Programming Merit Badge

What will be seen today?

1. Safety
2. History of programming
3. Programming today
4. Intellectual property
5. Careers

And we'll program a little bit!

What will *NOT* be seen today?

A lot!

Because programming exists since at least 50-100 years (since computers existed – approximately)

But if you have any question (on topics seen here or on anything else related to programming), let me know and I'll try my best to answer!

(also, Stack Overflow probably has the answer to most questions! https://stackoverflow.com/questions)

And so? What is programming?

- Programming is the act of inserting instructions into a computer or machine to be followed.

- There are many different career fields involving the programming of computers; each utilizing different languages, techniques, and systems.

- We are only going to cover a few of the different aspects of programming during this Merit Badge, but there are so many more.

Programming is converting ideas into instructions

Understand the purpose of the program (***requirements***)

Break instructions into steps (***design***)

Translate design into language (***coding***)

Verify actual results match expected results (***testing***)

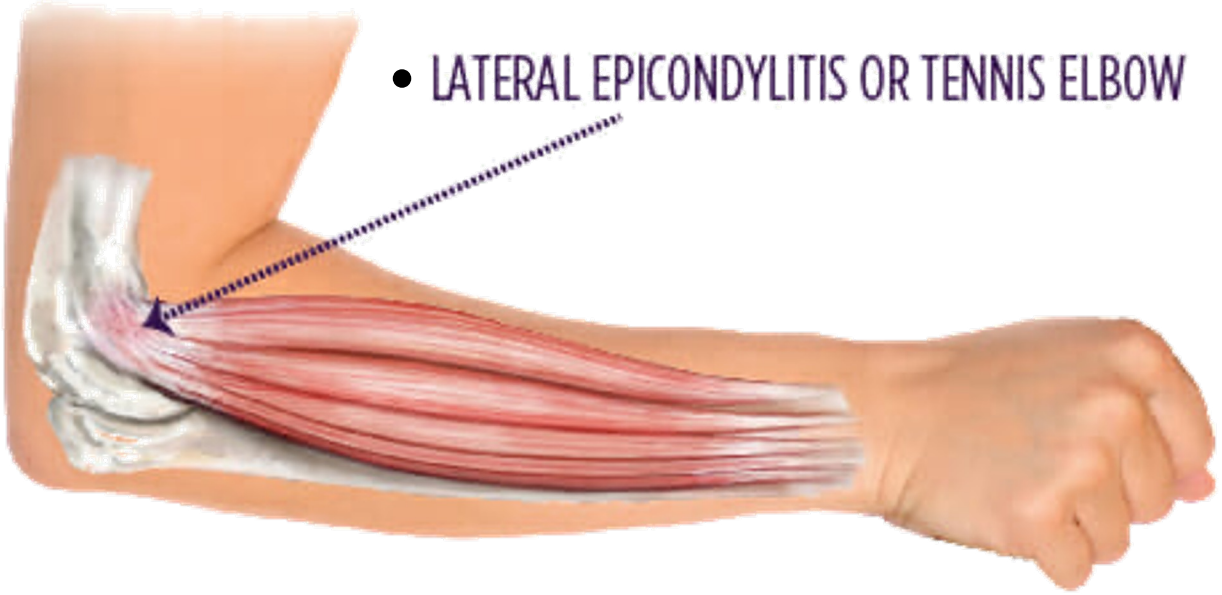
1. Safety

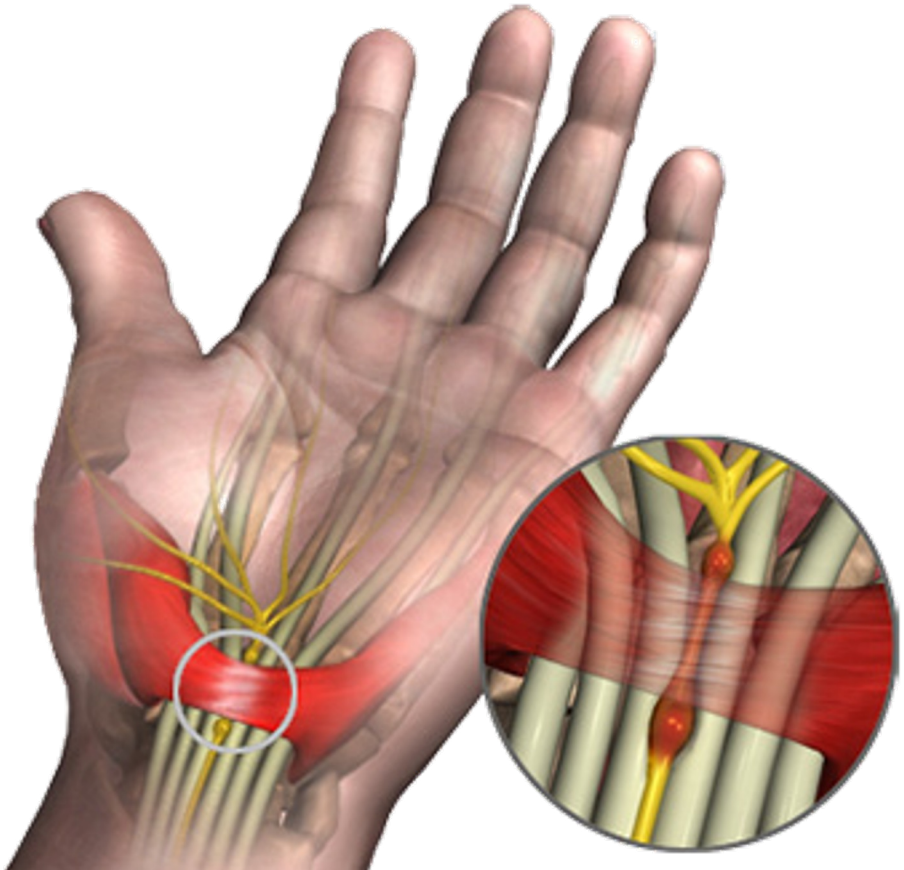
"It's the latest innovation in office safety.
When your computer crashes, an air bag is activated
so you won't bang your head in frustration."



RSIs - Repetitive Stress Injuries

- Carpal tunnel syndrome
- Cervical radiculopathy
- Reflex sympathetic dystrophy
- Tendonitis
- LATERAL EPICONDYLITIS OR TENNIS ELBOW





First Aid for RSIs

- Apply an ice pack to the injured area to help reduce pain and swelling
- Use an elastic joint support or wrap the area firmly with an elastic bandage to limit the swelling and to protect the injury. Do not wrap it so tightly that blood circulation is restricted!
- Rest the injured area
- Take an anti-inflammatory pain reliever as recommended by your physician
- After 24 hours, heat (hot packs, heating pad, whirlpool) may be applied
- As symptoms diminish, gently exercise the affected muscles or joints to help relieve remaining tenderness, stiffness, and tingling or numbness
- If pain is severe or persistent, seek medical attention

