String Data

DRAFT

- In computing, much data is plain text, so you will be dealing with *strings*. A "string" is a sequence of ***zero or more*** characters -- though usually 1+ *characters*
- You will use strings often, in different ways:
  - Printing as output
  - Fetching as input
  - Comparing
  - Reversing
  - Converting to/from other types
- Work and practice to become comfortable with this type and its many uses

Characters

- The *ASCII character set* is older and smaller (***8-bit***) than Unicode, but is still quite popular (in C programs)

- The ASCII characters are a subset of the Unicode character set, including:

uppercase letters	**A, B, C, …**
lowercase letters	**a, b, c, …**
punctuation	**period, semi-colon, …**
digits	**0, 1, 2, …**
special symbols	**&, |, \, …**
control characters	**carriage return, tab, …**

- See, for example, **`http://www.asciitable.com`**

Characters

- Each character, however, will correspond to an *integer value* in some *character set*, and there are methods to perform conversions.
- The following example uses *Python* based methods
  - ➢ Integer to character: `chr`
  - ➢ Example: `chr(97)` → `a`
  - ➢ Character to integer: `ord`
  - ➢ Example: `ord('a')` → `97`
- This can be useful when you want to do arithmetic with characters, for example.

Character Sets

- A *character set* is an ordered list of characters, with each character corresponding to a unique number
- Much software today uses the *Unicode character set*
- The Unicode character set uses sixteen bits per character, allowing for **65,536 (2^16)** unique characters
  - Unicode character values are often expressed as quartets of hex digits (4 hex digits equating to 16 bits), such as **Θ** (*Char #920, or* **`0398`** *in hex*)
  - It is an international character set, containing symbols and characters from many world languages
  - Obviously, this is much more expansive than the ASCII character set!
- Reference: **`https://unicode-table.com/en/`**

Character Encodings

- A *character encoding*, in contrast, deals with how characters (within a set) are to be represented in other, non-character forms: numerical, electrical, etc..
- For example, in computing, all data is encoded as bytes, which are made of ***bits*** **:** ones and zeroes
- A character encoding will entail
  - a sequence of bits
  - for each character
  - within some character set

Character Encodings

- An encoding for just `ASCII` characters would pretty simple, because char values are limited to *zero* through *127*, which requires only 7 bits (i.e., no more than 1 byte) per character
- For larger sets of characters, more bits would be needed.
- There are some other character encodings, more expansive than `ASCII` but still representable with 1 byte per character, such as `ISO-8859-1` (a.k.a., `Latin-1`)
- `Latin-1` is "ASCII-based" but includes a wider range of characters, such as accented vowels

Character Encodings

- An encoding for just `ASCII` characters would pretty simple, because char values are limited to *zero* through *127*, which requires only 7 bits (i.e., no more than 1 byte) per character
- For larger sets of characters, more bits would be needed.
- There are some other character encodings, more expansive than `ASCII` but still representable with 1 byte per character, such as `ISO-8859-1` (a.k.a., `Latin-1`)
- `Latin-1` is "ASCII-based" but includes a wider range of characters, such as accented vowels

Character Encodings

- An encoding for the Unicode character set would, in theory, entail two bytes (16 bits) per character.
- However, that could end up consuming space in memory, when frequently-used characters end up requiring the same space as rarely-used characters
- Also, if a character has a low numerical value, then many of its leading bits would be all zeroes (to make up the whole 16 bits)
- Fortunately, however, a character encoding -- the concrete representation of characters from a set -- can be designed *intelligently*

Character Encodings

- The most popular (currently) encoding for the Unicode character set is UTF-8
- In UTF-8, a character is represented using from one to four bytes
- For any particular character, some of its leading bits will signal whether it is going to take up one, two, three, or four bytes
  - Characters *zero* through *127* are 1 byte: **0***xxxxxxx*
  - Chars 128 through 2047 are 2 bytes: **110***xxxxx* **10***xxxxxx*
- This allows for more efficient usage of space for storing characters as textual data

Character Encodings

- The catch is that one must be mindful, to some extent, about which encoding is being used to…
  - *Write* the text *to* storage as bytes
  - *Read* the bytes *from* storage as text
  - (Here, consider "write" and "read" as roughly analogous to "save" and "open" -- in that they involve operations to and from disk)
- There are usually default encodings (within a program) for both writing and reading textual data
- These may be subject to user preference, to some extent

