

# MORE STRINGS AND FILE PROCESSING

# Bonus Points



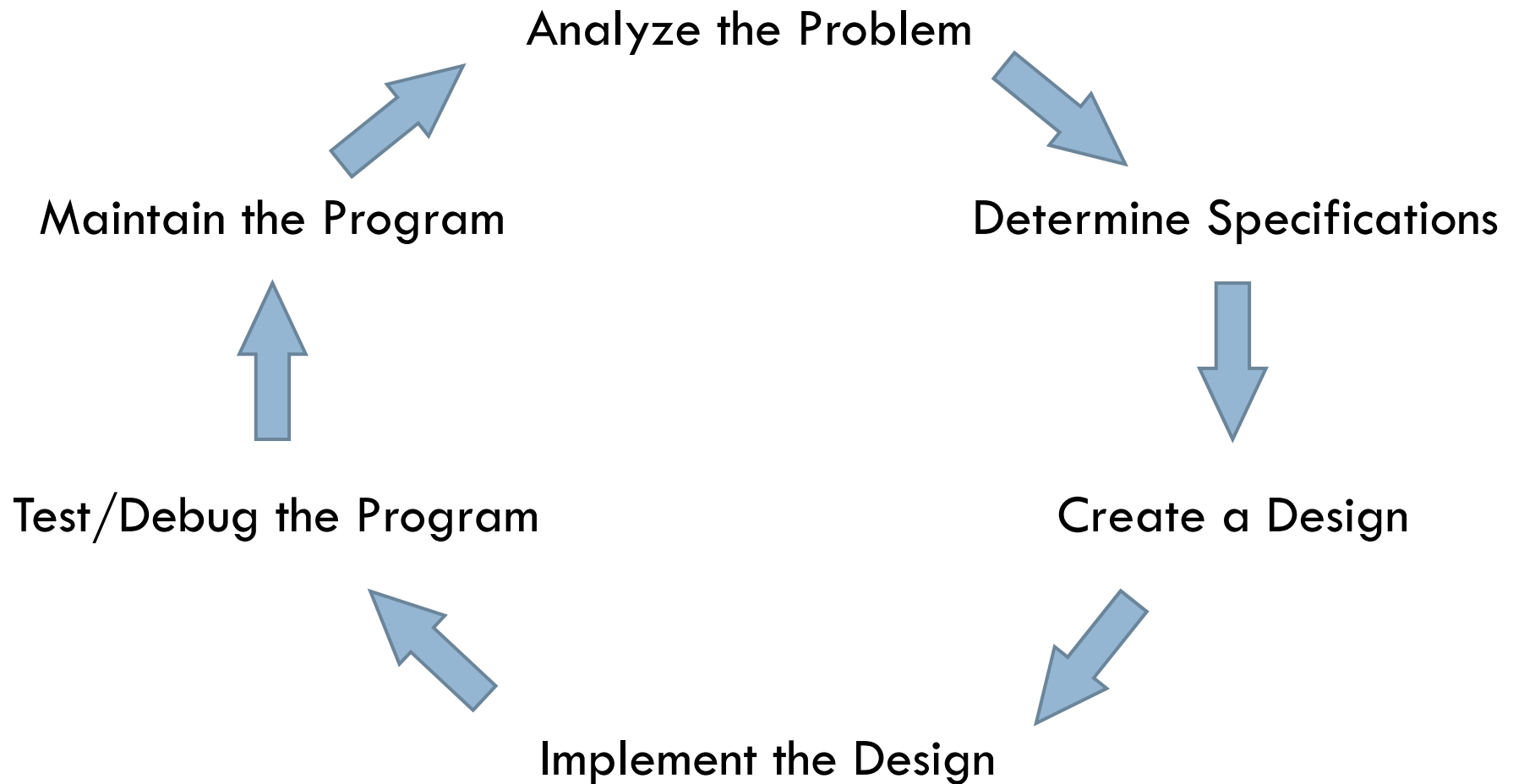
- If you did the Eclipse configuration for today, show me:
  - ▣ The output of either spam.py or greeting.py
  - ▣ spam.py source code if you have it
- While I am checking people's code, please do question 1 on the quiz (review).
  
- ALSO, please download from Angel: Lessons > Modules to Download > Session5 > session05.py

# 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

# 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

# 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 (or width omitted) means as many as needed
  - ▣ `0n` means pad with leading 0s to ***n*** **total** spaces
  - ▣ `-n` means “left justify” in the *n* spaces
- Precision gives digits after decimal point, **rounding if needed.**
- TypeChar is:
  - ▣ `f` for float, `s` for string, or `d` for decimal (i.e., int)
- Note: this RETURNS a string that we can print
  - ▣ Or write to a file using `write(string)`, as you’ll need to do on tonight’s homework

# 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: `outFile = open('average.txt', 'w')`
    - Replaces contents!
- Write file:
  - ▣ Syntax: `<filevar>.write(<string>)`
- Close file:
  - ▣ Syntax: `<filevar>.close()`
  - ▣ Example: `outFile.close()`

# File Reading in Python

- Open file: `inFile = 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: `inFile.close()`
  
- Create a program that reads and prints itself.

# A “Big” Difference

- Consider:

- `inFile = open ('grades.txt', 'r')`  
`for line in inFile.readlines():`  
    `# process line`  
`inFile.close()`

- `inFile = open ('grades.txt', 'r')`  
`for line in inFile:`  
    `# process line`  
`inFile.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 and file processing
  - ▣ Practice using the robots
- On Angel
  - ▣ Lessons → Homework → Homework 5 →  
Homework 5 Instructions