Injury prevention

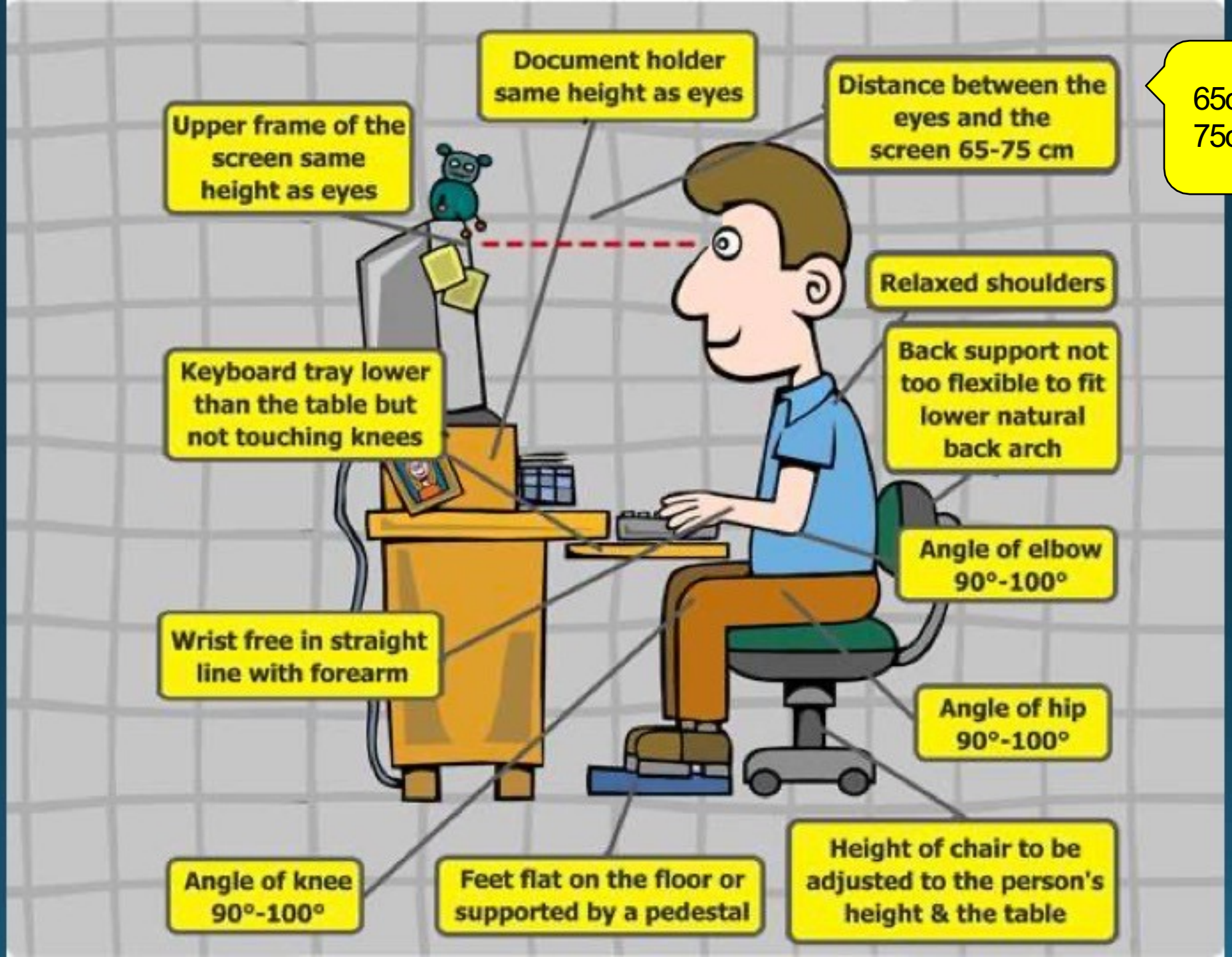
- Proper equipment (+ ergonomics)
- ***HYDRATION***
  - Soda, juices and other sweet drinks are ***not*** a substitute for water
- Program in a well-lit room (prevent eyestrain)
- Minimize the contrast between your monitor and the rest of the room
- Make sure there is no glare on the screen
- Take breaks to give your body time to recover

Take breaks

- **Eye breaks**
  - Look away from the monitor from time to time, preferably at something more than 20 feet away. This gives your eyes a chance to relax and helps prevent eyestrain.
- **Typing breaks**
  - Rest your hands in a relaxed, flat, straight manner to give them time to recover and prevent RSIs
- **Rest breaks**
  - Take a break every 30 minutes or so to give your body a chance to relax. You can use software programs that remind you to take a break so you don't get stuck in a trance staring at your monitor
- **Exercise breaks**
  - Get up and stretch, rotate your head and shoulders, move your arms and legs.

- You will find that you can program better and longer if you do this regularly

Ergonomics


Document holder same height as eyes
Distance between the eyes and the screen 65-75 cm
65cm = 25.6 in
75cm = 29.5 in
Upper frame of the screen same height as eyes
Relaxed shoulders
Keyboard tray lower than the table but not touching knees
Back support not too flexible to fit lower natural back arch
Angle of elbow 90°-100°
Wrist free in straight line with forearm
Angle of hip 90°-100°
Angle of knee 90°-100°
Feet flat on the floor or supported by a pedestal
Height of chair to be adjusted to the person's height & the table

Electrical safety

- Keep liquid and food away from plugged-in machines
- Be sure the equipment is properly grounded to prevent shock hazards
- It's best to unplug the computer when it's not in use, especially during a thunderstorm
- Make sure any cords are neatly stowed to prevent tripping

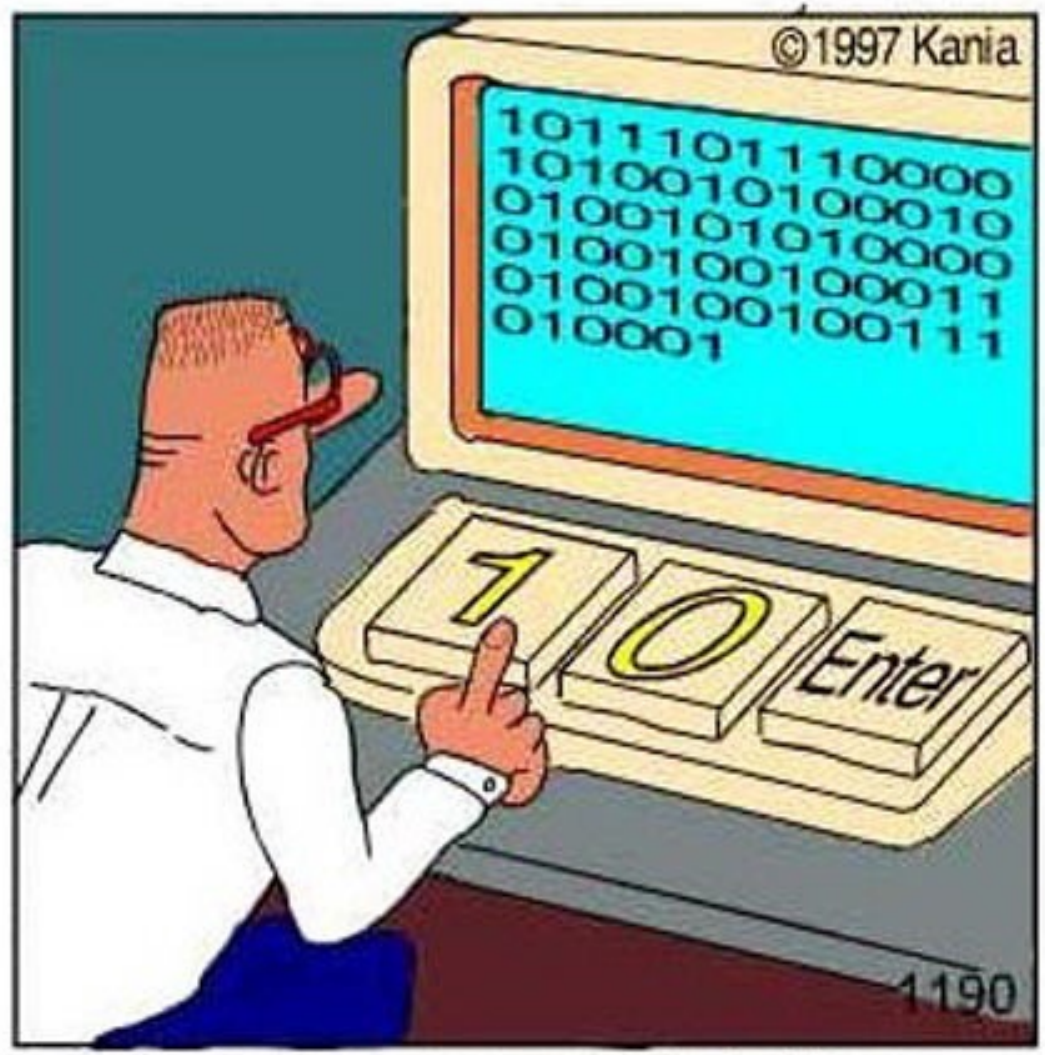
Additional pointers

- Computer-Related Repetitive Stress Injuries (Johns Hopkins): https://www.hopkinsallchildrens.org/Patients-Families/Health-Library/HealthDocNew/Computer-Related-Repetitive-Stress-Injuries
- 12 tips for an Ergonomic Computer Workstation (Cornell University): http://ergo.human.cornell.edu/DEA6510/dea6512k/ergo12tips.html
- 25 Ergonomic Tips For Students When Working At A Computer (Vista College): https://www.vistacollege.edu/blog/resources/25-ergonomic-tips-when-working-at-a-computer/

1. Safety

Now you can answer Requirements 1a And 1b!

