Data

Numbers, text, and more
Data in Ruby
Reading: 1.1 -- 1.5

Models

In everyday usage a “model” is a small-scale representation of some object
- built by architects, artists, scientists, ...
- measuring or manipulating the model often leads to insights about the original object

Mathematicians and scientists also create models
- a mathematical model is an equation
  - parameters of the equation represent attributes of the system being modeled
- Examples:
  - an equation that predicts how far an object will fall as a function of time (and the gravitation constant \(g\))
  - the “two-parameter model” of molecular evolution that describes probabilities for different types of mutations in DNA

A computer program is also a model
- a computer’s memory holds a representation of real-world objects

Examples
- numbers: bank balance, mileage, age, id numbers
- text: e-mail, research papers, web pages
- images: 2D plots, drawings, photos
- sounds: voice recordings, music

A programmer’s job is to use constructs available in a programming language to create representations of real-world items
- a program used to calculate distance traveled by a falling object has representations for gravity \((g)\), time, and distance
- an electronic voting machine would need to represent voter registrations, candidate names, descriptions of ballot measures, ...

Data Representations
Bits

- Computer memories are binary devices
- Anything that has two states can potentially be used to store or transmit data
- Mechanical devices
  - beads on an abacus (up/down)
  - relay switches (open/closed)
  - paper tape or punch card (hole/not)

Binary Data

- We’ll let computer engineers figure out the physical storage of bits
- As computer scientists all we care about is that the memory device or communication channel distinguishes between two states: 0 and 1
- A single bit can represent any type of data that has only two values
  - true or false (e.g. answers on a test)
  - yes or no (e.g. votes on a referendum)
  - male or female (e.g. gender on a survey)
- To represent a piece of data that can have more than two values we need to use a collection of two or more bits

Bits (cont’d)

- Magnetic devices
  - core memories (clockwise/counter)
  - magnetic tape
- Electronic devices
  - integrated circuits (0v / +5v)
  - frequencies (low, high) in radio, telephone, ...
- Optical devices
  - use frequency or intensity of light

An Important Formula

- If a set has \( n \) items, the number of bits required to represent an element of the set is
  \[ k = \lceil \log_2 n \rceil \]
- each item in the set can be assigned a unique pattern of \( k \) bits

Examples:

- 4 nucleotides in DNA: 2 bits per letter
- a choice of 6 colors: 3 bits per color (two patterns unused)
- a lower case letter: 5 bits (6 patterns unused)
Corollary

- The inverse of the formula on the previous slide is also an important formula
  - a set of $k$ bits can represent up to $2^k$ different items
- Example: the ASCII code (commonly used for text files) is a 7-bit code
  - with 7 bits there are $2^7 = 128$ distinct combinations of 0s and 1s
  - sufficient to represent upper and lower case letters, digits, punctuation

