TROOP 510

Merit Badge Clinic

Prepared. For Life.®

SCOUTS | BSA

What is the hidden message?

Prepared. For Life.®

SCOUTS | BSA


“BE CURIOUS”
EXPERIMENT.
FAIL.
LEARN.
REPEAT.

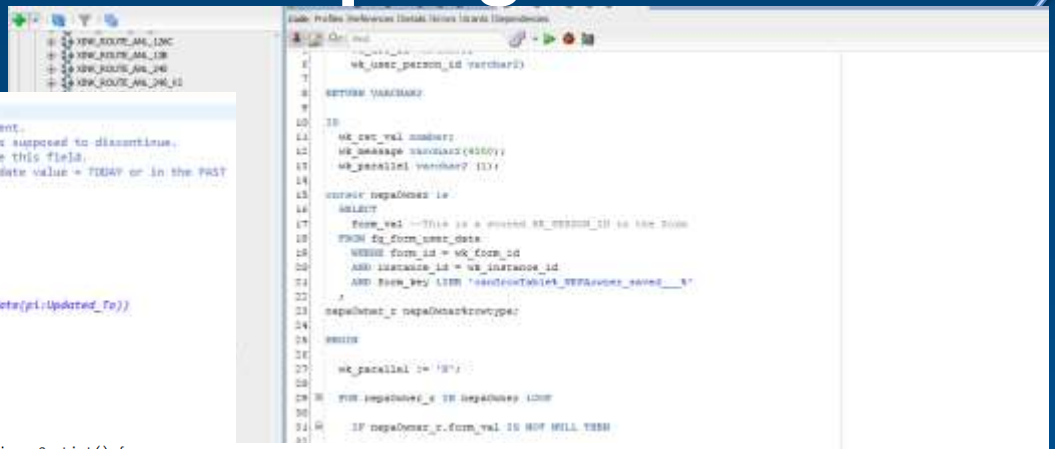

Code powers tomorrow
MIRA
And you can be a part of the excitement!

A day in the life of a programmer

- **Support daily work for all lab scientists and employees.**
- **Develop apps to run the lab's operations.**
- **Collaborate with my team in software design and problem solving.**

- ***Every day…***
  - Code & Test
  - Discuss solutions with users across the lab
  - **Debug, debug, debug**