2. History of programming

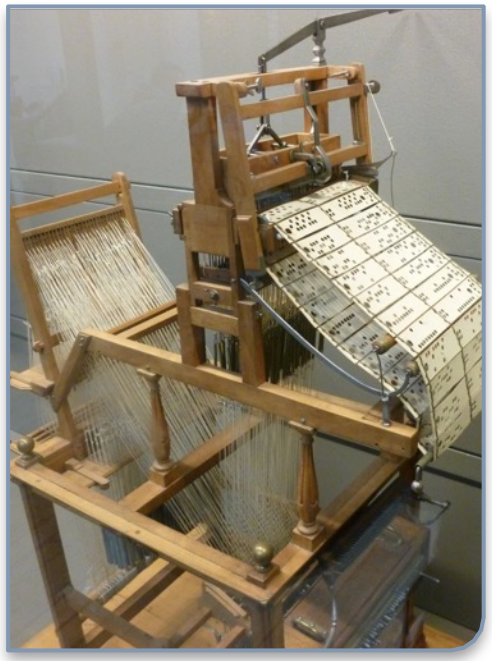


Real programmers code in binary.

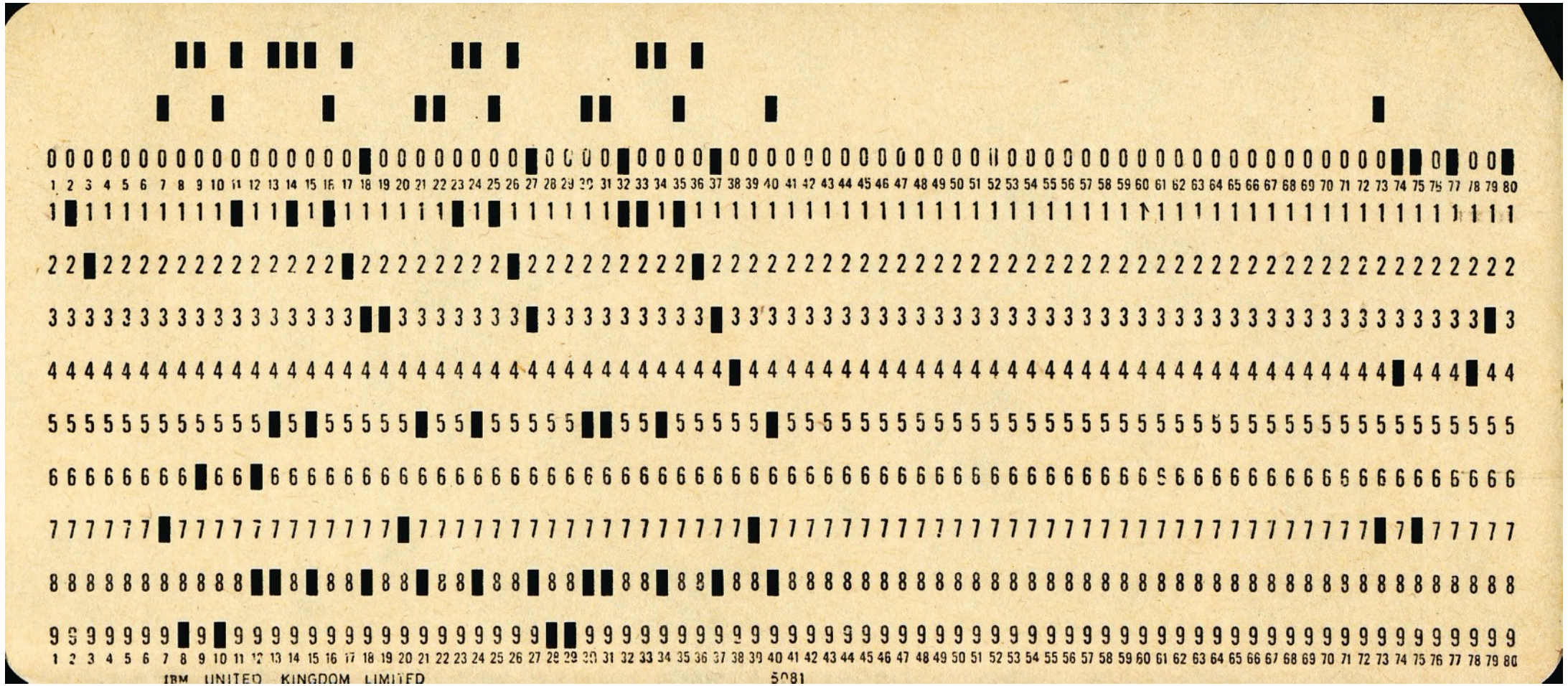
Before computers

Before the modern electrical computer, mechanical devices used in factories were the first machines to be programmed.

An example is the Joseph Jacquard Loom (1804) which used hole-punched cards to “program” patterns into fabric.

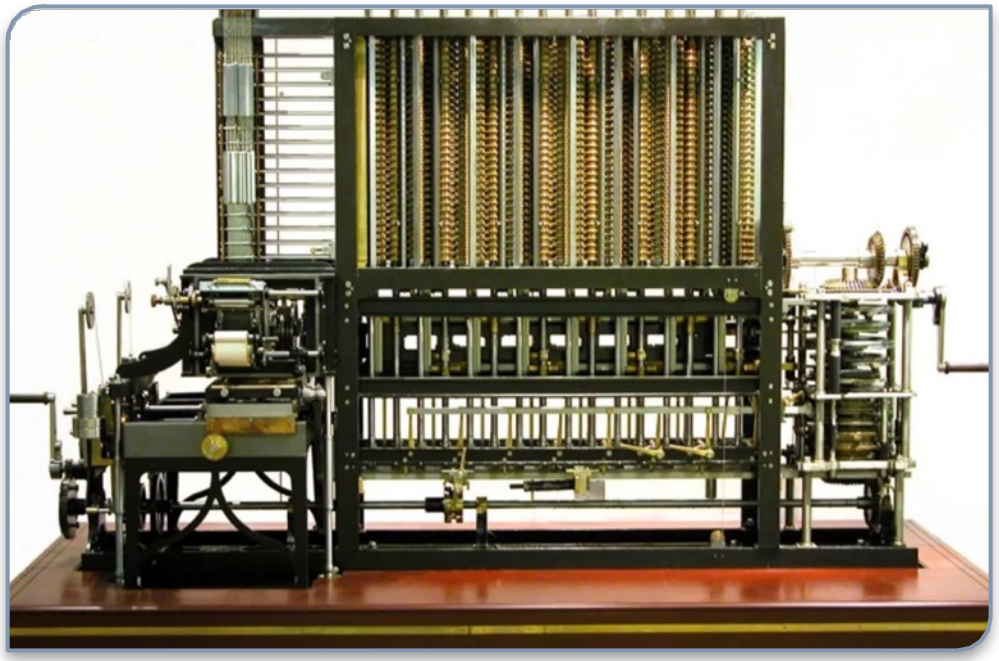


The picture on the left is the loom.
The picture on the right is a portrait of Jacquard was woven in silk on a Jacquard loom and required 24,000 punched cards to create (1839). One of these portraits in the possession of Charles Babbage inspired him in using perforated cards in his Difference Engine.



A punch card

Before computers



Ada Lovelace, the first programmer, theorized how to program Babbage's Machines.

Charles Babbage in 1823 started work on his Difference Engine. It was programmed using punch cards and could do simple calculations to 31 digits. Do to high costs, it was not built until 1991, well after his death. It weighed 15 tons and was 8 ft tall.

It used human-power to turn the gears and cranks and output the result using wheels with digits painted on.

Fun Fact: The gear technology didn't exist to build his machine, so Babbage invented new ways of cutting gears. This incidentally advanced machinery and factories during the end industrial revolution (1760-1840).

Ada Lovelace
(1815 – 1852)

Before computers

In 1985, Herman Hollerith designed the “Electric Tabulating System”, a machine designed to take on the 1890’s Census. It was an early Scantron machine using punch cards.

The 1880’s Census took 7 years to count, so do to the growing population, the 1890’s and 1900’s Censuses would have taken more than 10 years. This would not be good.

With his machine, the 1890’s Census only took 6 weeks rather than 10 years. This proved computers were a viable solution to many previously impossible problems.

