Objectives

- More: computer's representations of data types
- Encryption

Review

- How do we create formatted strings?
- What questions should we ask when creating formatted strings?
  - How would you format the table from the last slide on the handouts from last class?
- How does the computer represent data (e.g., numbers and text)?

Review: String Formatting

- There is a lot more you can do with string formatting
  - I presented a subset of the most commonly used functionality
- When formatting strings, consider
  - What is the data type of your data?
    - If a float, how many decimal places do you want?
  - How wide do you want the data to be?
  - What justification? Zero fill? Other flags?
- The answer to these questions help guide your creation of format specifiers

Example: Printing Out Tables

- A table of temperature conversions

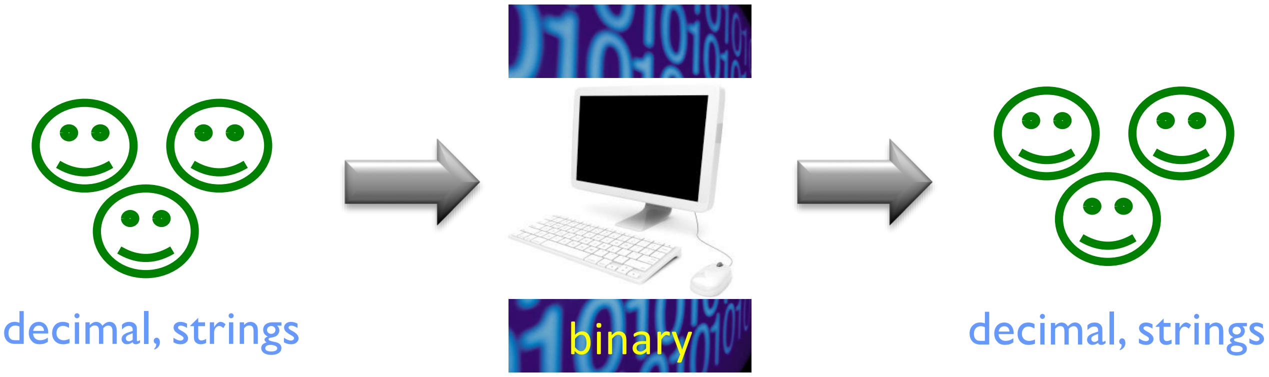
```
Temp F      Temp C      Temp K
------      ------      ------
-459.7      -273.1         0.0
   0.0       -17.8       255.2
  32.0         0.0       273.1
```

- If we want to print data in rows, what is the template for what a row looks like?
  - How do we make the column labels line up?
  - For above table, not as simple as using tabs. Why not?

`temp_table.py`

Review: Representations of Data

- Computer needs to represent different types of data
  - Eventually, all boils down to 1s and 0s
- Computer needs to translate between what humans know to what computer knows and back again



decimal, strings

binary

decimal, strings

String Representations

- A **string** is a *sequence* of characters
- Each character is stored as a binary number
- **ASCII** (American Standard Code for Information Interchange) is one standard encoding for characters
  - Limitation: ASCII is based on the English language
  - Cannot represent other types of characters
  - Handout is just a subset
- Unicode is a new standard – handles all languages

ASCII Table Handout

Translating to/from ASCII

- Translate a character into its ASCII numeric code using **built-in function** `ord`
  - `ord('a') ==> 97`
- Translate an ASCII numeric code into its character using **built-in function** `chr`
  - `chr(97) ==> 'a'`

`ascii_table.py`
`ascii.py`

ASCII Questions

- Lowercase letters are represented by what range of numbers?

- Uppercase letters are represented by what range of numbers?

- What is the difference between the decimal encoding of ‘M’ and ‘N’?
  - Between ‘m’ and ‘n’?

- Explain why `"Zebra" < "aardvarks"` evaluates to `True`

ASCII Questions

- Lowercase letters are represented by what range of numbers?
  - 97—122
- Uppercase letters are represented by what range of numbers?
  - 65—90
- What is the difference between the decimal encoding of ‘M’ and ‘N’?
  - Between ‘m’ and ‘n’?
  - 1
- Explain why `"Zebra" < "aardvarks"` evaluates to `True`
  - `ord("Z") < ord("a")`

Translating to/from ASCII

- Translate a character into its ASCII numeric code using **built-in function** `ord`
  - `ord('a’)` evaluates to `97`
- Translate an ASCII numeric code into its character using **built-in function** `chr`
  - `chr(97)` evaluates to `'a'`

`ascii_table.py`
`ascii.py`

Encryption

- Process of encoding information to keep it secure
- One technique: Substitution Cipher
  - Each character in message is replaced by a new character

Encryption: Caesar Cipher

- Julius Caesar used technique to communicate with his generals
- Replace letter with a letter X places away
    - X is called the ***key***

Original Letter	Key	Encrypted Letter
'a'	1	'b'
'b'	1	'c'
'z'	1	'a'

- “Wrap around” within the lowercase letters
- Write program(s) to do this in next lab

Caesar Cipher

- What would are the encoded messages?

