

PROGRAMMING

MERIT BADGE POWERPOINT

© 2018 - 2020 ROBERT BAKER



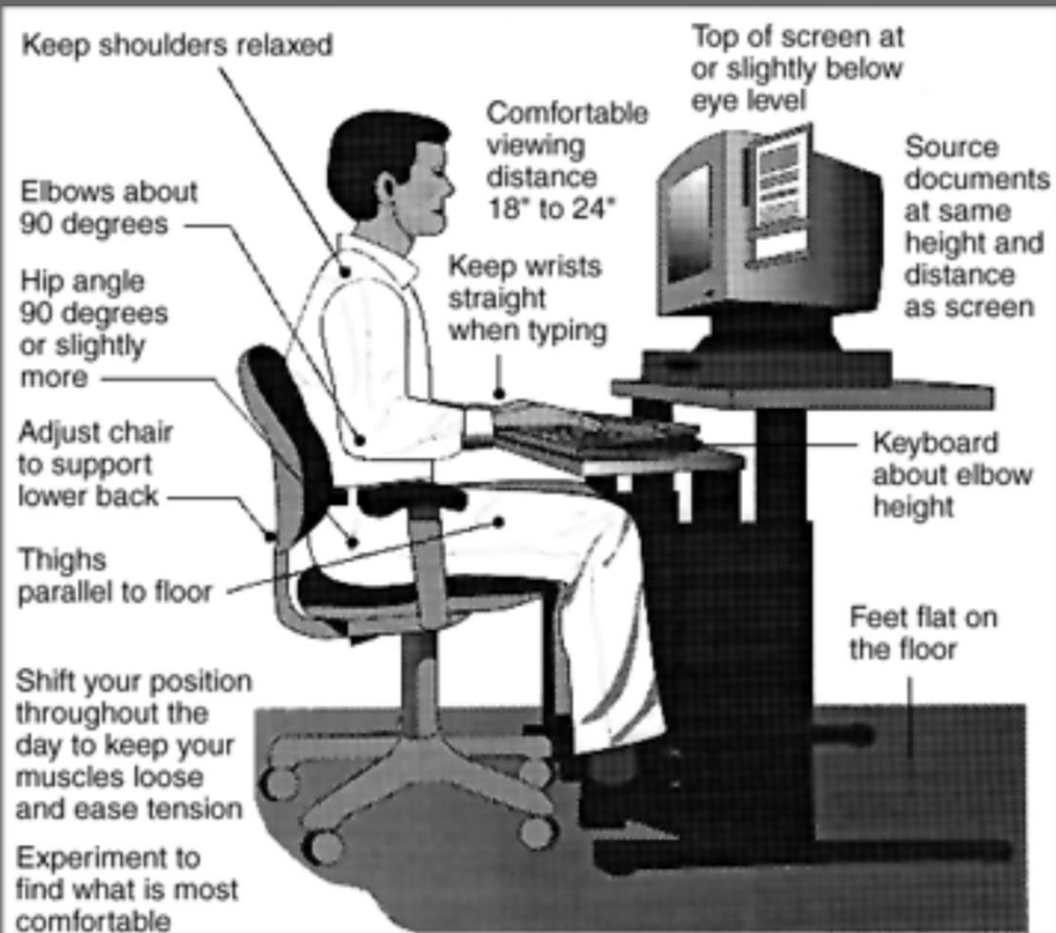
The background features a dark blue gradient with faint, blurred lines of code in various colors (yellow, green, blue) and white circuit-like patterns consisting of lines and small circles.

WHAT IS PROGRAMMING

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.

SAFETY



- Normally programming normally involves computers, which use electricity. It is important to make sure all power-cords are not frayed, and too keep liquids far away to prevent electric shock.
- RSI – Repetitive Stress Injury
 - Caused by typing for long periods of time and can cause pain in the wrists and hands
 - How can RSI be prevented?

SAFETY

- Eye Strain can be caused by using computer screens for extended periods of time.
 - How can eye strain be prevented?



