Quotes and substitution (1)

- Suppose that
  \[ k = 3; \]

- Single quotes allow no substitution except the escape sequences \ and \’ — that is why
  \[ \text{print('}k\text{\n');} \]
  gives a 4-character string \$k\n — no new line.

- Double quotes allow substitution of variables like \$k and control codes like \n (newline). So,
  \[ \text{print("}k\n");} \]
  gives 3 (and a new line).

Quotes and substitution (2)

- Back-quotes also allow substitution. Next, they try to execute the result as a system command, returning the command's output. For example:
  % cat bq1
  \$y = `date`; print($y);
  % perl bq1
  Thu Oct 25 20:17:54 EDT 2001
  % cat bq2
  \$x = "date"; print(`$x`);
  % perl bq2
  Thu Oct 25 20:18:01 EDT 2001
Command-line arguments

- Suppose a program is invoked with the command:
  
  `cla -o basket.html candle.html`

- The built-in list `@ARGV` contains three elements:
  
  `('-o', 'basket.html', 'candle.html')`

- These elements can be accessed as:
  