John von Neumann

Conditional Control Transfer

If <condition> then <action a> else <action b>

if <raining> then <stay inside> else <go outside>

While <condition> do <action>

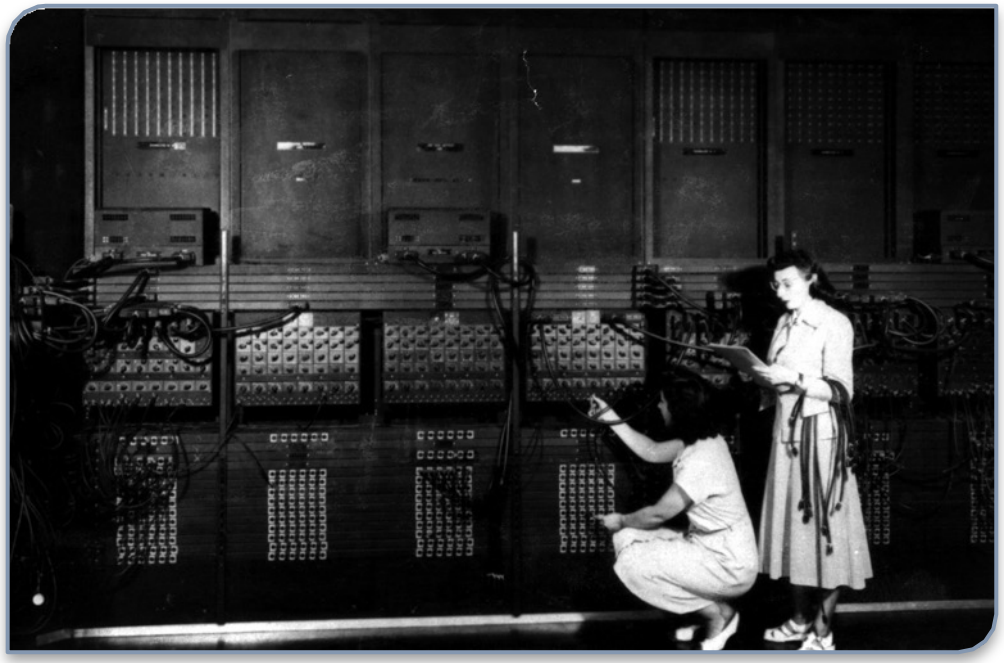
while <raining> do <hold umbrella>

subroutines, libraries, reusable code

go outside:
- stand up
- walk to the door
- open the door
- walk out the door
- close the door

Programming Pioneer
1903-1957

Early computers



ENIAC 1946 – What do you notice about this photo

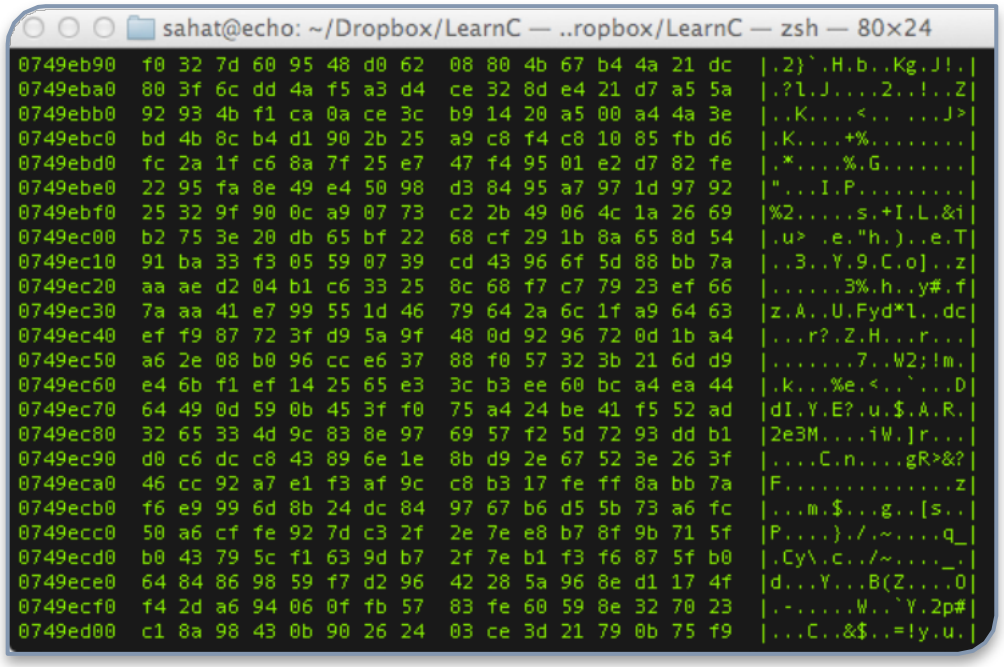
ENIAC – Electronic Numerical Integrator And Computer (1946)

- First general purpose computer
  - Used Base-10 instead of Binary (Base-2)
- Used to calculate firing-tables for the military.

UNIVAC – UNIVersal Automatic Computer (1951)

- First commercial computer
- Brought computers into the public eye after it correctly predicted the “total-upset, landslide”, 1952 Presidential Election.

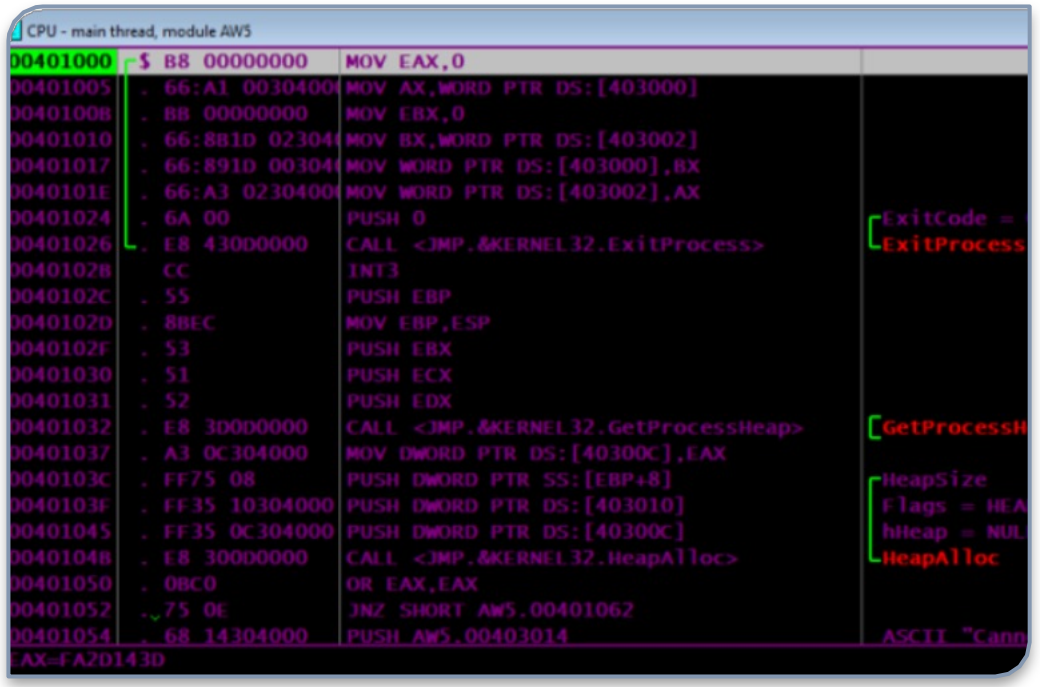
History of programming



- What was the first programming language?
  - Binary / Machine Language (ML)
- Binary / ML is really hard to read, but it can be done.
- Early computers used switches and cables to accomplish this.
- It is insanely fast, only limited by hardware speed.
- All programming languages end up as Binary / ML at some point during execution.

History of programming

- Next came Assembly Language (ASM)
- Slightly easier to read than Binary / ML
- Still very fast because it maps back to Binary / ML
- Very few people ‘need’ to program is ASM
- There is a different Assembly Language for each CPU design, so it is not portable code.
    - Why is portable code good?