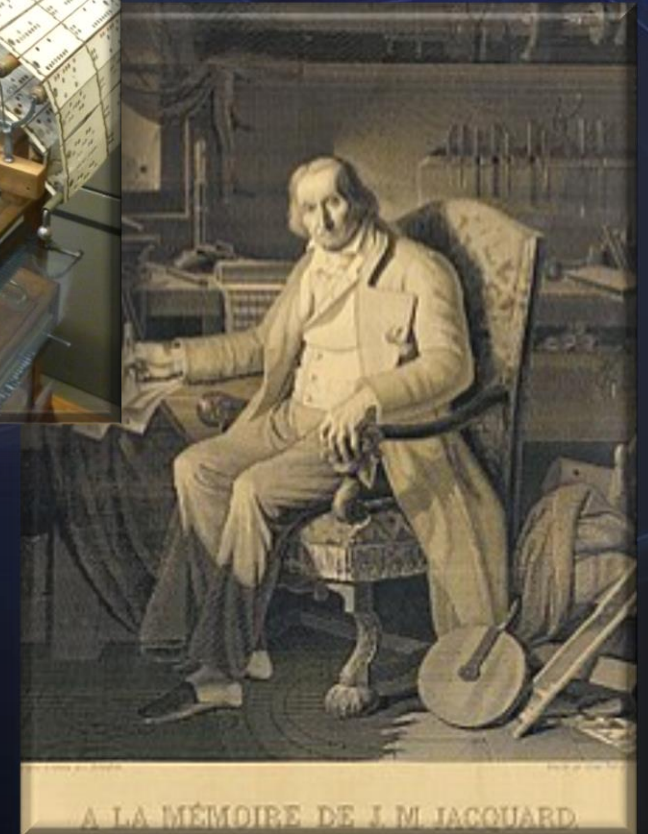
BEFORE COMPUTERS

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

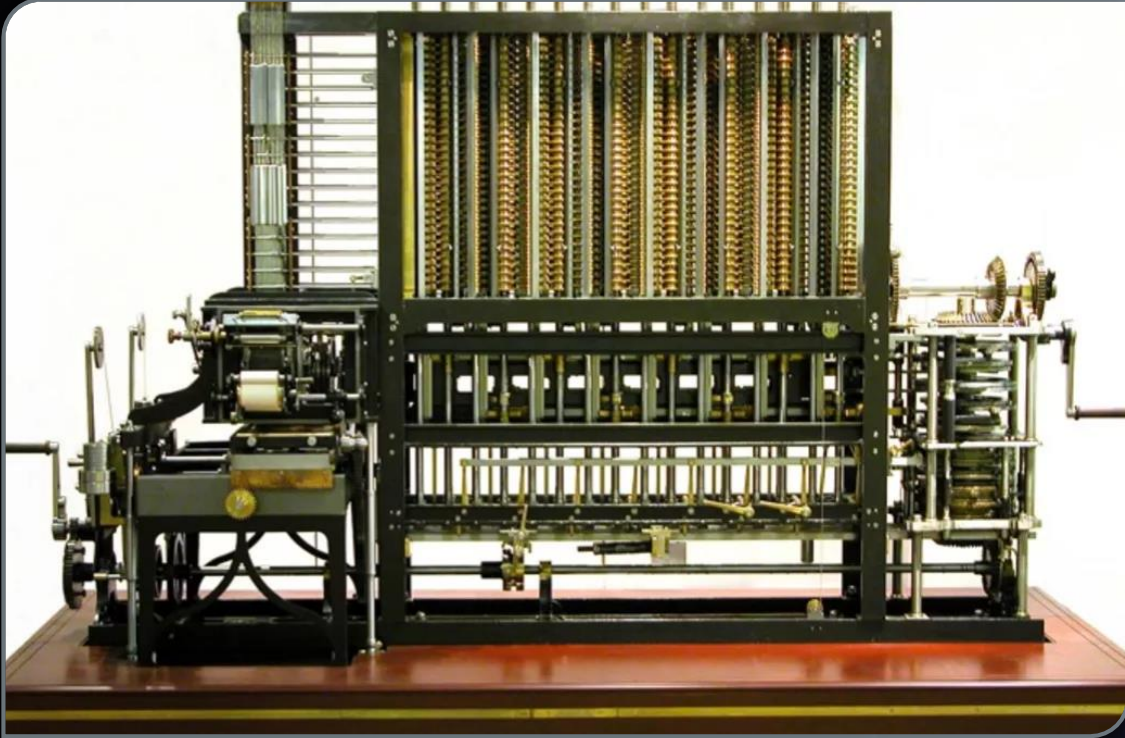
An example is the Joseph Jacquard Loom (1804) which used hole-punched cards to “program” patterns into fabric.

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.



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).

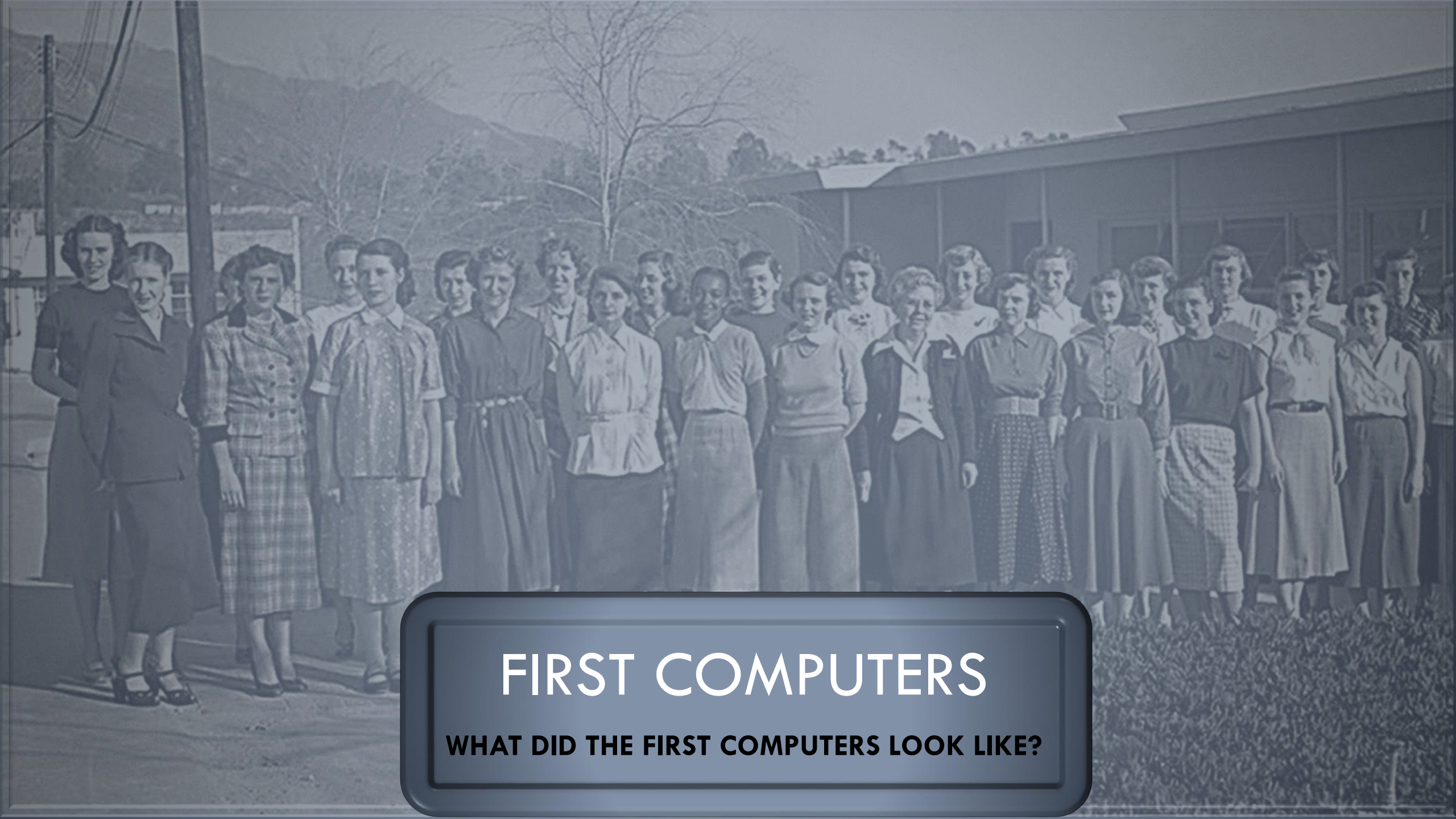
BEFORE COMPUTERS

In 1885, Herman Hollerith designed the “Electric Tabulating System”, a machine designed to take on the 1890’s Census. It was an early Scantron-like machine using punch cards.

