MORE STRINGS, FILE PROCESSING, AND METHODS VS. FUNCTIONS

CSSE 120 – Rose-Hulman Institute of Technology

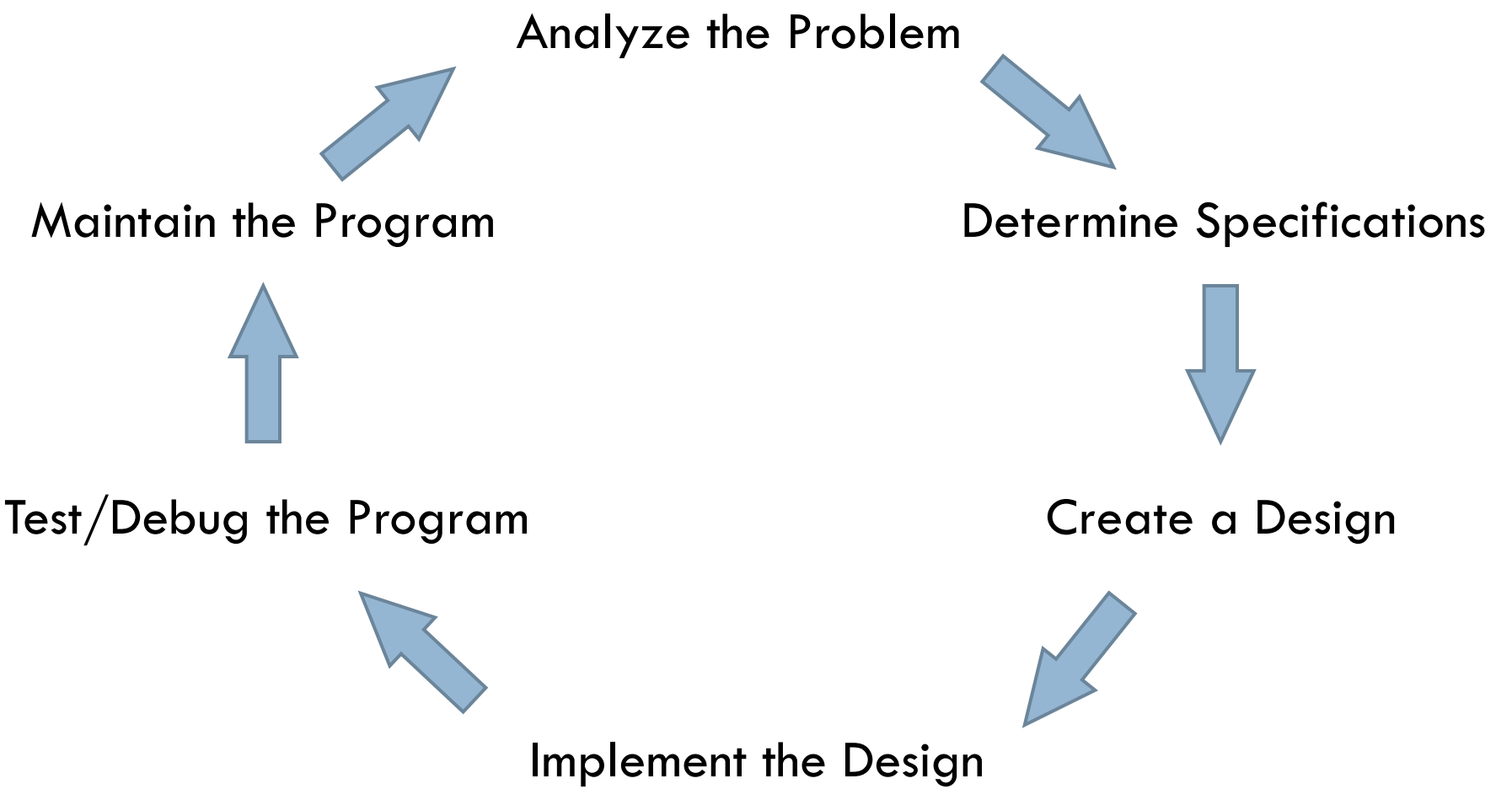
Outline

- Class exercise: processing with lists
- String representation
- More on string input and output
- File processing
- Coming attraction: Objects
- In-class practice time

Day, Month → Day of year

- When calculating the amount of money required to pay off a loan, banks often need to know what the "ordinal value" of a particular date is.
  - For example, March 6 is the 65th day of the year (in a non-leap year).
- We need a program to calculate the day of the year when given a particular month and day.