Assembly Language

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70   START   LDS     #STACK

                        ****************************************
                        * FUNCTION: INITA - Initialize ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ****************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF   HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP
```

Assembly language is hardware specific and is compiled into machine language (binary code)

```
01010100 01101000 01101001 01110011
00100000 01101001 01110011 00100000
01110100 01101000 01100101 00100000
01110100 01110101 01110100 01101111
01110010 01101001 01100001 01101100
00100000 01110100 01101111 00100000
01101100 01100101 01100001 01110010
01101110 00100000 01100010 01101001
01101110 01100001 01110010 01111001
00101110 00100000 01001001 00100000
01101000 01101111 01110000 01100101
00100000 01111001 01101111 01110101
00100000 01100101 01101110 01101010
01101111 01111001 00100000 01101001
01110100 00100001
```

History of programming

Next-Generation Languages came around the 1950's.

They allowed:

- Code portability between different systems
- Easier to write, read and debug code
- Allowed for new concepts (i.e. functions, classes, objects, OOP)
- Explored new fields (i.e. science, math, computer science, data science, business)

The first big languages were… (in order of creation)

FORTRAN, LISP, COBOL, BASIC and Pascal

Early Important Languages

1950s

- **FORTRAN (1957)**
  - John Backus (IBM)
  - Formula Translating System
  - High performance computing - weather and climate modeling
  - Computationally intensive
- **LISP (1958)**
  - John McCarthy (MIT)
  - Recursion and list processing
  - Artificial intelligence
- **COBOL (1959)**
  - Common Business Oriented Language
  - Primarily used in business, finance, and admin systems

1960s-1970s

- **BASIC (1964)**
  - John Kemeny and Thomas Kurtz (Dartmouth)
  - Beginner's All purpose Symbolic Instruction Code
  - Based on FORTRAN II
- **Pascal (1970)**
  - Niklaus Wirth
  - Developed to encourage good programming practices
  - Structured programming
  - Data Structuring
- **C (1972)**
  - Dennis Ritchie (AT&T Bell Labs)
  - Developed to implement Unix OS
  - One of the most widely used programming languages

Additional pointers

- History of programming languages: https://en.wikipedia.org/wiki/History_of_programming_languages
- Visual history of programming languages: https://visual.ly/community/infographic/technology/history-computer-programming

- Ada Lovelace: https://en.wikipedia.org/wiki/Ada_Lovelace

2. History of programming

Now you can answer Requirements 2a!

3. Programming today


THE
C
PROGRAMMING
LANGUAGE
C++
C#
Java
python
JavaScript
php
wikiHow to Start Learning Computer Programming

Programming Now

How many languages do you recognize?

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL
SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust
F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM
RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

Programming Now

Why are the languages grouped into colors?

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
BASH
TypeScript
PostScript
CoffeeScript
Perl
x86-Assembly MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

The Green Languages are General Programing Languages
The Purple Languages are Scripting Languages
The Red Languages are Markup Languages
The Blue Languages are Declarative Languages
The Orange Languages are Assembly Languages

Different types of languages have different purposes.
It is important to match the type of work to the correct language to insure the best results.

Programming Languages

Here are a few languages and the problems they try to tackle…

C++ – General Purpose, High Performance | ex. Game Engines, Desktop Apps (Adobe Photoshop, Chrome)
C – General Purpose, High Performance, Light Weight | ex. Linux OS, macOS, Integrated Circuits, Drivers
Java – General Purpose, Multiplatform | ex. Minecraft, Server Apps, Android Apps
C# – General Purpose, Windows Platform | ex. Unity Games, Server Apps, StackOverflow
Swift – General Purpose, iOS & macOS | ex. most apps for iPhones and macOS (replaced Objective-C)
SQL – Database Communication
JavaScript – General Web Scripting | ex. Interactive webpages, webpages that can run dynamic code
HTML – Webpage Design, Layout and Markup
CSS – Webpage Styling, Coloring, Fonts and Positioning
PHP – Web Server Code | ex. Backend Web Dev., Web Content Management Systems (i.e. WordPress)
TypeScript – Stricter Superset of JS that transpiles into JS | ex. Large JavaScript Apps
XML – Human and Machine readable file format for data sharing between apps

Programming Examples

Hello World

C++

```cpp
#include <iostream>
int main(int argc, char *argv[])
{
	char myString[] = “Hello World!”;
	std::cout << myString << std::endl;
	return 0;
}
```

Java

```java
class HelloWorld {
    private String myString = "Hello World!";
    public static void main(String args[]) {
        System.out.println(myString);
    }
}
```

Notice how different languages can look very different even when they are doing the same task.
Notice also how the bracing (i.e. “{}”) style is different between languages.

C - The foundation for many other language

"Combines the power of assembly language with the readability and maintainability of assembly language."

C is used for:

- Computer applications
- Embedded Softwares
- Creating compilers
- Unix Kernel

C++ - High performance programming language

"Enough rope to shoot yourself in the foot.""

C++ is used for:

- Software for large scale ecommerce
- videogames
- Adobe systems
- CAD (Autodesk)
- Most microsoft applications
- Browsers (Firefox))

Minecraft was programmed using Java

Programming Examples

Hello World

C#

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class HelloWorld
    {
        String myString = "Hello, world!";
        static void Main(string[] args)
        {
            Console.WriteLine(myString);
        }
    }
}
```

X86 Assembly

```
.486
    .model flat, stdcall
    .stack 100h
    option casemap :none

    ExitProcess PROTO Near32 stdcall, dwExitCode:dword
    putch PROTO Near32 stdcall, bChar:byte;
.data
    strMyString byte "Hello World",0
.code
main PROC
    mov ecx, LENGTHOF strMyString
    mov esi, OFFSET strMyString
L1:
    invoke putch, byte PTR esi
    inc esi
    loop L1
    invoke ExitProcess,0
main ENDP
END main
```

Programming Examples

Hello World

JavaScript

```
myString = "Hello World!";
console.log(myString);
```

Python

```
myString = 'Hello World!'
print(myString)
```

Notice how different languages can look very different even when they are doing the same task.

Python - simple zen like programming language

"Compile, run and ship your pseudo-code."

Python is used for:

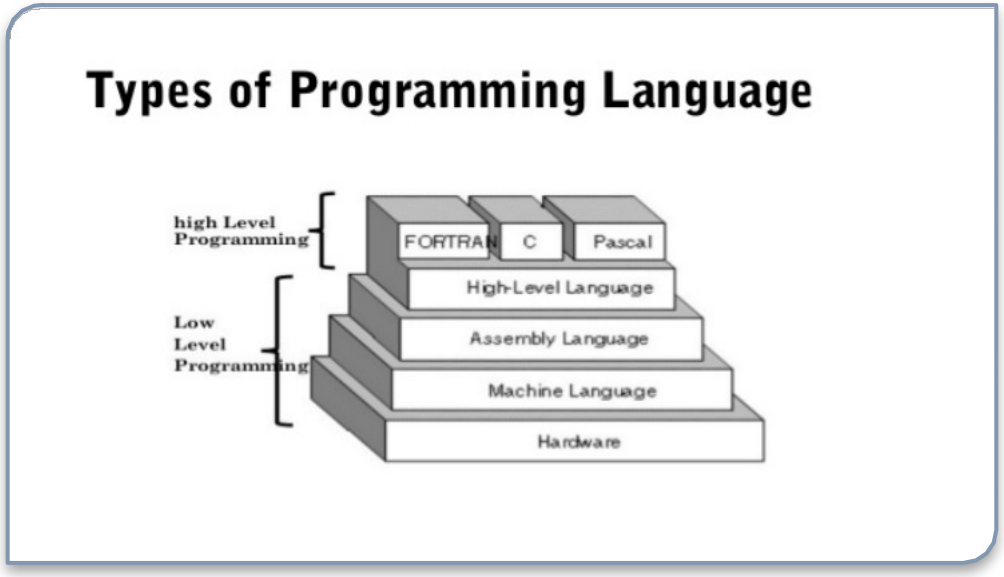
- scientific and numeric computing
- Web and Internet Development
- Teaching programming
- Software Development
- Desktop GUIs

Programming Language Types

Languages can be split into a three different levels..

- High-Level (ex. Python, Ruby, JavaScript, Java, SQL)
- C-Level (ex. C, C++, Rust)
- Low-Level (x86 Assembly, Machine Language)

Programming Language Types



Notice: Java, Python, etc. are one level higher than FORTAN, C and PASCAL

Why would you use a High-Level, Low-Level or C-Level language?

