CptS 111, Spring 2023
Lect. #1, Jan. 9, 2023
Class Notes

Today's Agenda:

0. Jupyter Notebook
1. How to succeed in CptS 111
2. A little about computer programs and how we communicate with computers
3. Compiled vs Interpreted programming languages
4. A little about programming environments (IDEs)
5. Simple code for getting input and showing output (`input()`, `print()` functions)
6. A quick tour of Canvas, zyBooks, and our class website

1. How to succeed in CptS 111

1. Do all the reading/activity assignments (preferably before class; access via Canvas).
2. Attend all 12 labs (2 make-up labs).
3. Do all 7 programming assignments.
4. Ask *right away* if you don't understand something (me, a classmate, your TA, a VCEA tutor). TAs and I have office hours. Take advantage of them.
5. Practice programming the same way you practice the piano or a foreign language, i.e., practice often (a good time is while you're doing the reading assignments), and memorize syntax and commands.
6. Persevere! It's hard when you can't figure things out, especially when everyone else seems to be doing okay, but don't think you're not "smart" enough! That's a fixed mindset. ***THINK GROWTH MINDSET, i.e., if you practice and work hard and smart, you'll learn*.** It's like training for a marathon or strength training--you have to push through the "pain" to make progress.
7. DON'T FALL BEHIND !!!

2. Computer Programs (AKA Code, Applications, Scripts for Interpreted Languages)

- Everything running on a computer or any other smart device is a program:
  - operating system (Linux, Windows, macOS)
  - search engines
  - games
  - Microsoft 365
  - gmail
  - iClicker Student app
  - Instagram
- Programs run other programs.
- At the hardware level, a CPU (central processing unit) is made up of a huge number of transistors connected together to perform operations in binary.
  - a transistor works as a binary switch which is either on or off
  - binary means 2; in computerese binary means 1 or 0 (on or off)
  - current record for number of transistors on a single chip: 2,600,000,000,000 (2.6 trillion) or 325GB
- We communicate with computers in binary, i.e., write programs written in binary:
  - binary digit = bit ('b' from binary and 'it' from digit)
  - 8 bits = byte
  - 1 bit = 2 unique items
  - 2 bits = 4 unique items
  - 3 bits = 8 unique items
  - n bits = 2^n unique items

How many bits do we need to represent the decimal number 6? The answer is 3.

Suppose n = 3. Then we can describe 8 unique items ($2^3 = 8$):

```
0 0 0  -> 0 in binary
0 0 1  -> 1 in binary
0 1 0  -> 2 in binary
0 1 1  -> 3 in binary
1 0 0  -> 4 in binary
1 0 1  -> 5 in binary
1 1 0  -> 6 in binary
1 1 1  -> 7 in binary
```

As you can see, the binary numbers above represent the decimal numbers 0 through 7.

- We can use binary not just for numbers but to represent everything needed to program: characters, colors, numbers, sounds, images.
- However, imagine writing a computer program using something like:

000101000100101000101001010101010001000100001010010001000100000000111011010000
111000101001010001001001101011100100100010100100010001000010101110010010 0
110001010010010001000100000000111101010010001010100010100010000100101000110100
111000101010010010001000100001001010011010111001001001000100101010001000100001
101001001101011100100100010100100111010100000100010000010101010010001000100001

- This might work for a computer, but it's definitely not human-friendly!
- Instead we use high-level languages such as Python, C, and Java.
- We use other programs to convert our Python, C, or Java programs into binary.

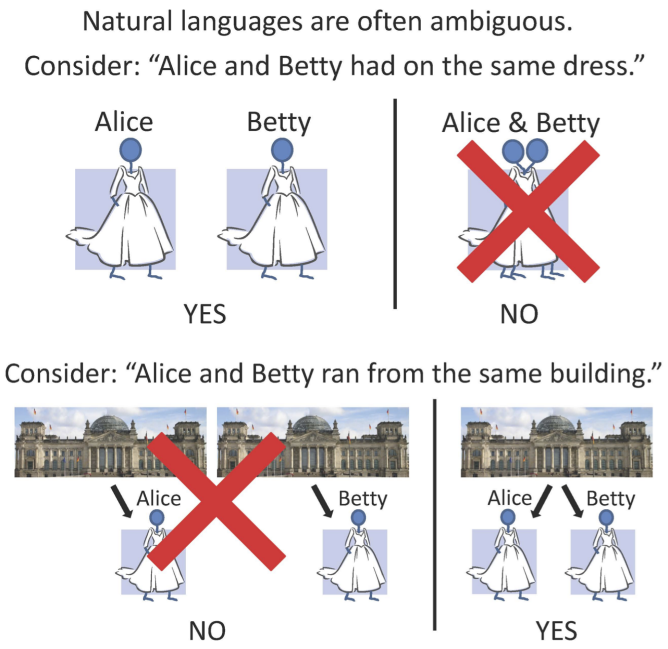
Note also something that you may not have realized before: Any integer number can be converted exactly into a binary number. Consider, for example, the number 42 which is the sum of 32, 8, and 2. In binary,