The 1880’s Census took 7 years to count, so due 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.

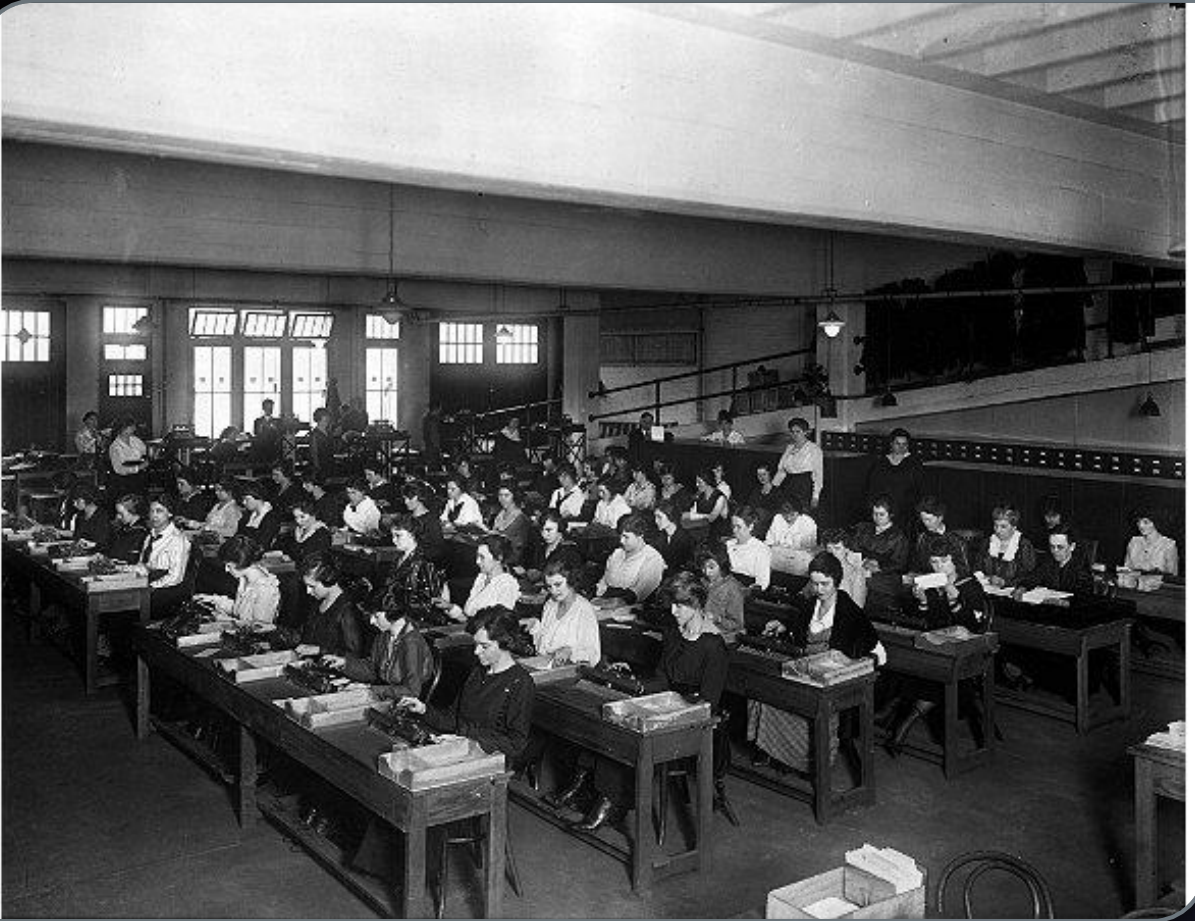




FIRST COMPUTERS

WHAT DID THE FIRST COMPUTERS LOOK LIKE?

FIRST COMPUTERS



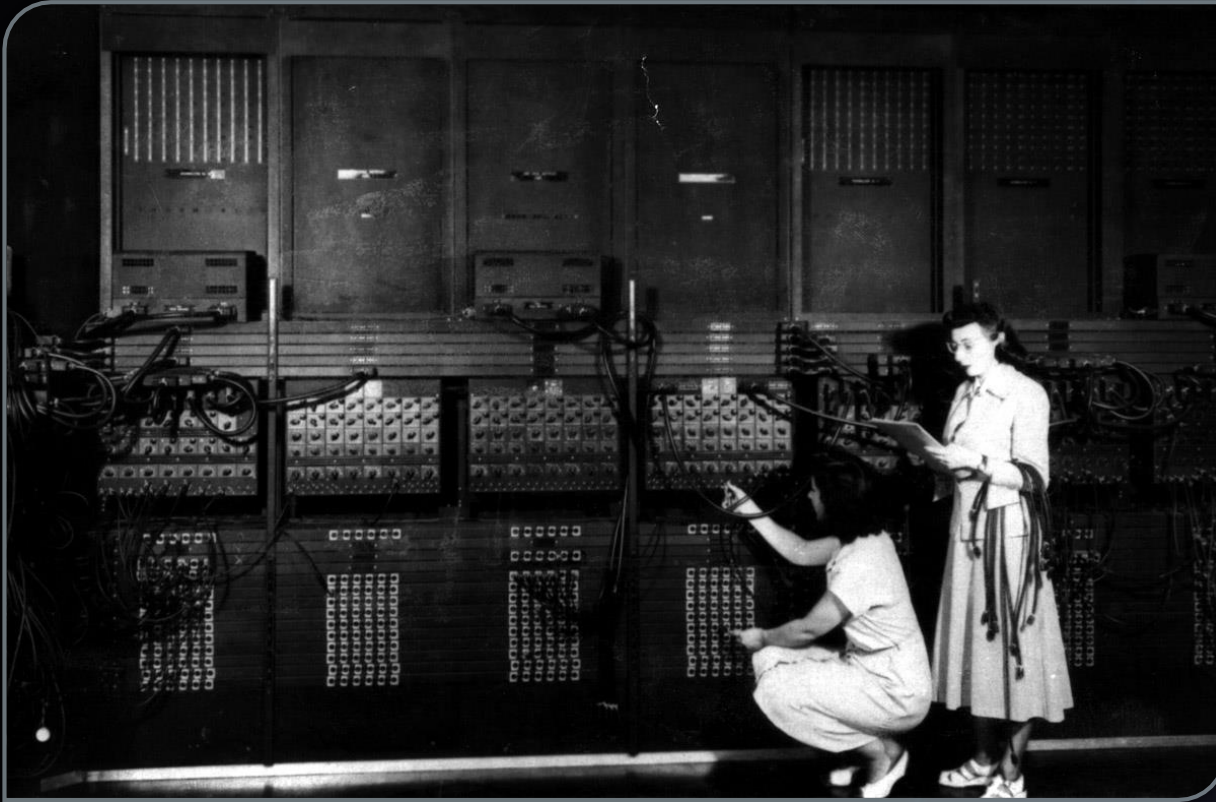
A “Computer” used to be a **job description**, not an electronic machine

