CHARACTER STRINGS

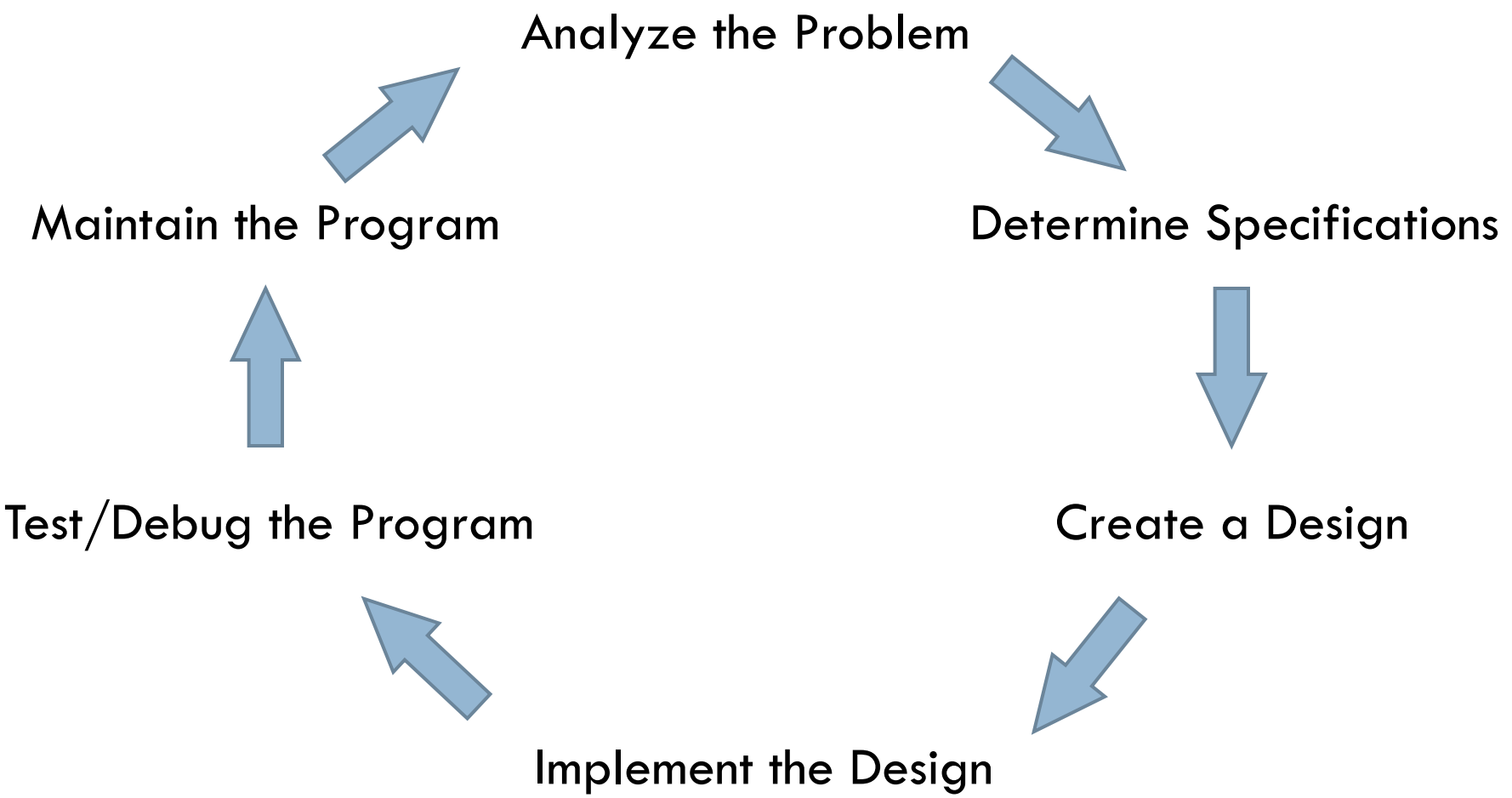
Bonus Points

- 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)