<table>
<thead>
<tr>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100000</td>
<td>40</td>
<td>00</td>
</tr>
<tr>
<td>0100001</td>
<td>41</td>
<td>01</td>
</tr>
<tr>
<td>0100010</td>
<td>42</td>
<td>02</td>
</tr>
<tr>
<td>0100011</td>
<td>43</td>
<td>03</td>
</tr>
<tr>
<td>0100100</td>
<td>44</td>
<td>04</td>
</tr>
<tr>
<td>0100101</td>
<td>45</td>
<td>05</td>
</tr>
<tr>
<td>0100110</td>
<td>46</td>
<td>06</td>
</tr>
<tr>
<td>0100111</td>
<td>47</td>
<td>07</td>
</tr>
<tr>
<td>1000000</td>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>1000001</td>
<td>101</td>
<td>81</td>
</tr>
<tr>
<td>1000010</td>
<td>102</td>
<td>82</td>
</tr>
<tr>
<td>1000011</td>
<td>103</td>
<td>83</td>
</tr>
<tr>
<td>1000100</td>
<td>104</td>
<td>84</td>
</tr>
<tr>
<td>1000101</td>
<td>105</td>
<td>85</td>
</tr>
<tr>
<td>1000110</td>
<td>106</td>
<td>86</td>
</tr>
<tr>
<td>1000111</td>
<td>107</td>
<td>87</td>
</tr>
<tr>
<td>1001000</td>
<td>110</td>
<td>90</td>
</tr>
<tr>
<td>1001001</td>
<td>111</td>
<td>91</td>
</tr>
<tr>
<td>1001010</td>
<td>112</td>
<td>92</td>
</tr>
<tr>
<td>1001011</td>
<td>113</td>
<td>93</td>
</tr>
<tr>
<td>1001100</td>
<td>114</td>
<td>94</td>
</tr>
<tr>
<td>1001101</td>
<td>115</td>
<td>95</td>
</tr>
<tr>
<td>1001110</td>
<td>116</td>
<td>96</td>
</tr>
<tr>
<td>1001111</td>
<td>117</td>
<td>97</td>
</tr>
<tr>
<td>1010000</td>
<td>120</td>
<td>A0</td>
</tr>
<tr>
<td>1010001</td>
<td>121</td>
<td>A1</td>
</tr>
<tr>
<td>1010010</td>
<td>122</td>
<td>A2</td>
</tr>
<tr>
<td>1010011</td>
<td>123</td>
<td>A3</td>
</tr>
<tr>
<td>1010100</td>
<td>124</td>
<td>A4</td>
</tr>
<tr>
<td>1010101</td>
<td>125</td>
<td>A5</td>
</tr>
<tr>
<td>1010110</td>
<td>126</td>
<td>A6</td>
</tr>
<tr>
<td>1010111</td>
<td>127</td>
<td>A7</td>
</tr>
<tr>
<td>1011000</td>
<td>130</td>
<td>B0</td>
</tr>
<tr>
<td>1011001</td>
<td>131</td>
<td>B1</td>
</tr>
<tr>
<td>1011010</td>
<td>132</td>
<td>B2</td>
</tr>
<tr>
<td>1011011</td>
<td>133</td>
<td>B3</td>
</tr>
<tr>
<td>1011100</td>
<td>134</td>
<td>B4</td>
</tr>
<tr>
<td>1011101</td>
<td>135</td>
<td>B5</td>
</tr>
<tr>
<td>1011110</td>
<td>136</td>
<td>B6</td>
</tr>
<tr>
<td>1011111</td>
<td>137</td>
<td>B7</td>
</tr>
<tr>
<td>1100000</td>
<td>140</td>
<td>C0</td>
</tr>
<tr>
<td>1100001</td>
<td>141</td>
<td>C1</td>
</tr>
<tr>
<td>1100010</td>
<td>142</td>
<td>C2</td>
</tr>
<tr>
<td>1100011</td>
<td>143</td>
<td>C3</td>
</tr>
<tr>
<td>1100100</td>
<td>144</td>
<td>C4</td>
</tr>
<tr>
<td>1100101</td>
<td>145</td>
<td>C5</td>
</tr>
<tr>
<td>1100110</td>
<td>146</td>
<td>C6</td>
</tr>
<tr>
<td>1100111</td>
<td>147</td>
<td>C7</td>
</tr>
<tr>
<td>1101000</td>
<td>150</td>
<td>D0</td>
</tr>
<tr>
<td>1101001</td>
<td>151</td>
<td>D1</td>
</tr>
<tr>
<td>1101010</td>
<td>152</td>
<td>D2</td>
</tr>
<tr>
<td>1101011</td>
<td>153</td>
<td>D3</td>
</tr>
<tr>
<td>1101100</td>
<td>154</td>
<td>D4</td>
</tr>
<tr>
<td>1101101</td>
<td>155</td>
<td>D5</td>
</tr>
<tr>
<td>1101110</td>
<td>156</td>
<td>D6</td>
</tr>
<tr>
<td>1101111</td>
<td>157</td>
<td>D7</td>
</tr>
<tr>
<td>1110000</td>
<td>160</td>
<td>E0</td>
</tr>
<tr>
<td>1110001</td>
<td>161</td>
<td>E1</td>
</tr>
<tr>
<td>1110010</td>
<td>162</td>
<td>E2</td>
</tr>
<tr>
<td>1110011</td>
<td>163</td>
<td>E3</td>
</tr>
<tr>
<td>1110100</td>
<td>164</td>
<td>E4</td>
</tr>
<tr>
<td>1110101</td>
<td>165</td>
<td>E5</td>
</tr>
<tr>
<td>1110110</td>
<td>166</td>
<td>E6</td>
</tr>
<tr>
<td>1110111</td>
<td>167</td>
<td>E7</td>
</tr>
<tr>
<td>1111000</td>
<td>170</td>
<td>F0</td>
</tr>
<tr>
<td>1111001</td>
<td>171</td>
<td>F1</td>
</tr>
<tr>
<td>1111010</td>
<td>172</td>
<td>F2</td>
</tr>
<tr>
<td>1111011</td>
<td>173</td>
<td>F3</td>
</tr>
<tr>
<td>1111100</td>
<td>174</td>
<td>F4</td>
</tr>
<tr>
<td>1111101</td>
<td>175</td>
<td>F5</td>
</tr>
<tr>
<td>1111110</td>
<td>176</td>
<td>F6</td>
</tr>
<tr>
<td>1111111</td>
<td>177</td>
<td>F7</td>
</tr>
</tbody>
</table>