Women almost exclusively filled these positions.

Large agencies would have “Computer Rooms” with many ladies doing calculations by hand

A “kilogirl” was a unit of measurement equaling 1000 hours of computing labor

EARLY COMPUTERS



ENIAC 1946 – What do you notice about this photo?

ENIAC – Electronic Numerical Integrator And Computer (1946)

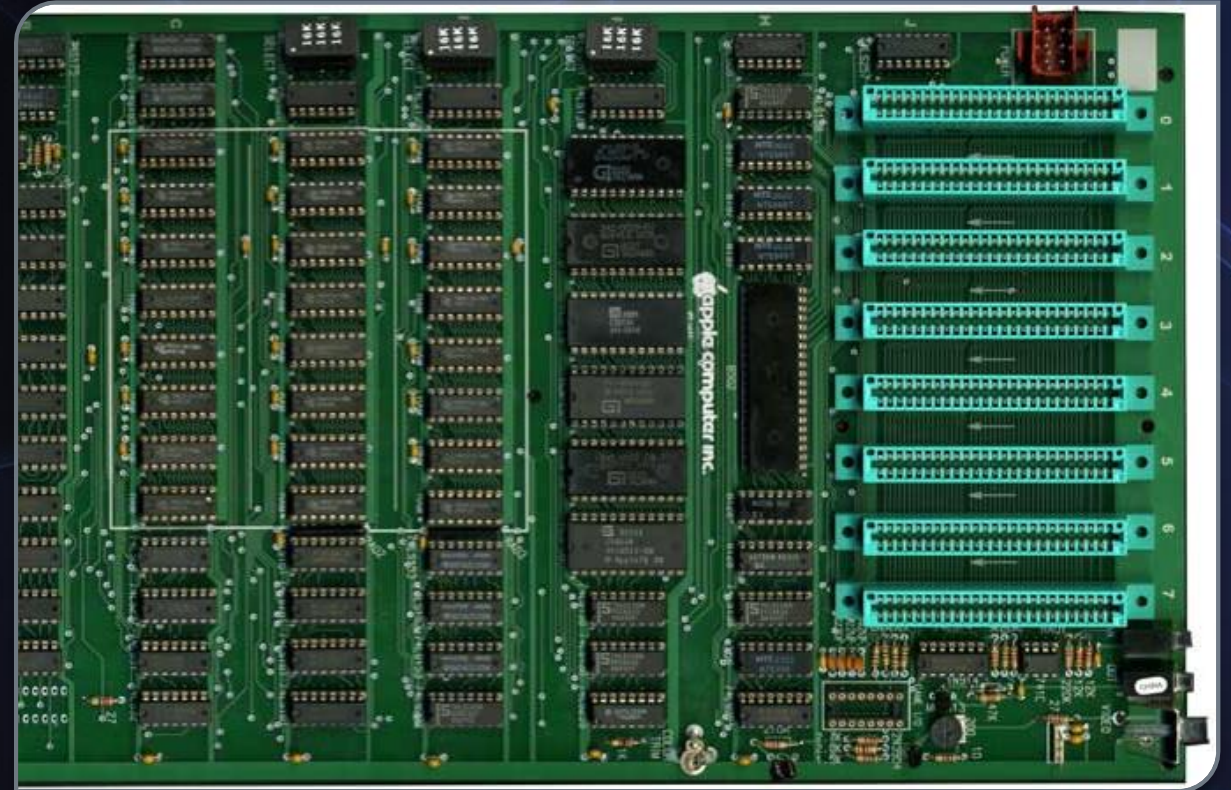
- First general-purpose computer
 - Used Base-10 instead of Binary (Base-2)
- They used Vacuum Tubes and Mechanical Switches
- Used to calculate firing-tables for the military.

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.

