1 #!/usr/bin/env python
   2 #
   3 # pretty_print.py: command to print programs in pretty format
   4 #
   5 # author: Rainer Doemer
   6 #
   7 # modifications:
   8 # 03/16/04 RD   initial version (based on print.py)
   9
  10 # imports
  11 import sys
  12 import re
  13 import ansi_color
  14
  15 # settings
  16 LinesPerPage = 20
  17
  18 # function definitions
...
```