Groups of Bits

- A byte is a collection of 8 bits
  - before the 1970s the number of bits in a byte varied from system to system
  - there were 6-, 7-, 8-bit bytes
  - by the 1980s the term had become standard
- A central processing unit (CPU) operates on several bytes at a time
  - a word is a collection of two or more bytes
  - typical word sizes are 32 bits (4 bytes) and 64 bits (8 bytes)
- Memory capacity is often described in terms of megabytes or gigabytes

<table>
<thead>
<tr>
<th>prefix</th>
<th>kilo-</th>
<th>mega-</th>
<th>giga-</th>
<th>tera-</th>
<th>peta-</th>
</tr>
</thead>
<tbody>
<tr>
<td>meaning</td>
<td>$10^3$</td>
<td>$10^6$</td>
<td>$10^9$</td>
<td>$10^{12}$</td>
<td>$10^{15}$</td>
</tr>
<tr>
<td>abbreviation</td>
<td>KB</td>
<td>MB</td>
<td>GB</td>
<td>TB</td>
<td>PB</td>
</tr>
</tbody>
</table>

Integers

- The number of bits in a word determines the range of numbers a processor can work on
  - integers are stored in words
  - if a CPU has $n$-bit words, there are $2^n$ patterns
- When a number is stored in memory, the bit pattern corresponds to the string of digits in the binary form of the number
  - example: the pattern 00100101 represents the number 37
  - The $2^n$ patterns in an $n$-bit word can represent the numbers from 0 to $2^n$ - 1
- in a 32-bit system the numbers range from 0 to 4,294,967,295

Integers

- What happens if a program has an expression that generates a result outside this range?
  - the result is an overflow
- Depending on the computer hardware, operating system, and programming language, evaluating the expression $2^{26} + 100$ on an 8-bit machine might
  - a) evaluate correctly to 336 (by changing the representation to use two words),
  - b) generate a runtime error ("integer overflow at ...."), or
  - c) generate 70 as the result and continue without an error message
- Why 70?
  - the binary pattern for 336 is 101000110, which is 9 bits long
  - the system ignores the bit on the left, so the pattern using 8 bits is 01000110
  - 01000110 is the pattern for the number 70 ($64 + 4 + 2$)
Summary: Integers

- The example on the previous slide might seem contrived
- But the basic principle is:
  - all CPUs have a finite word size (typically 32 or 64 bits in a modern system)
  - the word size defines the range of integer values (0 to $2^n - 1$)
- Software might generate a result outside this range
- What happens when a result is outside the range depends on
  - the CPU (a program will behave differently on 64-bit machines and 32-bit machines)
  - the O/S or other software settings (error or no error message)
  - the language (C++ and Ruby do different things)

Other Numbers

- Topics covered in the text (and maybe in future lectures):
  - representing positive and negative numbers (“two’s complement”)
  - “floating point” representations for fractions (e.g. 3.14159)
  - implications of the representation for fractions: overflow, underflow, roundoff errors

Numbers in Ruby

- Ruby has two basic types of numbers
  - integers
  - real numbers
- If you write a sequence of digits in an expression Ruby assumes you want it to be an integer, e.g.
  