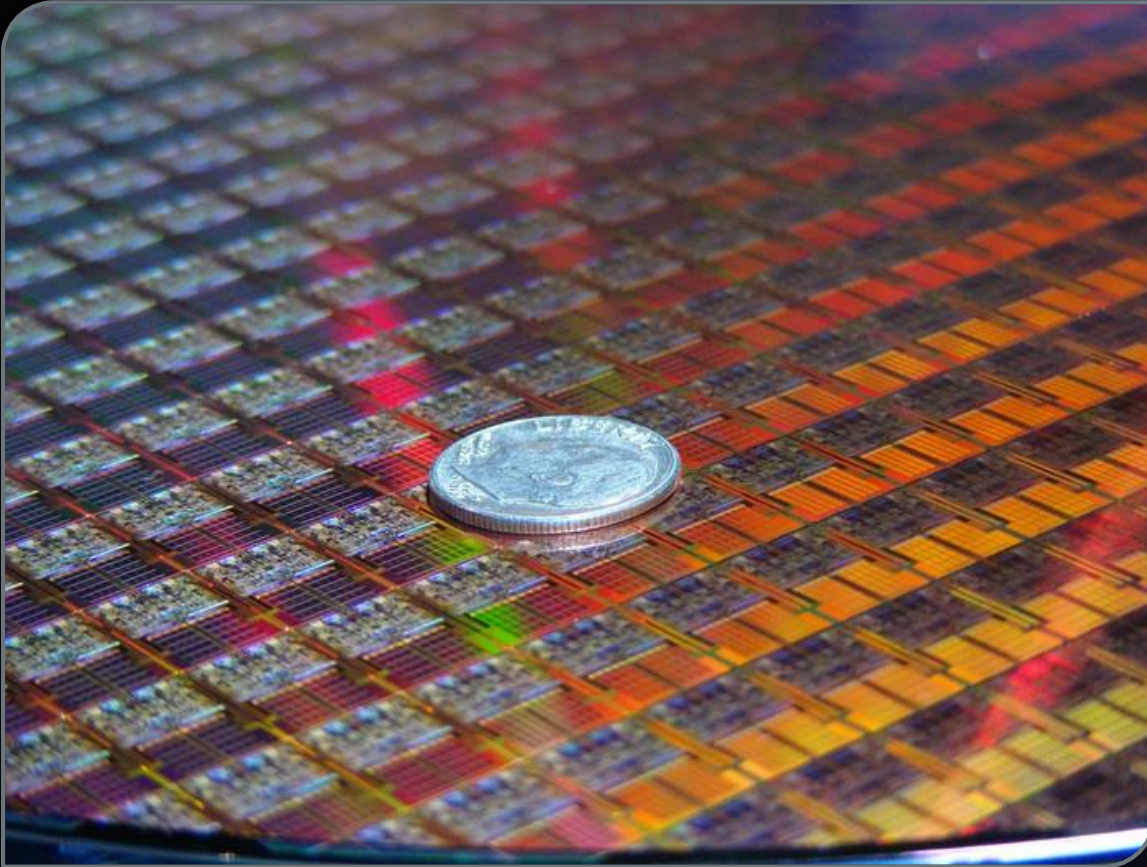
PRE-MODERN COMPUTERS

After Vacuum Tube and Mechanical Switched computers, **Integrated Circuits (ICs)** allowed computers to get much smaller. Computers went from the Size of buildings to the size of desks. This also allowed more powerful computers to be built because less space was needed.



This is an Apple2 motherboard from the late 1970s.
All the black chips make the CPU.
Each one is about 1" wide.

MODERN COMPUTERS



The **Microprocessor** allowed computers to go from the size of desks to the size of a dime!

Each small square in this picture is a computer!

This allowed use to make computers even more powerful and allow us to use even more powerful language features.

Each "switch" in these chips are 10nm wide

HISTORY OF PROGRAMMING

```
sahat@echo: ~/Dropbox/LearnC — ..ropbox/LearnC — zsh — 80x24
0749eb90 f0 32 7d 60 95 48 d0 62 08 80 4b 67 b4 4a 21 dc |.2}`.H.b..Kg.J!.|
0749eba0 80 3f 6c dd 4a f5 a3 d4 ce 32 8d e4 21 d7 a5 5a |. ?l.J...2...!..Z|
0749ebb0 92 93 4b f1 ca 0a ce 3c b9 14 20 a5 00 a4 4a 3e |..K...<...J>|
0749ebc0 bd 4b 8c b4 d1 90 2b 25 a9 c8 f4 c8 10 85 fb d6 |.K...+%.....|
0749ebd0 fc 2a 1f c6 8a 7f 25 e7 47 f4 95 01 e2 d7 82 fe |.*....%.G.....|
0749ebe0 22 95 fa 8e 49 e4 50 98 d3 84 95 a7 97 1d 97 92 |"...I.P.....|
0749ebf0 25 32 9f 90 0c a9 07 73 c2 2b 49 06 4c 1a 26 69 |%2....s.+I.L.&i|
0749ec00 b2 75 3e 20 db 65 bf 22 68 cf 29 1b 8a 65 8d 54 |.u>..e."h.)..e.T|
0749ec10 91 ba 33 f3 05 59 07 39 cd 43 96 6f 5d 88 bb 7a |..3...Y.9.C.o]..z|
0749ec20 aa ae d2 04 b1 c6 33 25 8c 68 f7 c7 79 23 ef 66 |.....3%.h..y#.f|
0749ec30 7a aa 41 e7 99 55 1d 46 79 64 2a 6c 1f a9 64 63 |z.A...U.Fyd*L..dc|
0749ec40 ef f9 87 72 3f d9 5a 9f 48 0d 92 96 72 0d 1b a4 |...r?.Z.H...r...|
0749ec50 a6 2e 08 b0 96 cc e6 37 88 f0 57 32 3b 21 6d d9 |.....7...W2;!m.|
0749ec60 e4 6b f1 ef 14 25 65 e3 3c b3 ee 60 bc a4 ea 44 |.k...%e.<...`...D|
0749ec70 64 49 0d 59 0b 45 3f f0 75 a4 24 be 41 f5 52 ad |dI.Y.E?.u.$..A.R.|
0749ec80 32 65 33 4d 9c 83 8e 97 69 57 f2 5d 72 93 dd b1 |2e3M....iW..]r...|
0749ec90 d0 c6 dc c8 43 89 6e 1e 8b d9 2e 67 52 3e 26 3f |....C.n....gR>&?|
0749eca0 46 cc 92 a7 e1 f3 af 9c c8 b3 17 fe ff 8a bb 7a |F.....Z|
0749ecb0 f6 e9 99 6d 8b 24 dc 84 97 67 b6 d5 5b 73 a6 fc |...m.$...g..[s..|
0749ecc0 50 a6 cf fe 92 7d c3 2f 2e 7e e8 b7 8f 9b 71 5f |P....}./~....q_|
0749ecd0 b0 43 79 5c f1 63 9d b7 2f 7e b1 f3 f6 87 5f b0 |.Cy\c.../~...._|
0749ece0 64 84 86 98 59 f7 d2 96 42 28 5a 96 8e d1 17 4f |d...Y...B(Z....0|
0749ecf0 f4 2d a6 94 06 0f fb 57 83 fe 60 59 8e 32 70 23 |..-....W...`Y.2p#|
0749ed00 c1 8a 98 43 0b 90 26 24 03 ce 3d 21 79 0b 75 f9 |...C...&$...!y.u.|
```

