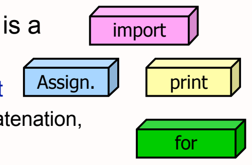
Objectives

- More on definite for loops, xrange
- Formatting output
- Four Puzzles From Cyberspace

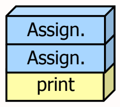
Programming Building Blocks

- Each type of statement is a building block
  - Initialization/Assignment
    - Arithmetic, string concatenation, input/raw_input
  - Print
  - Import
  - For loop



Design Patterns

- General, repeatable solution to a commonly occurring problem in software design
  - Template for solution
- Example (**Standard Algorithm**)
  - Get input from user
  - Do some computation
  - Display output



x = input("…")
ans = …
print ans

Using the `For` Loop

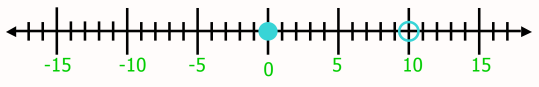
- Good for when know how many times loop will execute
  - Repeat N times

```
for count in xrange(10):
        statement_1
        statement_2
        …
        statement_n
```

- "Body" of for loop
- Gets repeated
- Note indentation

`xrange()`

- `xrange` is a built-in function
  - 1 argument: xrange(stop)
  - 2 arguments: xrange(start, stop)
  - 3 arguments: xrange(start, stop, step)

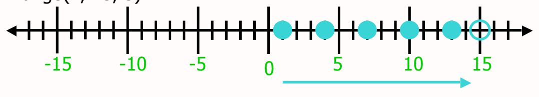


xrange(10)
xrange(0,10)

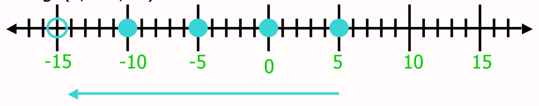
[start, stop)

`xrange()`

xrange(1, 15, 3):



xrange(5, -15, -5):



new_for.py

Accumulator Design Pattern

- Initialize accumulator variable
- Loop until done
  - Update the value of the accumulator
- Display result

Review Practice

- Add 5 numbers, inputted by the user
  - Step through in memory

Formatting Output

- Make the output from the program easy for user to read, understand

- Formatting Options:
  - Using str() constructor
  - Format specifiers

Problem with `print`

- By default, `print` puts spaces around numbers when they get printed out
  - Example:

```
x = 13.54
print “You owe $”, x, “.”
```

Solution: using str()

- Recall: str() is constructor/converter function to convert other data types to strings
  - Example: str(33) → ‘33’

- Use constructor with the **+** (i.e., concatenation) operator when printing output
  - print “You owe $” + str(x) + “.”

Another problem with print

```
SALES_TAX=.05  # the sales tax in VA

value = input("How much does your item cost? ")

tax = value * (1+SALES_TAX)

print "Your item that cost ($”, value, ")",
print "costs $”, tax, "with tax"
```

sales_tax.py

Example using Format Specifiers

Formatting operator

Format specifier

```
print "Your item that cost ($%.2f)" % value,
print "costs $%.2f with tax" % tax
```

Replacement values

- Format specifiers give control over how output is displayed to user
  - Right, left justification
  - Number of decimals to display

Format Specifiers

The [] mean "optional"

- General format:
  %[flags][width][.precision]code
  - flags:
    - 0: zero fills
    - +: adds a + sign before positive values
    - -: left-justification (default is right-justification)
  - width:
    - *Minimum* number of character spaces reserved to display the entire value
    - Includes decimal point, digits before and after the decimal point and the sign

Format Specifiers

- General format:
  %[flags][width][.precision]code
  - precision:
    - Number of digits after the decimal point for **real** values
  - code:
    - Indicates the value's **type**/way to format
      - s - string
      - d (or i) - integer
      - f - floating point
      - e - floating point with exponent

Using Format Specifiers

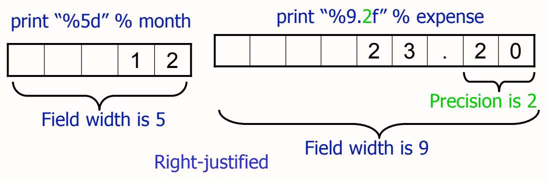
- Basic format is
  - print <templatestring> % (<value1>, <value2>, …, <valuen>)