  ```ruby
  >> x = 125
  >> x
  125
  ```
- Ruby calls these “fixnums”, for “fixed precision numbers”
  - the “fixed precision” means a finite number of bits used in the representation
  ```ruby
  >> x = 2**14
  => 16384
  >> x
  16384
  ```

Numbers in Ruby

- When an integer is too big to fit into one machine word Ruby automatically converts it to an arbitrary precision representation
  ```ruby
  >> y = 2 ** 40
  => 1099511627776
  >> y
  1099511627776
  => Bignum
  >> z = 2**100
  => 126765060028228229401496703205376
  ```
- Bignums can be pretty big....
Numbers in Ruby

- Fixnums and Bignums are both types of integers
- If you include a decimal point in a number Ruby makes an object that it calls a “Float”

```
>> x = 3.14159
=> 3.14159
>> x.class
=> Float
>> y = 4.0
=> 4.0
>> y * 5
=> 20.0
```

Numeric Computing

- The modern era of computing machines (1945–) was initiated by the requirements for large scale “number crunching”
  - calculation of ballistic tables and other military applications
- Scientific computing is an important part of computer science today
  - modeling and simulation
  - computational science
  - engineering and design

Characters

- We’ve already seen how single characters can be represented by ASCII codes
  - originally a 7-bit character set
  - representation for 32 “control characters” and 96 printing characters
  - letters, digits, punctuation, etc
  - search for ASCII at [http://wikipedia.org](http://wikipedia.org)
- Later extended to an 8-bit code to include many other symbols
  - math symbols, Greek letters, ...
Strings

- In computer science terminology, a **string** is a sequence of 0 or more characters.
- Two common representations for strings:
  - use a special “end of text” marker (e.g. \0) to denote the last character
  - prefix the string with a byte that defines how many characters are in the string

Strings in Ruby

- When you type a group of characters between quotes Ruby creates a string
  ```ruby
  >> s = "Bonjour"
  "Bonjour"
  >> s.class
  => String
  ```
- There are operators that do calculations with strings
  - Example: the `+` operator concatenates two strings
    ```ruby
    >> t = "Je m'appelle John"
    "Je m'appelle John"
    >> s + t
    "BonjourJe m'appelle John"
    >> s + ". " + t
    "Bonjour. Je m'appelle John"
    ```

Methods

- Most operations on strings are performed by **methods**
- To apply a method to an object, write the object’s name, a period, and the method name
  ```ruby
  >> s
  "Bonjour"
  >> s.length
  7
  >> s.reverse
  "ruojnoB"
  ```
- There are dozens of methods for the String class
  - if you are curious read about them in the online documentation
  - if you need a method for a lab project we’ll include a description

Elements in a String

- The **index operator** is used to access parts of a string
- If a string `s` has `n` characters, the expression
  ```ruby
  s[i]
  ```
  returns the ASCII code of the `i`th character (where `0 \leq i \leq n-1`)
- Important note: the first character in `s` is `s[0]`
  ```ruby
  >> s = "Hello, world"
  "Hello, world"
  >> s[0]
  72
  >> s[1]
  101
  ```
Elements in a String

- You can also modify a string with the [] operator
  ```ruby
  s = "Gilbert"
  s[0] = 68
  s
  => "Dilbert"
  ```

- If you don’t want to be looking up codes all the time (or don’t want to memorize the table) Ruby has a shortcut:
  ```ruby
  s[0] = ?D
  => 68
  ```

String Search

- The index method does a string search
  ```ruby
  s = "Now is the time for all good men to...."
  s.index("men")
  => 29
  s.index("droids")
  => nil
  ```

- Don’t forget the first index is 0
  ```ruby
  s.index("Now")
  => 0
  ```

String Processing Example

- One last example (for now) of operations on strings
  ```ruby
  s = "abcdefg"
  s.insert(3,"*")
  => "abc*defg"
  s = "Now is the time for all good men to...."
  s.insert(s.index("men"),"wo")
  => "Now is the time for all good women to...."
  ```

Files, Web Pages, and Other Text

- Large pieces of text are often represented in a program as one long string
  - DNA sequences (e.g. the E. coli genome is 4.6M DNA letters)
  - text files (e.g. documents or data files)
  - web pages (plain text plus “markup” symbols such as `<table>..</table>`) 

- Ruby has classes and methods that will create and operate on large strings
  - Example: to get scores from the NCAA men’s basketball tournament use a URL like this:
    ```ruby
    ```
  - Ruby code that fetches this page:
    ```ruby
    url = "www.ncaa.com"
    base = "/basketball-mens/default.aspx?id=114&date="
    http = Net::HTTP.new(url)
    page = http.get(base + "20080405")
    ```
Images