Message	Key	Encoded Message
apple	5	
zebra	5	
the eagle flies at midnight	-5	

Caesar Cipher

Message	Key	Encoded Message
apple	5	fuuqj
zebra	5	ejgwf
the eagle flies at midnight	-5	ocz zvbgz agdzn vo hdyidbco

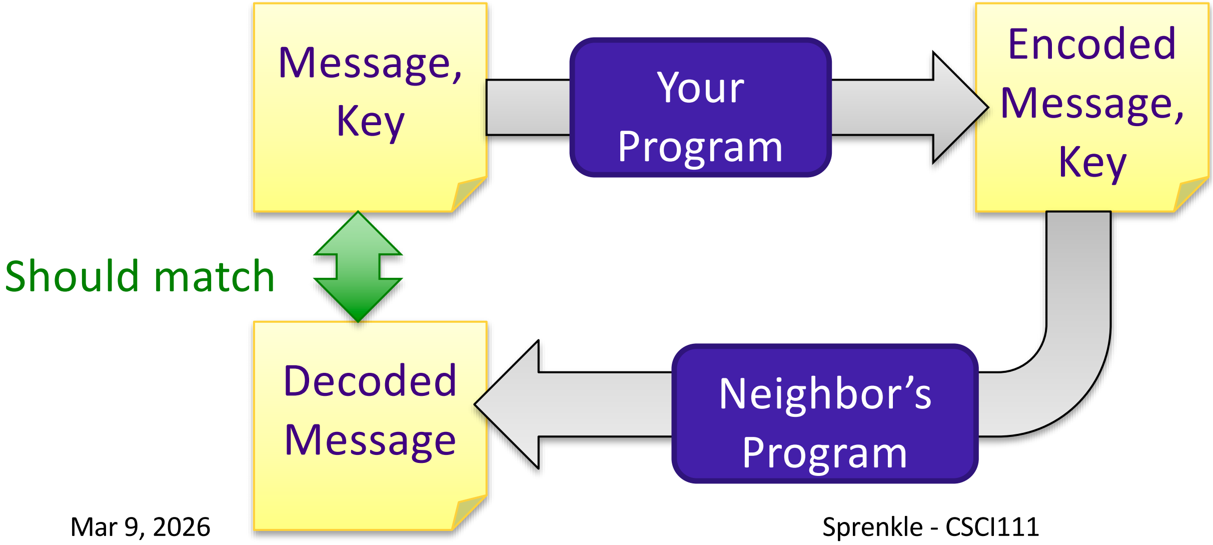
How would you ***decrypt*** an encrypted message?

Top-Down Design

- Break a problem into subproblems
  - Continue process until you reach “base problems” to solve

Next Lab

- Write an encoding/decoding program
  - Encode a message
  - Give to a friend to decode



What is your algorithm for the encoding process?
→ Break into pieces

Top-Down Design

1. Get user input for message and key
2. Check that the message and key are valid
3. Encrypt the message using the key
4. Output the encrypted message

Top-Down Design

1. Get user input for message and key
2. Check that the message and key are valid
3. Encrypt the message using the key
4. Output the encrypted message

Break this down: what happens in this step?

Top-Down Design: Encrypt Message

- Go through each character in the message and encrypt it

Top-Down Design: Encrypt Message

- Go through each character in the message and encrypt it

Encrypt Letter

- API: Takes a *lowercase letter* and a *key* as parameters and returns the encrypted letter
- Write test cases on white boards
- Write algorithm
- What are the preconditions for the function?

Top-Down Design: Encrypt Message

Original algorithm: Go through each character in the message and encrypt it

- Now that we have the `encrypt_letter` function, consider the algorithm and implementation of the `encrypt_message` function
  - What are good test cases?
  - What are the preconditions for the function?

Looking Ahead

- Pre Lab 7 due before lab
  - Shorter assignment
  - Some repetition with previous assignments, as we go into more depth on some topics
- Think about the encoding/encryption problem and how you will implement it
- Broader Issue: Cryptography and the Government