32 => 2^5 => 100000
8 => 2^3 => 1000
2 => 2^1 => 10
42 => 101010

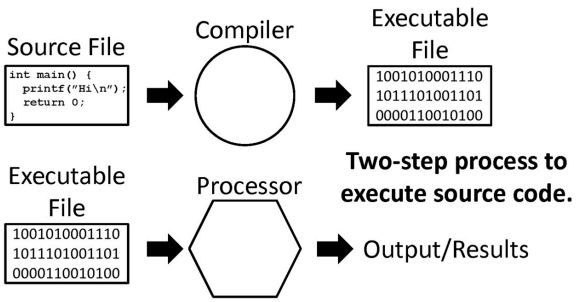
Why is this important in programming? Because it means we can represent integers, regardless of their size, *exactly*.

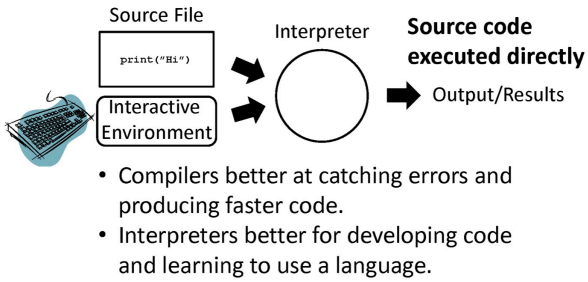
3. Computer Programming Languages

- Programming languages provide the framework for humans to communicate with computers in a way that is **almost** natural to humans.
- However, **computers won't tolerate even a single typo**, and programming languages all have to follow strict syntax (word structure) and rules. **You need to memorize these!**
- Computers can't tolerate ambiguity or mistakes the way we can. Thus, statements (instructions) must be totally unambiguous and without mistakes. **You'll understand this when you start coding!**



- For now we'll consider two types of programming languages: compiled and interpreted.





- C is a compiled language.
- Python is an interpreted language.
  - very readable
- Goal of good Python programming is to be "pythonic"; write code that's terse but readable and easy to follow.
- Commenting code is very important! Use # at beginning of line to denote a comment.
- code = program statements and commands that lead to a result

4. Integrated Development Environment (IDE)

Most programming languages have an IDE which is used to develop programs. There are a lot of different IDEs for Python, but the IDLE IDE is a part of the Python package. It's a simple IDE, and we will use it in part for this reason and also because I want everyone to use the same one. The >>> prompt in the IDLE Shell window indicates that the Python interpreter is ready to interpret a statement.

5. Simple Input and Output

Input
We use the `input()` function to receive input from *stdin* (standard input) which is your monitor.

Output
We use the `print()` function to output text to *stdout* (standard output) which is again your monitor. We'll learn multiple ways to format arguments for the `print()` and `input()` functions, i.e., different ways of doing string formatting. Stay tuned!

As I mentioned before, Jupyter Notebook allows us to use text, images, Python code, and more in a single file.

In [1]:
```
# Here's our very first Python statement in this class (it's also a
# standard statement used to demonstrate how different programs print
# something).

print('Hello, World!')
```

```
Hello, World!
```

In [2]:
```
# The importance of syntax!

print('Hello, World!)
```

```
  Input In [2]
    print('Hello, World!)
                         ^
SyntaxError: unterminated string literal (detected at line 3)
```

Notice that we used quotes around some text. We call the text a **string**, and a string always requires the use of quotes, even if it's a single character. Let's look at some more input and output statements.

In [3]:
```
# We prompt the user for their name.  Again, we use quotes around the
# string.  The = sign  isn't the same as it is in math.  We call it an
# assignment operator, and it assigns the results of running the input()
# command to an lvalue.  Here we call the lvalue 'name'.

name = input('Enter your name: ')  # The = is an assignment operator; it assign
                                   # the value entered by the user to 'name'
```

```
Enter your name: SpongeBob
```

In [4]:
```
# We can now use the name we entered because we assigned it to the lvalue
# 'name'.  'sep' is short for separator, and here the separator is no
# space, i.e., there's no space between the first string 'My name is ',
# the argument 'name', or the second string '.'
# Note the use of the commas; they're all necessary!

print('My name is ', name, '.', sep='')
```

```
My name is SpongeBob.
```

6. A Quick Tour of Canvas, zyBooks, and Our Class Website

zPAs: 1.1.2, 1.1.3, 1.2.1

Be sure to register your iClicker remote or purchase an iClicker Student app for your smart phone by next Wednesday, 1.18.23, when we'll start doing iClicker polls! This is 10% of your grade!