Escape Sequences

- What if we want to include the quote character itself?
- The following line would confuse the interpreter because it would interpret the two pairs of quotes as two strings and the text between the strings as a syntax error:

```
print ("I said "Hello" to you.")
```

- One option would be to replace the beginning and ending double-quote symbols with single-quotes:

```
print ('I said "Hello" to you.')
```

- The *reverse* would also be valid

```
print ("I said 'Hello' to you.")
```

Escape Sequences

- Another option is to use *escape sequences*, which are character combinations that have a special meaning within a string
- Some Escape Sequences:

Escape Sequence	Meaning
`\t`	tab
`\n`	newline
`\r`	carriage return
`\"`	double quote
`\'`	single quote
`\\`	backslash

Example:

```
print ("Hello,\n\tworld")
```

```
Hello,
   world
```

Understanding and Working With Data

- No matter which route you take in the IT field, you will be dealing with data, in some form
- You can go with this definition for now: Data are pieces of information about the real world…
  - That are gathered and maintained -- as well as…
  - Made expressible and readable in some form(at) or another
  - For one or more purposes:
    - Knowledge
    - Analysis
    - Decision-making
    - Reporting
    - Interpretation
    - Problem-solving!
- There are many kinds of data….

Numeric Data

- First, we have two type of ***real numbers***:
  - ***Integers*** are whole numbers (No fractional component):

    7, **-358**, **0**, **-10**, **12398**
  - ***Decimals*** (or "floating-point") numbers do have a fractional component: **7.6**, **-35.8**, **-1.09**
- A ***complex number*** has an imaginary component.
  - In other words, some non-zero multiple of the constant i
  - We define i as the square root of **-1**
  - We call i "imaginary" because no two real numbers can be squared to produce a negative result