- 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 in ASM
- There is a different Assembly Language for each CPU design, so it is not portable code.
 - Why is portable code good?

```
CPU - main thread, module AW5
00401000  5 B8 00000000  MOV EAX,0
00401005  66:A1 00304000  MOV AX,WORD PTR DS:[403000]
00401008  B8 00000000  MOV EBX,0
00401010  66:8B1D 02304000  MOV BX,WORD PTR DS:[403002]
00401017  66:891D 00304000  MOV WORD PTR DS:[403000],BX
0040101E  66:A3 02304000  MOV WORD PTR DS:[403002],AX
00401024  6A 00  PUSH 0
00401026  E8 430D0000  CALL <JMP.&KERNEL32.ExitProcess>
00401028  CC  INT3
0040102C  55  PUSH EBP
0040102D  8BEC  MOV EBP,ESP
0040102F  53  PUSH EBX
00401030  51  PUSH ECX
00401031  52  PUSH EDX
00401032  E8 3D0D0000  CALL <JMP.&KERNEL32.GetProcessHeap>
00401037  A3 0C304000  MOV DWORD PTR DS:[40300C],EAX
0040103C  FF75 08  PUSH DWORD PTR SS:[EBP+8]
0040103F  FF35 10304000  PUSH DWORD PTR DS:[403010]
00401045  FF35 0C304000  PUSH DWORD PTR DS:[40300C]
00401048  E8 3D0D0000  CALL <JMP.&KERNEL32.HeapAlloc>
00401050  0BC0  OR EAX,EAX
00401052  75 0E  JNZ SHORT AW5.00401062
00401054  68 14304000  PUSH AW5.00403014
EAX=FA20143D
```

ExitCode = 0
ExitProcess
GetProcessHeap
HeapSize
Flags = HEAP_...
hHeap = NULL
HeapAlloc
ASCII "Cann...

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

PROGRAMMING NOW

How many languages do you recognize?

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

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

It is important to match the type of work to the correct language to insure the best results.

Different types of languages have different purposes.

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 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++

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

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.

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.

PROGRAMMING LANGUAGE TYPES

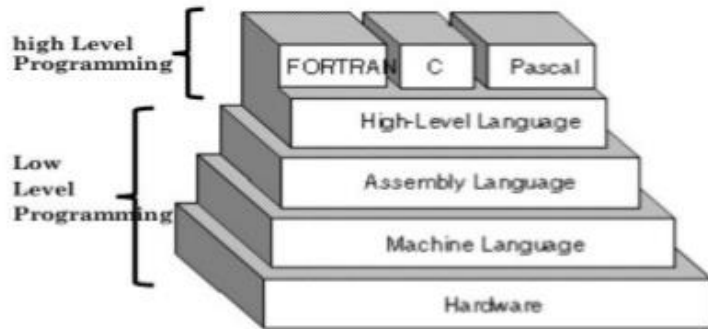
Languages can be split into a three different levels..

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

PROGRAMMING LANGUAGE TYPES

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

Types of Programming Language



Note: Java, Python, etc.
are one level higher than
FORTRAN, C and PASCAL

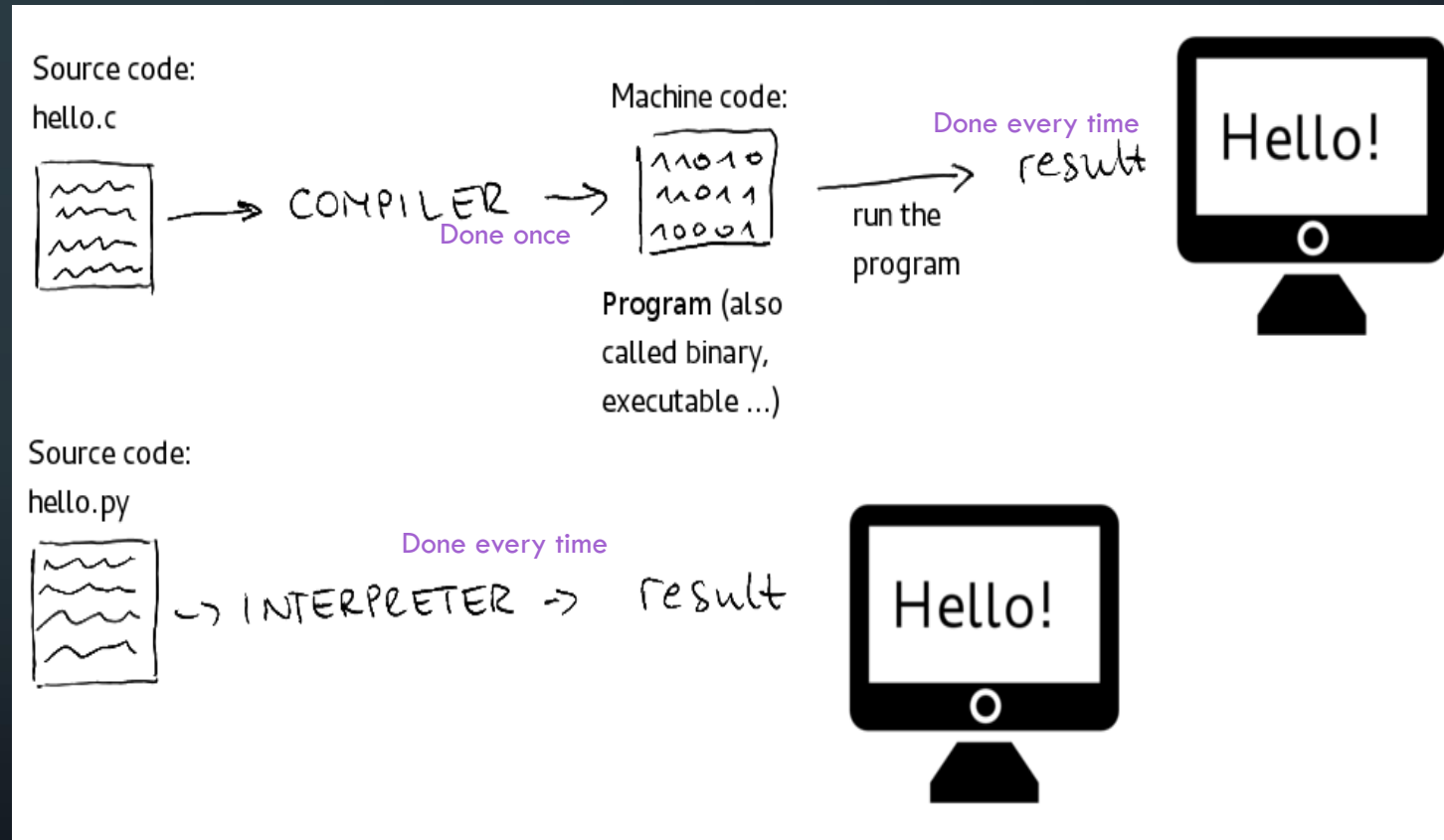
- Low Level
 - Pros: Fast Execution, No Overhead, Single Platform, **Compiled**
 - Cons: Hard to read, write, debug, and maintain
 - Examples: **ML**, **MASM**, **TASM**, **NASM**, **MIPS**
- High-Level
 - Pros: Easier to read, write, debug, and maintain, Multi-Platform, **Compiled** or **Interpreted**
 - 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 graphic shows 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?