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

Strings (character strings)

- String literals (constants):
- `"One\nTwo\nThree"`
- `"Can't Buy Me Love"`
- `'I say, "Yes."  You say, "No." '`
- `"'A double quote looks like this \",' he said."`
- `"""I don't know why you say, "Goodbye,"`
  `I say "Hello." """`

String Operations

- Many of the operations listed in the book, while they work in Python 2.5, have been superseded by newer ones
- + is used for String concatenation: "xyz" + "abc"
- * is used for String duplication: "xyz " * 4
  - **>>> franklinQuote = 'Who is rich? He who is content. ' + 'Who is content? Nobody.'**
  - **>>> franklinQuote.lower()**

    'who is rich? he who is content. who is content? nobody.'
  - **>>> franklinQuote.replace('He', 'She')**

    'Who is rich? She who is content. Who is content? Nobody.'

    >>> franklinQuote.find('rich')

Strings as Sequences

- A string is an **immutable** sequence of characters
- >>> alpha = "abcdefg "
- >>> alpha[2]
- >>> alpha[1:4]
- >>> alpha[3] = "X" # illegal!

Strings and Lists

- A String method: **split** breaks up a string into separate words
  - `>>> franklinQuote = 'Who is rich? He who is content. ' + 'Who is content?  Nobody.'`
  - `>>> myList = franklinQuote.split()`
    `['Who', 'is', 'rich?', 'He', 'who', 'is', 'content.', 'Who', 'is', 'content?', 'Nobody.']`
- A string method: **join** creates a string from a list
  - `'#'.join(myList)`
  - `'Who#is#rich?#He#who#is#content.#Who#is#content?#Nobody.'`
- What is the value of `myList[0][2]`?
- Finish the exercises in **session04.py** that you downloaded last time.

Getting a string from the user

```
>>> name = input('Enter your name:')
Enter your name:John

Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    name = input('Enter your name:')
  File "<string>", line 1, in <module>
NameError: name 'John' is not defined
>>> name = raw_input('Enter your name: ')
Enter your name: John
>>> name
'John'
>>>
```

Q9, take a break

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>)

input() and raw_input() are related through the eval function

- Syntax:
  - eval(<string>)
- Semantics of eval
  - Input: any string
  - Output: result of evaluating the string as if it were a Python expression
- How does `eval` relate `raw_input` to `input`??

Consistent String Encodings

- Needed to share data between computers, also between computers and display devices
- Examples:
  - ASCII—American Standard Code for Info. Interchange
    - "Ask-ee"
    - Standard US keyboard characters plus "control codes"
    - 8 bits per character
  - Extended ASCII encodings (8 bits)
    - Add various international characters
  - Unicode (16+ bits)
    - Tens of thousands of characters
    - Nearly every written language known

String Formatting

- The % operator is *overloaded*
  - Multiple meanings depending on types of operands
- What does it mean for numbers?
- Other meaning for <string> % <tuple>
  - Plug values from tuple into "slots" in string
  - Slots given by *format specifiers*
  - Each format specifiers begins with % and ends with a letter
  - Length of tuple must match number of slots in the string

Format Specifiers

- Syntax:
  - %<width>.<precision><typeChar>
- Width gives total spaces to use
  - 0 (or width omitted) 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, rounding if needed.
- TypeChar is:
  - f for float, s for string, or d for decimal (i.e., int) [can also use i]
- Note: this RETURNS a string that we can print
  - Or write to a file using write(string), as you'll need to do on the homework 7assignment (HW7)

Q13-14, submit quiz

Begin HW5

- Although you have a reading assignment and Angel quiz, you are strongly encouraged to begin working on your homework early.
- If you have not completed the Eclipse-Pydev installation and configuration, you **must** do it before the next class session.
  - Instructions are in the HW5 document.