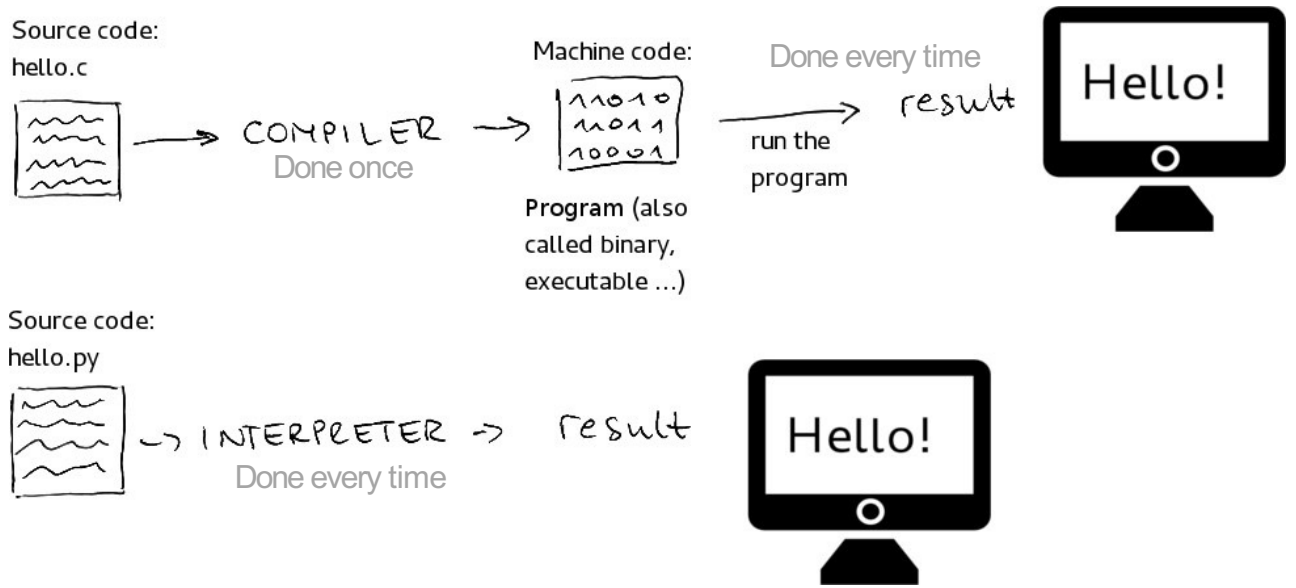
- Low Level
  - Pros: Fast Execution, No Overhead, Single Platform, Compiled
  - Cons: Hard to read, write, debug, and maintain
  - Examples: ML, MASM, TASM, NASM, MIPS
- High-Level
  - Pros: Easier to read, write, debug, and maintain, Multi-Platform, Compiled or Interpreted
  - Cons: Slower than Low-Level, not as much control over hardware
  - Examples: Python, Ladder-Logic, JavaScript, Java, SQL
- C-Level
  - Best of both worlds, Compiled
  - Good control over hardware with ease of writing.
  - Examples: C, C++, Rust, FORTRAN, PASCAL

Programming Language Types

This photo illiterates the difference between compiled and interpreted languages…

Use your computers to make a list of 3 compiled languages and 3 interpreted languages.

Where would you use a compiled languages vs an interpreted language?



Grace Hopper creates first compiler

The first compiler A-0 turned statements into ones and zeros which the computer could understand.

Programmed Devices

Our lives are filled with so many programmed devices, you many not even notice…

What are somethings around your house that are programmed?

- Smart TVs, Smart Door Bells
- Xbox, PlayStation, Wii, Ms. Pacman
- Microwave, Wi-Fi Router (these two are the same thing)
- Etc..

What language do you think these were programmed in?

Additional pointers

- 100 Of The Most Popular Programming Languages Explained In Minutes: https://www.whoishostingthis.com/resources/programming/
- Code academy: www.codeacademy.com
- Code.org: www.code.org
- Learn Python: www.learnpython.org
- Learn C++: www.learncpp.com
- Scratch: scratch.mit.edu
- Killer PHP: www.killerphp.com
- W3 Schools: www.w3schools.com

Programming Merit Badge

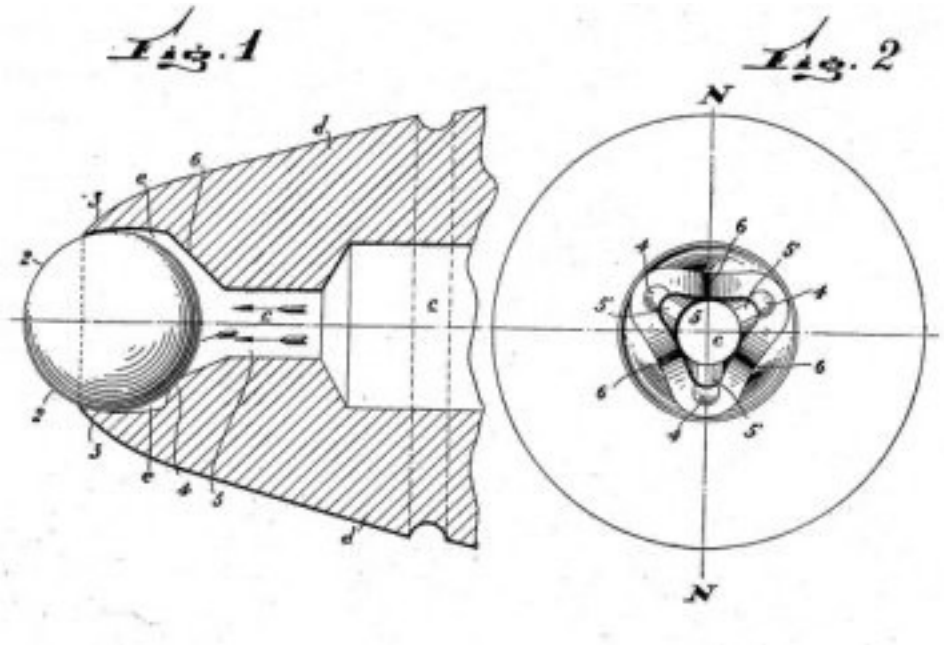
Welcome! And ...

3. Programming today

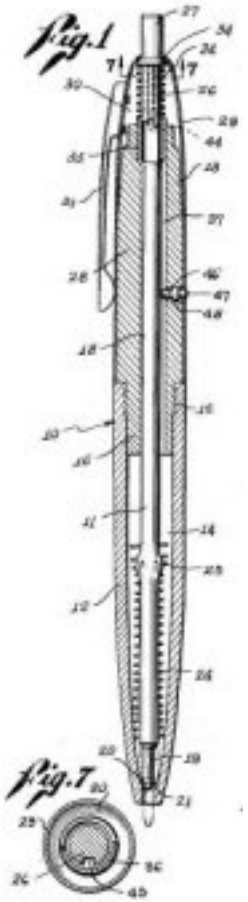
Now you can answer Requirements 3a and 3b!

4. Intellectual property

László Jozsef Bíró,
the inventor of the ballpoint pen

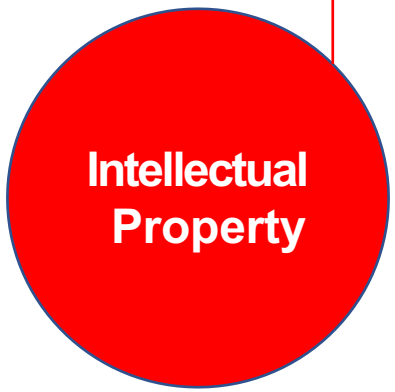


U.S. Patent No. 2390636:
“Writing Instrument.”



U.S. Patent No. 2734484:
“Ball Point Pen.”