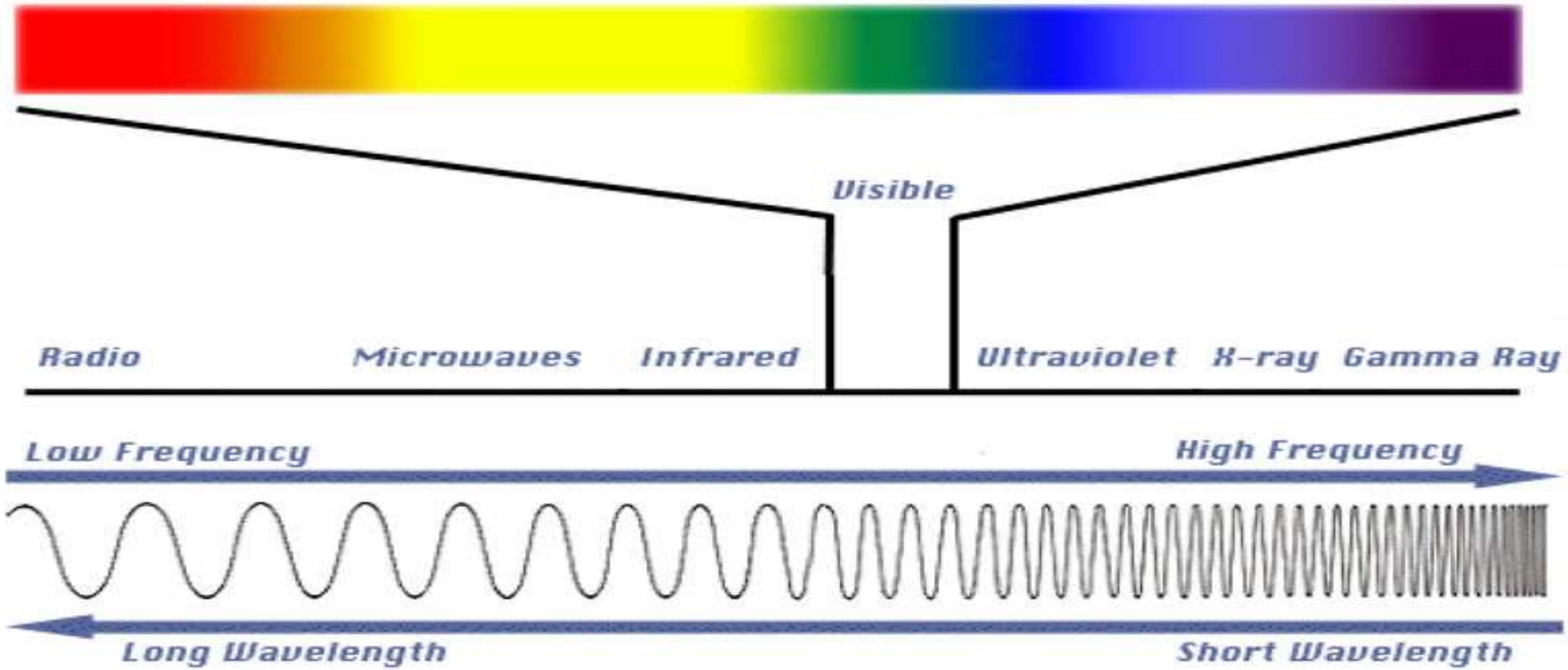
<u>Boolean Data</u>

- A **`boolean`** datum (*<u>singular of "data"</u>*) can have either of two possible values: **`True`** or **`False`**
- This is applicable to many ***either/or*** scenarios:
  - ***<u>Yes</u>*** or ***<u>No</u>***
  - **`1`** or **`0`**
  - ***<u>Open</u>*** or ***<u>Closed</u>***
  - ***<u>Up</u>*** or ***<u>Down</u>***
  - ***<u>On</u>*** or ***<u>OffA</u>***

EMR, and The Science Behind It...

- The behavior of fiber optic cabling is based upon the transmission of ***electromagnetic radiation (EMR)***:
  - What is EMR?
  - What are some technologies that make use of it?
    - Radios
    - Microwave Ovens
    - X-Ray machines
- You will hear the term "light" used much more generally to refer to EMR – versus our more common definition of *visible* light.
- These varieties of EMR are all located on the ***electromagnetic spectrum***.