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?

LANGUAGES & THEIR USES

- C
 - Computer applications
 - Embedded software
 - Compilers
 - Unix kernel
- C++
 - Software
 - Video games
 - Adobe systems
 - Most MS apps
 - Browsers
- Java
 - Minecraft
 - social media
- Financial services
- E-commerce
- Insurance applications
- JavaScript
 - Video game development
 - Interactive web functions
- HTML
 - Backbone of internet
 - Website presentation
- CSS
 - Web styling
- Python
 - Scientific and numeric computing
- Web and Internet Development
- Teaching programming
- Software Development
- Desktop GUIs
- Perl
 - Web automation
 - Database
 - Email handling
 - Text generation
 - System Administration

INTELLECTUAL PROPERTY (IP)

What are the four types of IP?

Open your computers and go online.
Find the definitions of all four of these types of IP.

1. Copyright
2. Patent
3. Trademark
4. Trade Secret

INTELLECTUAL PROPERTY (IP)

What is ...

1. Copyright – protects a particular expression of an idea that the author created (i.e. PowerPoints, Game Art, Specific Code)
2. Patent – protects useful innovative processes or methods, machines, manufactured items, or “compositions of matter” (i.e. a new and revolutionary math algorithm used in an app)
3. Trademark – protects a word, phrase, symbol or sound that identifies and distinguishes the source of a particular product or service (i.e. Windows Logo, “Your mattress is freeee”, etc.)
4. Trade Secret – protects valuable information be not disclosing it to anyone, enforced by a contract called a NDA (i.e. what info Facebook collects)

OWNING VS 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 can do what ever you want to the software. Most people do not own software.
- Licensing is where you “buy or get permission” to use the software, often subscription based.

LICENSE TYPES EXPLAINED

Use your computers to research the following key terms...

- Open-Source –
- Closed-Source –
- Freeware –
- Shareware –
- Demo –
- Public Domain –

LICENSE TYPES EXPLAINED

Use your computers to research the following key terms...

- Open-Source – the code is exposed to the public and can be modified or distributed, may be limits or restrictions (doesn't mean free).
- Closed-Source – the code is NOT exposed to the public and cannot be edited or distributed (doesn't mean free).
- 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.
- 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.

CAREERS IN PROGRAMMING

What are some careers you have heard of in a programming field?

- Computer Scientist
- Mobile App Developer (dev)
- Desktop App Dev
- UI / UX Engineer
- Software Engineer
- Game Engine Dev
- Gameplay Dev
- Database Engineer
- Hardware Engineer
- Computer Engineer
- Sysadmin
- Hacker / Pen-Tester
- Web Dev (frontend and backend)

PROJECT TIME

Requirement: Write / modify programs using languages from 3 different industries.

Arduino

C

C#

C++

ColdFusion

Java

Javascript

Labview

Ladder Logic

Lisp

Lua

Matlab

Objective-C

Perl

PHP

Python

ROBOTC

Romotive

Ruby

Scratch

Simulink

StateFlow

VBasic

PROJECT TIME - PERL

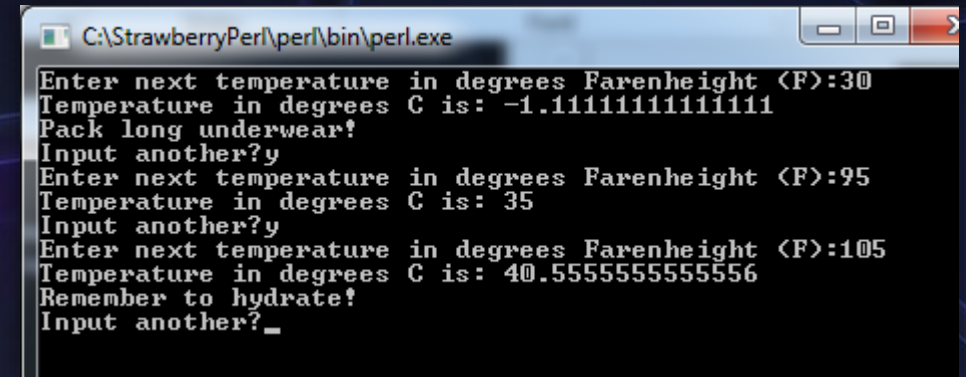
Industry 1: Science

Language: Perl

Project: Temperature Converter (F to C)

Requirement: Get user input of temp in F, convert to C and give them feedback on what to wear based on the temp.