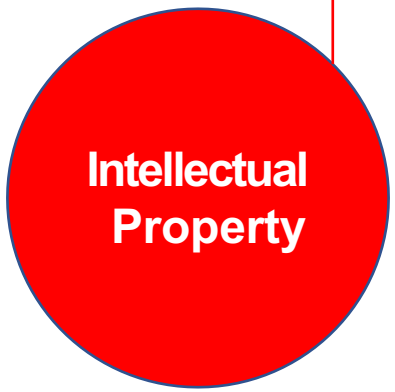

Intellectual
Property



Copyright protection (©)

Protects a particular expression of an idea that the author created

Automatic + registration with the U.S. Copyright Office (~ $85)

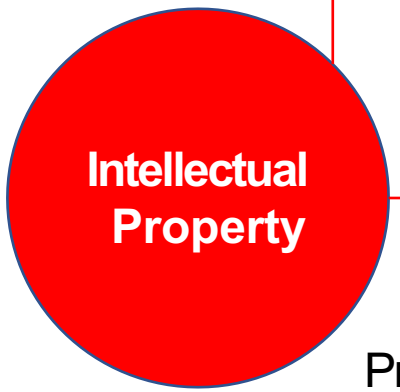


Copyright protection (©)

Protects expression of idea

The Merit Badge Book is © BSA

Powerpoints, game art, specific code, …



Copyright protection (©)

Protects a particular expression of an idea that the author created
Automatic + registration with the U.S. Copyright Office (~ $85)

Patents

Protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter”.
Must be applied for (USPTO)
Need to describe inner workings

Intellectual Property

Copyright protection (©)

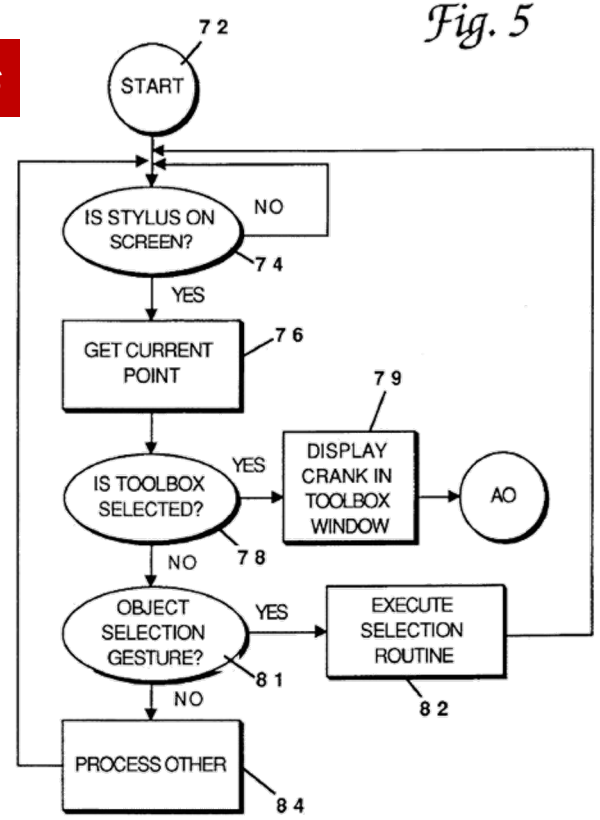
Protects a particular expression of an idea that the author created
Automatic + registration with the U.S. Copyright Office (~ $85)

Patents

Protects useful and innovative processes or methods, machines, manufactured items, …

A new and revolutionary math algorithm used in an app

Fig. 5



Intellectual Property

Trademark (™)

Protects a word, phrase, symbol, sound or color that identifies or distinguishes the source of a particular product or service

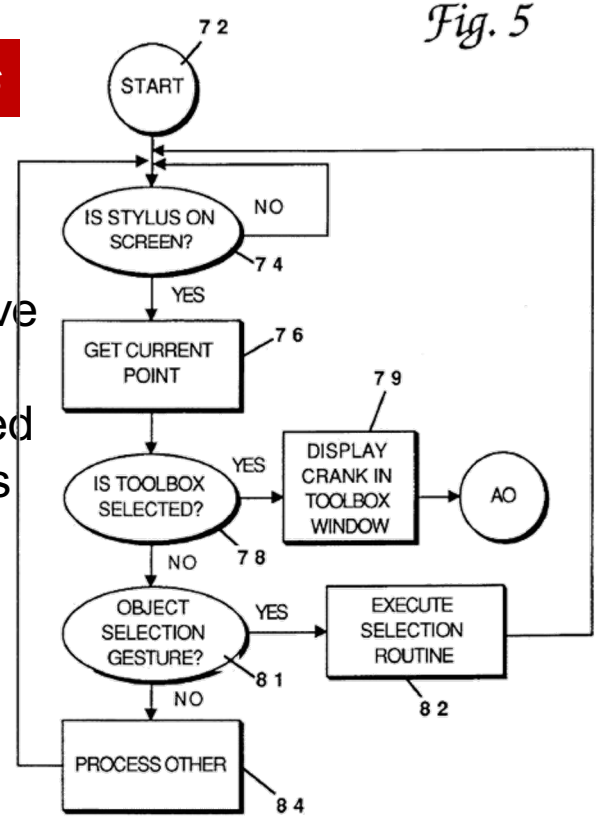
Copyright protection (©)

Protects a particular expression of an idea that the author created
Automatic + registration with the U.S. Copyright Office (~ $85)

Patents

Protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter”.
Must be applied for (USPTO)
Need to describe inner workings

Fig. 5



Intellectual Property

Trademark (™)

Protects a word, phrase, symbol, sound or color that identifies or distinguishes the source of a particular product or service

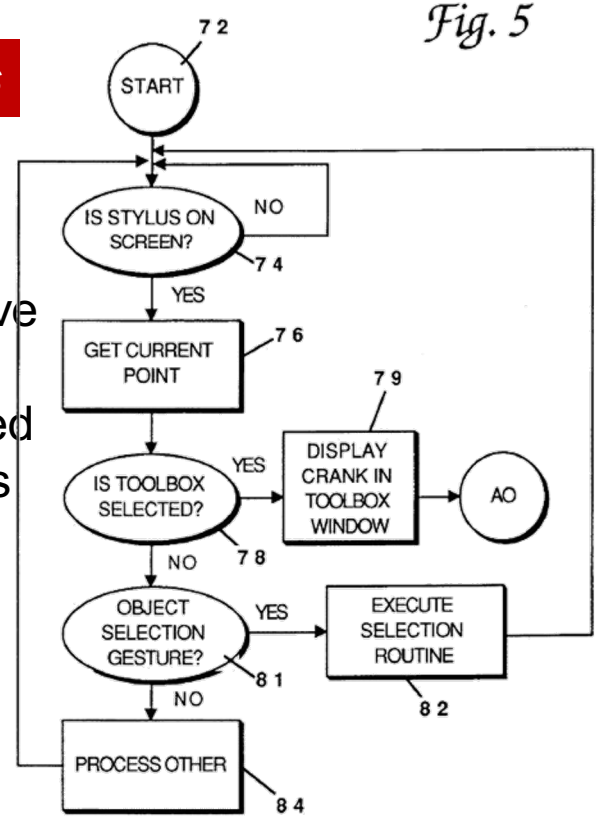
Copyright protection (©)

Protects a particular expression of an idea that the author created
Automatic + registration with the U.S. Copyright Office (~ $85)

Patents

Protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter”.
Must be applied for (USPTO)
Need to describe inner workings

Fig. 5



Intellectual Property

Trademark (™)

Protects a word, phrase, symbol, soud or color that identifies or distinguishes the source of a particular product or service

Trade secrets

Protects valuable information be not disclosing it to anyone, enforced by a contract called a NDA (Non Disclosure Agreement)

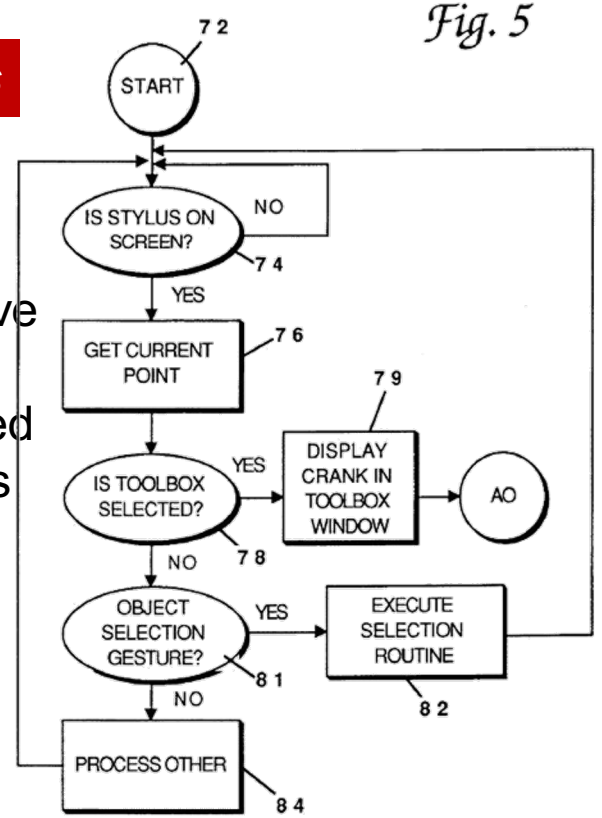
Copyright protection (©)