- Digital photos are made up of **pixels**
  - A scanner or digital camera divides the image into a regular grid
  - Each cell in the grid is given a single color
- The result is a like a mosaic
  - When viewed from a distance, the cell boundaries disappear

Pixel = picture cell

Image Quality

- The realism of an image depends on two factors
  - **Resolution** refers to the number of pixels in the image
    - A high resolution image has more pixels
    - Can use more pixels per area for higher quality (think of a mosaic with smaller tiles)
    - Can also extend the size of an image (e.g., 10 megapixel camera vs 3 megapixel camera)
  - **Bit depth** refers to the number of bits used to represent each pixel
    - 1 bit: black or white (e.g., fax machine)
    - 8 bits: shades of gray
    - 24 bits: 8 bits each for intensity of three primary colors (red, green, and blue)

Image Formats

- A file containing the exact representation of each pixel is known as a **bitmap** or a **pixmap**
  - Some digital cameras will produce “raw image” data
  - Upload these to Aperture or other fancy photo editing software
- It is possible to condense an image by removing redundant information
  - Example: if the same color occurs often, save one copy of the pixel and a “map” of where it is found
- There are several methods for compressing images (GIF, JPG, TIFF, etc)
  - Digital cameras compress images so they are much smaller when they are saved on a memory card or uploaded to a computer
- To learn more: search Wikipedia for “pixel” and “image file formats”

Vector Graphics

- An alternative to representing an image as a set of pixels is **vector graphics**
  - An image consists of a set of objects
  - The file that stores the image contains representations of each object
    - X, y coordinate, size, shape, color, etc
- Many software applications let users create drawings as a set of objects
Music

- Sounds we hear are produced by compression waves in air
- A microphone transforms pressure waves into electronic waves
- Electronic waves can be
  - amplified
  - turned into electromagnetic (radio) waves
  - recorded on tape or other media
- Electronic waves are analogs of the pressure wave
  - analog = analogy

Analog-to-Digital Converter

- An analog-to-digital converter can produce a pattern of bits from an electronic wave
- the ADC “samples” the audio wave at periodic intervals
- the magnitude of the wave is converted into an integer
- A digital recording is a sequence of samples taken from the input source
- A digital-to-analog converter reverses the process, turning a sequence of numbers into an electronic wave

Music Quality

- Two factors determine the quality of a digital recording
- The sampling rate determines the amount of time between samples
  - measured in Kilohertz (thousands of samples per second)
  - higher rate = more samples = higher quality
- Resolution refers to the number of bits used for each sample
  - the picture shows a resolution of 4 bits (numbers between 0 and 15)
  - Digital CDs are made with a sampling rate of 44KHz and 16-bit resolution

Summary

- Information is stored in a computer in the form of bits
  - computer scientists are (usually) not concerned about the physical storage medium
  - we only care that something has two states that we can label 0 and 1
- A single piece of data is represented by a sequence of bits
  - $n$ bits $\Rightarrow 2^n$ different patterns
  - for integers, $n$ bits can represent the numbers $0..2^n-1$
- Ruby has representations for numbers and text
  - Fixnum, Bignum, Float
  - Strings (where individual elements are ASCII characters)

The word “resolution” was used to describe digital images, also. Is it describing the same concept in digital music recording?
Summary

- Images and music are stored in a computer by “digitizing”
  - a digital camera or scanner creates a set of pixels
  - an AD converter creates a series of audio samples
- In both cases the quality of the digital version depends on the number of bits in the representation
  - image quality depends on spatial resolution (number of pixels) and bit depth (number of bits per pixel)
  - audio quality depends on sampling rate (number of samples per second) and resolution (number of bits per sample)
- See Question 4 on page 78 of Brookshear (the Social Issues section at the end of Chapter 1)
  - what are the consequences of digitizing information? is loss of quality inevitable?

A Look Ahead

- In a few weeks we’ll look at techniques for adding more information to the representation of data
  - the information in the additional bits will allow us to do error detection
  - we will look at algorithms that scan a set of bits and say “an error occurred (one or more of these bits changed) since the data was stored” (or since the data left a transmitter)
  - if we add enough information the algorithm can even fix the error
- Probably a good requirement for a system that counts votes and transmits election results.....