Bonus: Change the program to convert from C to F (or both) and ask for user's name

A screenshot of a Windows command prompt window titled "C:\StrawberryPerl\perl\bin\perl.exe". The window shows the execution of a Perl script that converts Fahrenheit to Celsius. The script prompts the user for a temperature in Fahrenheit, calculates the Celsius equivalent, and provides feedback based on the temperature. The output shows three iterations: 30°F converts to -1.11°C with feedback "Pack long underwear!"; 95°F converts to 35°C with feedback "Input another?y"; and 105°F converts to 40.55°C with feedback "Remember to hydrate!".

```
C:\StrawberryPerl\perl\bin\perl.exe
Enter next temperature in degrees Farenheight (F):30
Temperature in degrees C is: -1.111111111111111
Pack long underwear!
Input another?y
Enter next temperature in degrees Farenheight (F):95
Temperature in degrees C is: 35
Input another?y
Enter next temperature in degrees Farenheight (F):105
Temperature in degrees C is: 40.5555555555556
Remember to hydrate!
Input another?_
```

PROJECT TIME - PERL

Download software: <http://www.perl.org/>

1. Install the Perl compiler on your computer or laptop. Recommend using “Strawberry Perl” for Windows.
2. Open a text editor (notepad, textedit, vi, or your favorite) and copy/paste the provided text into a file.
3. Save it as “Temp.pl” to your desktop.
4. Right-click the file and choose to “Execute Perl Program” or follow steps on next page.

PROJECT TIME - PERL

4. Open a command window (cmd.exe, or Unix shell) on your computer and navigate the default directory to the place where you stored “Temp.pl”
 - Use commands like a real programmer:
 - cd to change directory
 - dir to get a directory listing (dir /ON will sort the directories in alphabetical order)
 - Hint: start with “cd C:\Users” then type dir to find the next level, etc. until you get to the desktop → TAB is a shortcut to scroll thru the directories
5. Run the program with the command “perl Temp.pl”

PROJECT TIME - PYTHON

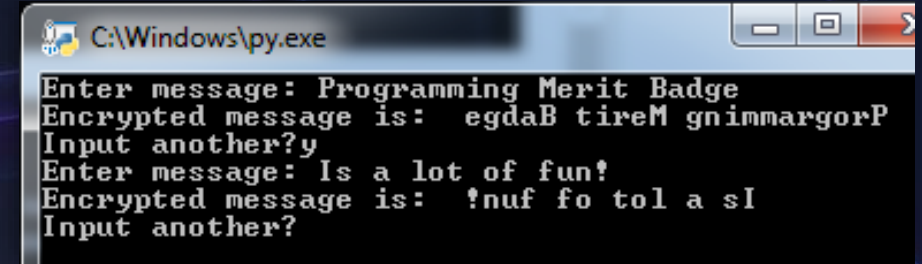
Industry 2: Business

Language: Python

Project: Simple encryption

Requirement: Make a program that will take user input and reverse all the characters, then display the output on screen.

Bonus: Convert each character to a 2-digit number instead. Consider how to handle special characters like spaces and punctuation. See if you can send a message to another member of the class.



```
C:\Windows\py.exe
Enter message: Programming Merit Badge
Encrypted message is:  egdaB tireM gnimmargorP
Input another?y
Enter message: Is a lot of fun!
Encrypted message is:  !nuf fo tol a sI
Input another?
```

PROJECT TIME - PYTHON

1. Install the Python compiler and tools on your computer or laptop. This is available in a single download file from python.org. For Mac OS X and some Linux computers, Python is already installed.
2. Open a text editor (notepad, textedit, or your favorite) and copy/paste the provided program into a file. If you installed Python into a Windows computer, it included a program called IDLE, which can edit and run Python programs.
3. Save it as “Flip.py” to your desktop.
4. You can often double-click to open the program in Python, or follow steps on next page.

PROJECT TIME - PYTHON

4. Open a command window (cmd.exe, or Unix shell) on your computer and navigate the default directory to the place where you stored “Flip.py”
 - Use commands like a real programmer:
 - cd to change directory
 - dir to get a directory listing (dir /ON will sort the directories in alphabetical order)
 - Hint: start with “cd C:\Users” then type dir to find the next level, etc. until you get to the desktop → TAB is a shortcut to scroll thru the directories
5. Run the program with the command “python Flip.py”. If you are running IDLE, then just select the menu option Run -> Run Module.

PROJECT TIME - JAVASCRIPT

Industry 3: Internet / Social Media

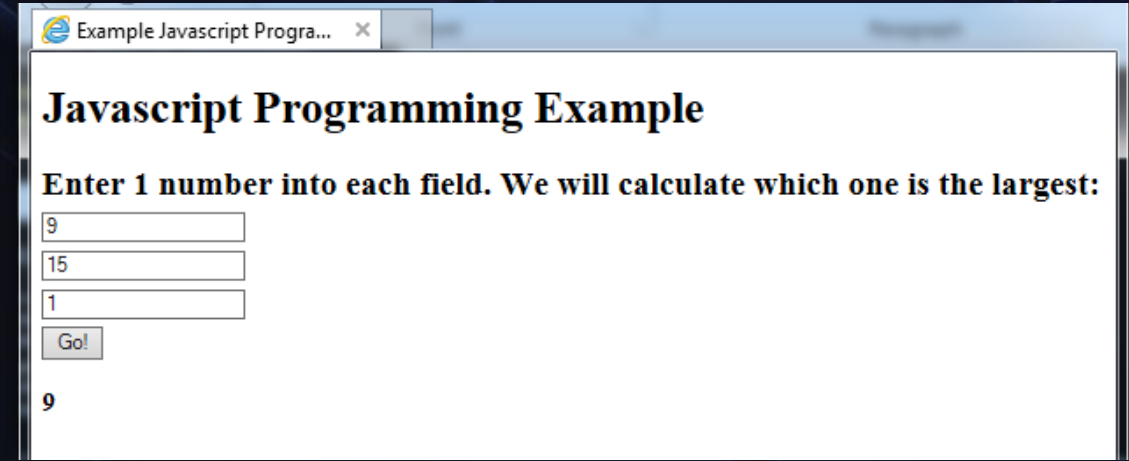
Language: Javascript +

HTML wrapper

Project: Biggest number

Requirement: Get user input of 3 numbers. Compare the numbers and return the largest number.

Bonus: Change the program to return the smallest number instead.



Example Javascript Progra... x

Javascript Programming Example

Enter 1 number into each field. We will calculate which one is the largest:

9

15

1

Go!

9

PROJECT TIME - JAVASCRIPT

1. Nothing to install.
2. Open a text editor (notepad, textedit, or your favorite) and copy/paste the provided text into a file. Save it as “ThreeNums.htm” to your desktop.
3. You can open this by double clicking (default browser) or but right-clicking to select the specific browser you want to use.