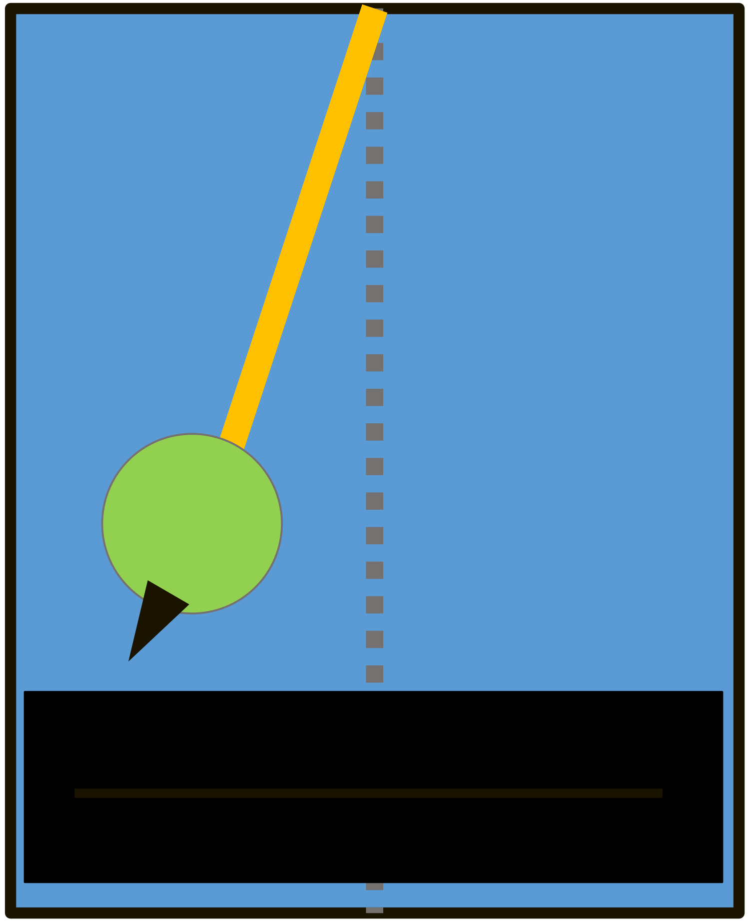
The EM Spectrum



Video: `https://www.youtube.com/watch?v=cfXzwh3KadE`

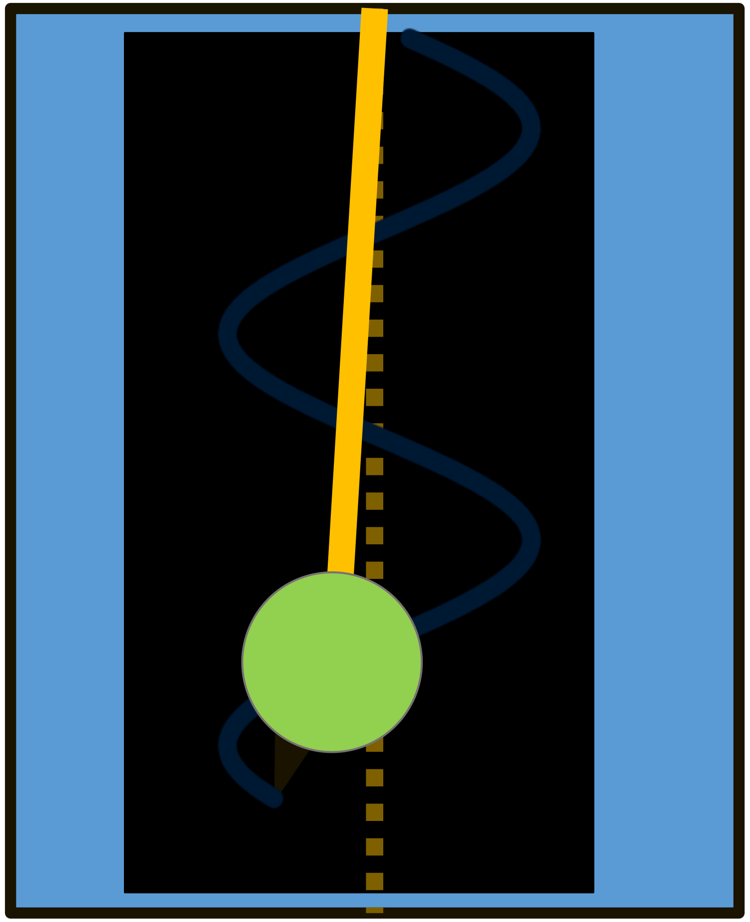
The EM Spectrum

- EMR is generally conceptualized as ***waves*** – cyclic variations about a "center", in which energy is transferred. (In this case, the energy comes from various forms of EM activity.)
- To envision this, imagine a ***pendulum*** swinging to and fro…
  - With a *pencil lead* at the lowest point
  - In constant contact with *paper*
- If the paper is *stationary*, all the marks will remain within a single, one-dimensional space.