Formatting operator

Replacement values

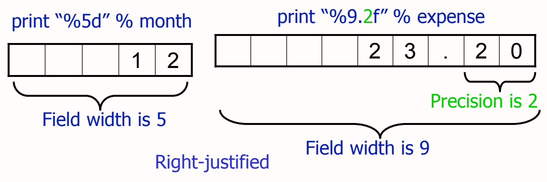
- templatestring is a template for the print statement with format specifiers instead of the values
  - For each format specifier in templatestring, should have a **replacement value**
  - Throws **TypeError** if not enough replacements for specifiers in templatestring
  - If only one replacement value, don't need ()

Format Specifiers

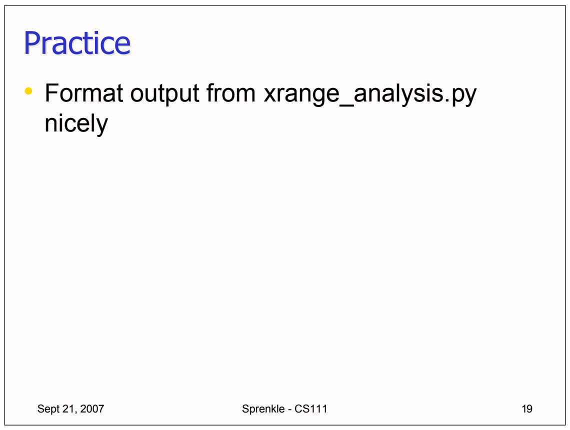


- What if precision is bigger than the decimal places?
- What if field width is smaller than the length of the value?

Format Specifiers

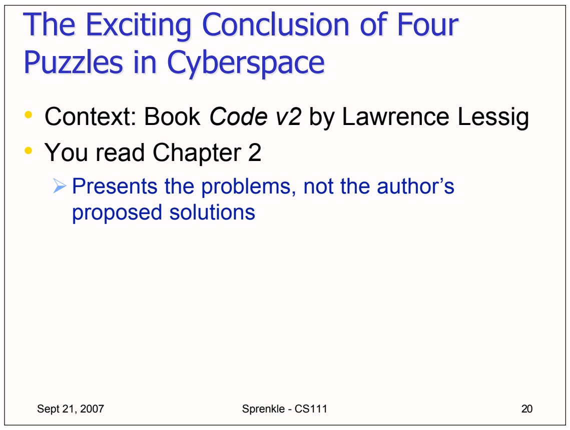


- What if precision is bigger than the decimal places?
  - Fills decimal with 0s
- What if field width is smaller than the length of the value?
  - Prints entire value



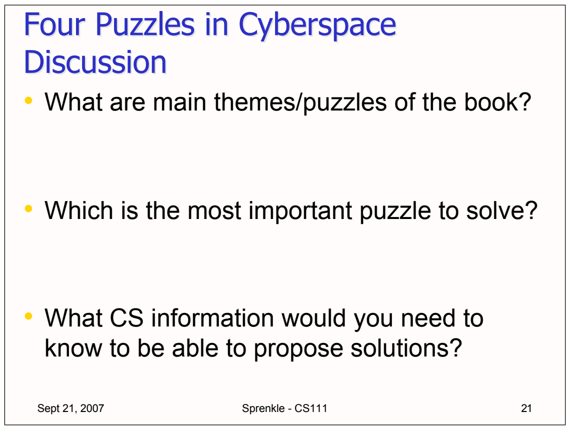
Practice

- Format output from xrange_analysis.py nicely



The Exciting Conclusion of Four Puzzles in Cyberspace

- Context: Book *Code v2* by Lawrence Lessig
- You read Chapter 2
  - Presents the problems, not the author's proposed solutions



Four Puzzles in Cyberspace Discussion

- What are main themes/puzzles of the book?
- Which is the most important puzzle to solve?
- What CS information would you need to know to be able to propose solutions?