Objectives

- Computer's representations of data types

Big Step Forward

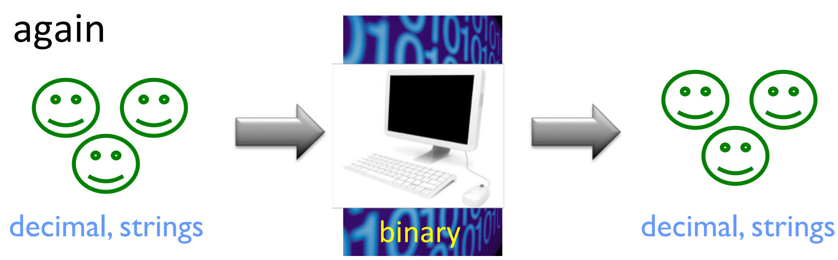
- Reflection: How far have I come in Computer Science?
- A lot of String operations
  - Previously: a lot of arithmetic operations, but you're familiar with those
- As we move forward, requires a lot more "play" and practice
  - Handouts and your notes help with review
  - Textbook

Pair Programming

- Getting the vocabulary down
  - Reinforcing the knowledge
  - Despite "ugh, I hate explaining"
- Frequent role switch
- Discussions of strategy
- Push each other

Representations of Data

- Computer needs ways 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



Seems like a divergence on strings but just wait…

Decimal Representations

- Decimal is base 10
- Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each *position* in a decimal number represents a power of 10

Decimal Representations

- Decimal is base 10
- Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Each *position* in a decimal number represents a power of 10
- Example: 54,087

5	4	0	8	7
10^4	10^3	10^2	10^1	10^0

- $= 5*10^4 + 4*10^3 + 0*10^2 + 8*10^1 + 7*10^0$
- = 5*10,000 + 4*1000 + 0*100 + 8*10 + 7*1

Number Representations

Characteristic	Decimal	Binary
Base	10	2
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1
Position represents	Power of 10	Power of 2

- Binary: two values (0, 1)
  - Like a light switch (either **off** or **on**) or booleans (either True or False)
- 0 and 1 are *binary digits* or **bits**
  - 64-bit machine: represents numbers (and other data) with 64 bits

Binary Representation

- Binary number: 1101

1	1	0	1
2^3	2^2	2^1	2^0

- $= 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$
- $= 1*8 + 1*4 + 0*2 + 1*1$
  - Decimal value: 13

Practice: what is the decimal value of the binary number **10110**?

Binary Representation

- Binary number: 10110

1	0	1	1	0
2^4	2^3	2^2	2^1	2^0

- $= 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$
- $= 1*16 + 0*8 + 1*4 + 1*2 + 0*1$
  - 22

Generalize this process into an algorithm.
Implement as function:
`binaryToDecimal(binaryNum)`

Algorithm: Converting Binary → Decimal

Accumulator design pattern

Given the binary number as a string

1. The starting exponent will be the length of the string-1
2. Initialize the result to zero
3. For each bit in the binary number
   - Multiply the bit by the appropriate power of 2
   - Add this to the result
   - Reduce the exponent by 1
4. Return the result

Implement algorithm
`binaryToDecimal.py`

Algorithm: Converting Decimal → Binary

Given the decimal as an integer…

1. Initialize the result to the empty string
2. Repeat until the decimal is 0:
   - result = str(decimal % 2) + result
   - decimal = decimal // 2
3. Display the result

Try out algorithm with 22 as input
Practice implementing this algorithm

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

ASCII Table Handout

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’?

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

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`

Encryption

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

Caesar Cipher

- Replace with a character X places away
  - X is the ***key***
- Julius Caesar used technique to communicate with his generals
- "Wrap around" within the lowercase letters
- Write program(s) to do this in next lab

Caesar Cipher

- Using the ASCII handout, what would be the encoded messages?

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

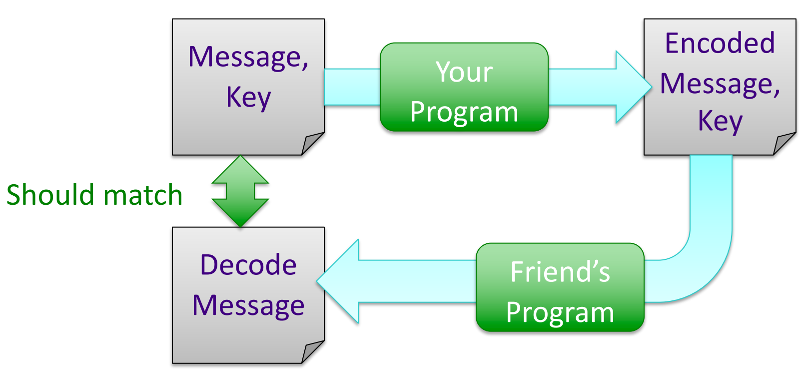
Caesar Cipher

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

What is your algorithm for the encoding process?
How would you *decode* an encrypted message?

Next Lab

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



Caesar Cipher (Partial) Algorithm

- For each character in the message
  - Check if the character is a space; if it is, it stays a space
  - Otherwise
    - Convert the character to its ASCII value
    - Add the key to that value
    - Make sure that the new value is a "valid" ASCII value, i.e., that that new value is in the range of lowercase letter ASCII values
      - If not, "wrap around" to adjust that value so that it's in the valid range
    - Convert the ASCII value into a character

This is a lot! Let's break it down…

Looking Ahead

- Lab 6 due Friday
- Broader Issue – App Data