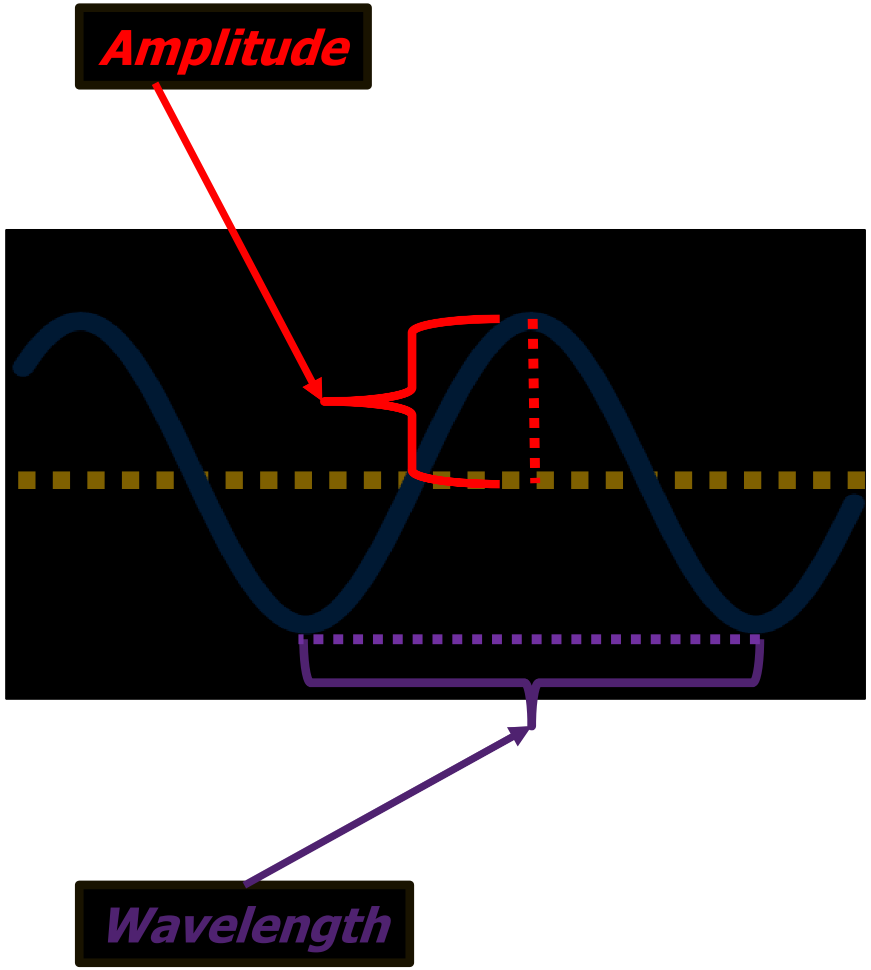
The EM Spectrum

- However, if the paper – here being of indefinite length – is moving…
  - in one direction
  - at a constant speed
  - perpendicular to the pendulum's plane of motion
- …then you will see a graph of the pendulum's position, relative to the center, *over time*
- In other words, a ***wave***



The EM Spectrum

- A wave – of EM radiation or another type – will feature a number of properties:
  - ***Amplitude* (A):** The height of a wave, from the center. **Unit:** ***meters***.
  - ***Wavelength* (λ):** The distance between two analogous points on the wave graph. **Unit:** ***meters***.
  - ***Frequency* (f):** The number of waves passing a given point during a given time. **Unit:** ***Hertz*** or s^{-1}
  - ***Wave speed* (v):** wavelength multiplied by frequency. **Unit:** $m*s^{-1}$ or m/s



The EM Spectrum

- We will encounter rather large and small numbers, that would normally have a lot of zeros.
- This would quickly become confusing, so we use a method called ***scientific notation*** to simplify this problem.
  - 300000000 = $\mathbf{3x10^{8}}$
  - .0000000000000000000000602 = $\mathbf{6.02*10^{-23}}$
  - In computing context, these may be expressed as **3e8** and **6.02e-23**, respectively
  - We have *prefixes* for the different exponents of 10…

MULTIPLICATION FACTOR	PREFIX	SYMBOL
1 000 000 000 000 000 000 = 10^{18}	exa	E
1 000 000 000 000 000 = 10^{15}	peta	P
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto	h
10 = 10^{1}	deka	da
0.1 = 10^{-1}	deci	d
0.01 = 10^{-2}	centi	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	m
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

The EM Spectrum

- In a vacuum, light (i.e., electromagnetic radiation) travels at a speed of $3x10^8$ m/s (or $m*s^{-1}$).
- **Recall:** v = f * λ. Therefore, if you know a signal's frequency, then you can calculate its ***wavelength*** by dividing the ***speed of light*** **(c)** by the ***frequency***. **λ = c / f**
- Consider WBZ, a Boston radio station broadcasting at a frequency of 1030 kHz – or 1.03 Mhz:

$$\lambda = \frac{3.00 \times 10^{8}\ m*s^{-1}}{1.03 \times 10^{6}\ s^{-1}}$$

→

$$\lambda = \frac{3.00 \times 10^{\not{8}\,2}\ m*\not{s^{-1}}}{1.03 \times 10^{\not{6}}\ \not{s^{-1}}}$$

The EM Spectrum

$$\lambda = \frac{3.00 \times 10^{8\,2}\ \text{m} \cdot \text{s}^{-1}}{1.03 \times 10^{6}\ \text{s}^{-1}} \rightarrow \lambda = \frac{3.00 \times 10^{2}\ \text{m}}{1.03} \rightarrow \lambda \approx 2.91 \times 10^{2}\ \text{m}$$

- In doing this, it is important to keep track of your units, including which ones combine or cancel out in the arithmetic.
- Understanding these concepts will be helpful not only for this chapter but also for subsequent chapters, such as wireless networking, which also uses EMR

The EM Spectrum