The Software Development Process



Phases of Software Development

- **Analyze:** figure out exactly what the problem to be solved is
- **Specify:** WHAT will program do? NOT HOW.
- **Design:** SKETCH how your program will do its work, design the algorithm
- **Implement**: translate design to computer language
- **Test/debug**: See if it works as expected.
  bug == error, debug == find and fix errors
- **Maintain**: continue developing in response to needs of users

Solution

```
# Calculate day of year for a given date in a non leap year
months = ["jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "nov", "dec"]
length = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
m = raw_input("Enter month name (3-letters, lowercase): ")
d = input("Enter the day of the month: ")

# Find out where in list of months this month falls
indx = months.index(m)
daysOfYr = 0
for i in range(indx):
    daysOfYr = daysOfYr + length[i]

daysOfYr = daysOfYr + d
m = m[0].upper() + m[1:]
print m, d, "is day", daysOfYr, "of this year."
```

String Representation

- Computer stores 0s and 1s
  - Numbers stored as 0s and 1s
  - What about text?
- Text also stored as 0s and 1s
  - Each character has a code number
  - Strings are sequences of characters
  - Strings are stored as sequences of code numbers
  - Does it matter what code numbers we use?
- Translating: ord(<char>) chr(<int>)

Consistent String Encodings

- Needed to share data between computers
- Examples:
  - ASCII—American Standard Code for Info. Interchange
    - "Ask-ee"
    - Standard US keyboard characters plus "control codes"
  - Extended ASCII encodings
    - Add various international characters
  - Unicode
    - Tens of thousands of characters
    - Nearly every written language known

Class Exercise: Encoding

- Download encode.py from Angel:
  - Lessons → Modules to Download in Class → Session 5 → encode module
- Verify that the module works on your computer:
  - Run the module
  - Enter the message: Spam
  - Enter the key: 1
  - Expected result: [84, 113, 98, 110]
- Answer In-class Quiz questions 4 and 5

The eval Function

- Syntax:
  - eval(<string>)
- Semantics
  - Input: any string
  - Output: result of evaluating the string as if it were a Python expression

String Formatting

- The % operator is *overloaded*
  - Multiple meanings depending on context
- What does it mean for numbers?
- Other meaning for <string> % <tuple>
  - Plug values from tuple into “slots” in string
  - Slots given by *format specifiers*

Format Specifiers

- Syntax:
  - %<width>.<precision><typeChar>
- Width gives total spaces to use
  - 0 means as many as needed
  - 0*n* means pad with leading 0s to *n* **total** spaces
  - -*n* means "left justify" in the *n* spaces
- Precision gives digits after decimal point
- TypeChar is:
  - f for float, s for string, or d for decimal (i.e., int)

File Processing

- Manipulating data stored on disk
- Key steps:
  - *Open* file
    - For reading or writing
    - Associates file on disk with a *file variable* in program
  - *Manipulate* file with operations on file variable
    - Read or write information
  - *Close* file
    - Causes final "bookkeeping" to happen

File Writing in Python

- Open file:
  - Syntax: <filevar> = open(<name>, <mode>)
  - Example: file_to_write = open('average.txt', '**w**')
    - Replaces contents!
- Write file:
  - Syntax: <filevar>.write(<string>)
- Close file:
  - Syntax: <filevar>.close()
  - Example: file_to_write.close()

File Reading in Python

- Open file: file_to_read = open('grades.txt', '**r**')
- Read file:
  - <filevar>.read() Returns one **BIG** string
  - <filevar>.readline() Returns next line, including \n
  - <filevar>.readlines() Returns **BIG** list of strings, 1 per line
  - for <ind> in <filevar> Iterates over lines efficiently
- Close file: file_to_read.close()

A “Big” Difference

- Consider:
  - file_to_read = open ('grades.txt', 'r‘)
    for line in file_to_read.readlines():
      # process line
    file_to_read.close()
  - file_to_read = open ('grades.txt', 'r‘)
    for line in file_to_read:
      # process line
    file_to_read.close()
- Which takes the least memory?

Up Next: Objects

- Why do we apply some operations like this:
  - infile = open('file.txt','r')
  - abs(-1.2)
- and others like this:
  - infile.read()
  - circle.draw(win)
- Files and circles are *objects*—data plus operations
- <object>.<methodName>() is a *method call*
  - Tells object to do something

Practice

- Hand in quiz
- Start working on HW5
  - Practice with string formatting
  - Practice with file processing
- On Angel
  - Lessons → Homework → Homework 5 → Homework 5 Instructions