Protects a particular expression of an idea that the author created
Automatic + registration with the U.S. Copyright Office (~ $85)

Patents

Protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter”.
Must be applied for (USPTO)
Need to describe inner workings

Fig. 5



Intellectual Property

Trademark (™)

Protects a word, phrase, symbol, soud or color that identifies or distinguishes the source of a particular product or service

Trade secrets

Protects valuable information be not disclosing it to anyone, enforced by a contract called a NDA

What info Facebook collects, PWD cars, …

Copyright protection (©)

Protects a particular expression of an idea that the author created

Automatic + registration with the U.S. Copyright Office (~ $85)

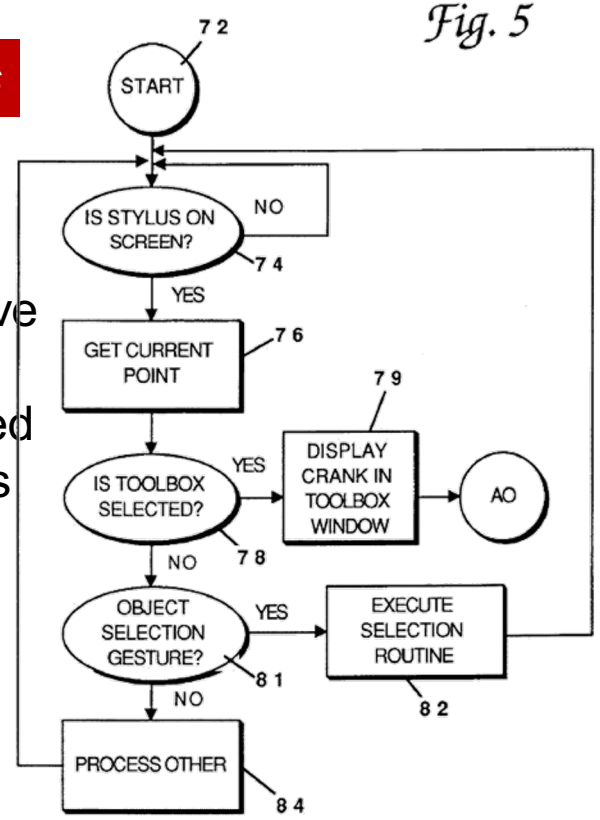
Patents

Protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter”.

Must be applied for (USPTO)

Need to describe inner workings

Fig. 5

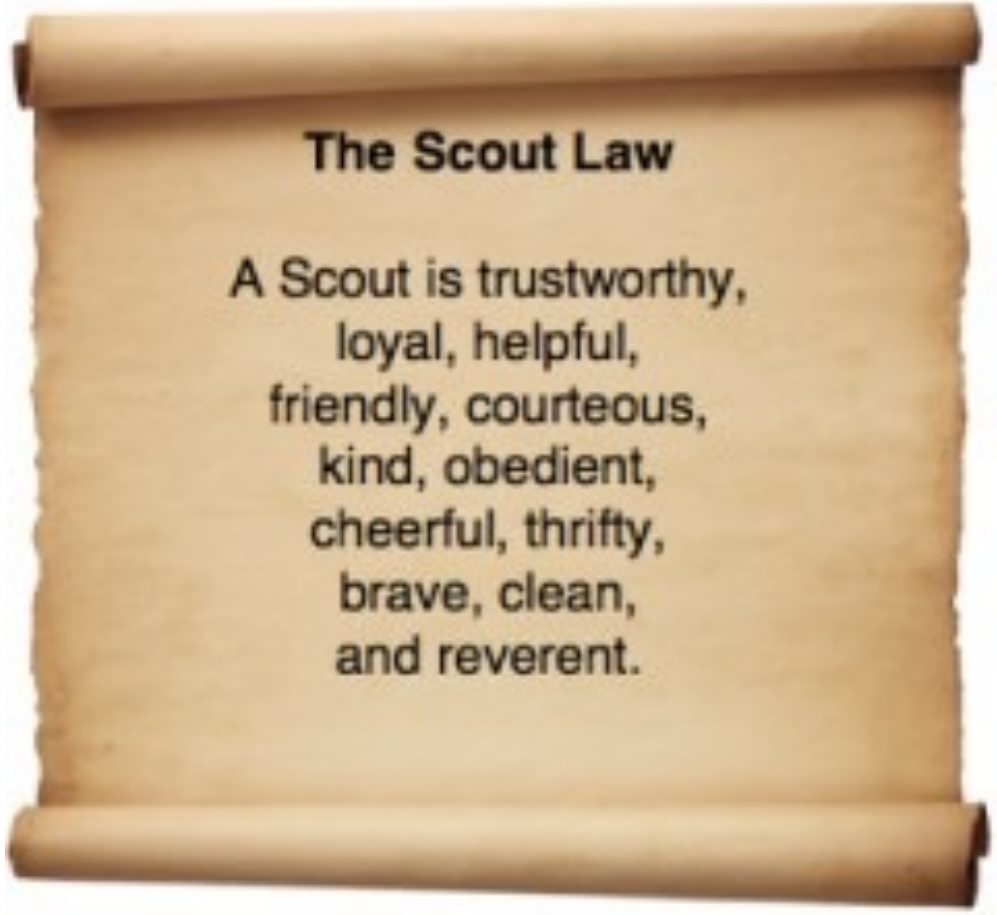


Selling your software

- **Freeware** - 100% free to use, not necessarily free to be modified or distributed
- **Shareware** - free to download and use but asked for donations (i.e. Ad-Block). Not free to modify or distribute
- **Demo** - A free trial version of the program, may not have all the features enabled. Not free to modify or distribute
- **Closed Source** - the code is NOT exposed to the public and cannot be edited or distributed (doesn't mean free)
- **Open Source** - the code is exposed to the public and can be modified or distributed, may be limits or restrictions (doesn't mean free)
- **Public Domain** – There is absolutely no ownership such as copyright, trademark, or patent. Software in the public domain can be modified, distributed, or sold even without any attribution by

  anyone

Software piracy



Owning or licensing?

Do I own a copy of PowerPoint?

Do I own a copy of Google Chrome?

Do I own a copy of an App I built?

What is the difference between owning and licensing?

- Owning means you have every right to do what you want with the software or code. Most people do not own software.
- Licensing is where you “buy or get permission” to use the software, often subscription based.

Additional pointers

- What is Open Source? https://opensource.com/resources/what-open-source
- The Open Source Initiative: https://opensource.org/
- Intellectual Property: https://en.wikipedia.org/wiki/Intellectual_property
- Intellectual Property Theft/Piracy: https://www.fbi.gov/investigate/white-collar-crime/piracy-ip-theft

4. Intellectual property

Now you can answer Requirements 4a, 4b and 4c!

6. Careers

A Day in the Life of a...

Computer Programmer

Median Salary: $84,280

- Know computer languages
- Write computer programs
- Collaborate with other programmers
- Test software programs

the balance

Where is programming used?

(A: Everywhere!)

Where is programming used?

- Mobile devices
- Business applications
- Factory automation
- Robotics
- Internet
- Animation
- Entertainment
- Outdoor

- Engineering
- Science
- Automobiles
- Traffic control
- Information security
- Gadgets
- Medical devices
- Healthcare
- …

Can you find examples around you?

Additional pointers

- Examples of programs for different industries (Boys Life): https://boyslife.org/merit-badges/programming-merit-badge/
- How to become a computer programmer? (Learn How To Become): https://www.learnhowtobecome.org/computer-programmer/
- 9 Programming Careers for Coding Connoisseurs (Rasmussen College): https://www.rasmussen.edu/degrees/technology/blog/programming-careers-for-coding-connoisseurs/
- What Does a Computer Programmer Do? (The Balance Career): https://www.thebalancecareers.com/what-does-a-computer-programmer-do-525996
- 10 signs a career in coding and software development might be right for you (The Guardian): https://www.theguardian.com/careers/ten-career-coding-software-development-right-for-you signs

6. Careers

Now you can answer Requirements 6!

What's next?

- Check the requirements (https://filestore.scouting.org/filestore/Merit_Badge_ReqandRes/Programming.pdf)
- Use the worksheet to track progress (https://boyslifeorg.files.wordpress.com/2019/06/programming.pdf)
- If you need anything, tell me. I'm happy to explain, re-explain, or debug with you!
- When you are ready, let's discuss! The easiest is 20-30 minutes before a Troop meeting