A day in the life of a programmer



```
<!-- Apply Earnings Deduction template -->
<!-- [7.20.2017 | mtd]: Add filter to ignore records that include the 'First_Day_No_Longer_Applies' element.
     Workday is including this tag sometimes, and it represents a previous date that the deduction was supposed to discontinue.
     If the record is passed to ADP, then the deduction will be applied because ADP will not recognize this field.
[8.8.2017 | mtd]: pi:First_Day_No_Longer_Applies if this value should be filtered out only when its date value = TODAY or in the PAST
-->
<xsl:apply-templates select="
   pi:Earnings_Deductions[
      pi:Earning_or_Deduction = 'D'
      and (   not(pi:First_Day_No_Longer_Applies)
              or
              (pi:First_Day_No_Longer_Applies and pi:Operation = 'REMOVE')
              or
              (pi:First_Day_No_Longer_Applies != '' and xs:date(pi:First_Day_No_Longer_Applies) > xs:date(pi:Updated_To))
      )
      and $position/pi:Worker_Type != 'CS'
      and (pi:Operation != 'MOVE'
              or .../pi:Earnings_Deductions[pi:Earning_or_Deduction = 'E'
                  and contains(current()/pi:Code_Name, pi:Code_Name)
                  and pi:Operation != 'MOVE']
              or .../pi:Status/pi:Staffing_Event[@pi:PriorValue] = 'PSI'
          )
   ]
"/>
</xsl:template> <!-- END pi:Employee template -->

<xsl:template match="pi:Earnings_Deductions">
   <xsl:variable name="credit" select="
         .../pi:Earnings_Deductions[
            pi:Earning_or_Deduction = 'E'
            and contains(current()/pi:Code_Name, pi:Code_Name)
            and pi:Operation != 'REMOVE'
         ]/pi:Amount
   "/>
   <Employee>
      <Co_Code><xsl:value-of select="$pay_group"/></Co_Code>
      <!-- if badge number starts with E make co it fails on import -->
      <xsl:choose>
         <xsl:when test="substring(pi:Summary/pi:Employee_ID,1,1)= 'E'">
            <File_Num><xsl:text>BAD BADGE</xsl:text></File_Num>
         </xsl:when>
         <xsl:otherwise>
            <File_Num><xsl:value-of select="substring(../pi:Summary/pi:
         </xsl:otherwise>
      </xsl:choose>
```

```
function generateSupervisoryOrgList() {

    var workerDef_selected = document.getElementById("workerDef");
    var supervisorBadge = document.getElementById("supervisorBadge");

    var supervOrg = document.getElementById("selectSupervisorOrg"); // Used for display to user
    var supervisorOrg_saved = document.getElementById("supervisorOrg"); // Used to pass through to backend processing

    if (supervisorBadge.value != "" && supervisorBadge != null) {

        if (workerDef_selected.value == "Employee") {
            // If the worker is to be an Employee, then pull the Supervisory Org for "Position Management"

            var urlSupOrg = '/xink/lookupData?tagName=temp&key=temp&sectionElementId=1073608&formId='
                + document.getElementById('hiddenFormId').value
                + '&instanceId='
                + document.getElementById('hiddenInstanceId').value
                + '&replaceString=PERSON_BADGE_NUMBER,'
                + document.getElementById('supervisorBadge').value;

            if (window.XMLHttpRequest) {
                var ajax = new XMLHttpRequest();
            } else {
                var ajax = new ActiveXObject('Microsoft.XMLHTTP');
            }

            if (ajax) {
                ajax.open('GET', urlSupOrg, false); // Automatically waits for the response or a timeout.
                ajax.send(null);
                var dataSupOrg = Ext.decode(ajax.responseText);

                if (dataSupOrg && dataSupOrg.length > 0) { // If one or more
                    // results found
                    supervOrg.value = dataSupOrg[0].innerHTML;
                    supervisorOrg_saved.value = dataSupOrg[0].innerHTML2;
                } else {
                    // If no supervisory org is found, leave field blank and
                    // disable:
                    supervOrg.value = "";
                    supervisorOrg_saved.value = "None";
                }
            }

        } else if (workerDef_selected.value == "Contingent Worker") {
            // If the worker is to be a Contingent Worker, then pull the
            // Supervisory Org for "Job Management"
```

SCOUTS | BSA

```
if (supervisorBadge.value != "" && supervisorBadge != null) {

    if (workerDef_selected.value == "Employee") {
        // If the worker is to be an Employee, then pull the Supervisory Org for

        var urlSupOrg = '/xink/lookupData?tagName=temp&key=temp&sectionElementId=
            + document.getElementById('hiddenFormId').value
            + '&instanceId='
            + document.getElementById('hiddenInstanceId').value
            + '&replaceString=PERSON_BADGE_NUMBER,'
            + document.getElementById('supervisorBadge').value;

        if (window.XMLHttpRequest) {
            var ajax = new XMLHttpRequest();
        } else {
            var ajax = new ActiveXObject('Microsoft.XMLHTTP');
        }

        if (ajax) {
            ajax.open('GET', urlSupOrg, false); // Automatically waits for the re
            ajax.send(null);
            var dataSupOrg = Ext.decode(ajax.responseText);

            if (dataSupOrg && dataSupOrg.length > 0) { // If one or more
                // results found
                supervOrg.value = dataSupOrg[0].innerHTML;
                supervisorOrg_saved.value = dataSupOrg[0].innerHTML2;
            } else {
                // If no supervisory org is found, leave field blank and
                // disable:
```

We need you!

There are **13,647** open computing jobs in Illinois and only **2,596** CS graduates!

How fast is your math?

Get ready…

set…

How fast is your math?

1 + 1 = ?

Did it take you one second?

One-half of a second?

Prepared. For Life.®

SCOUTS | BSA

How fast is your math?

Argonne's supercomputer...

2 quintillion calculations

2,000,000,000,000,000,000

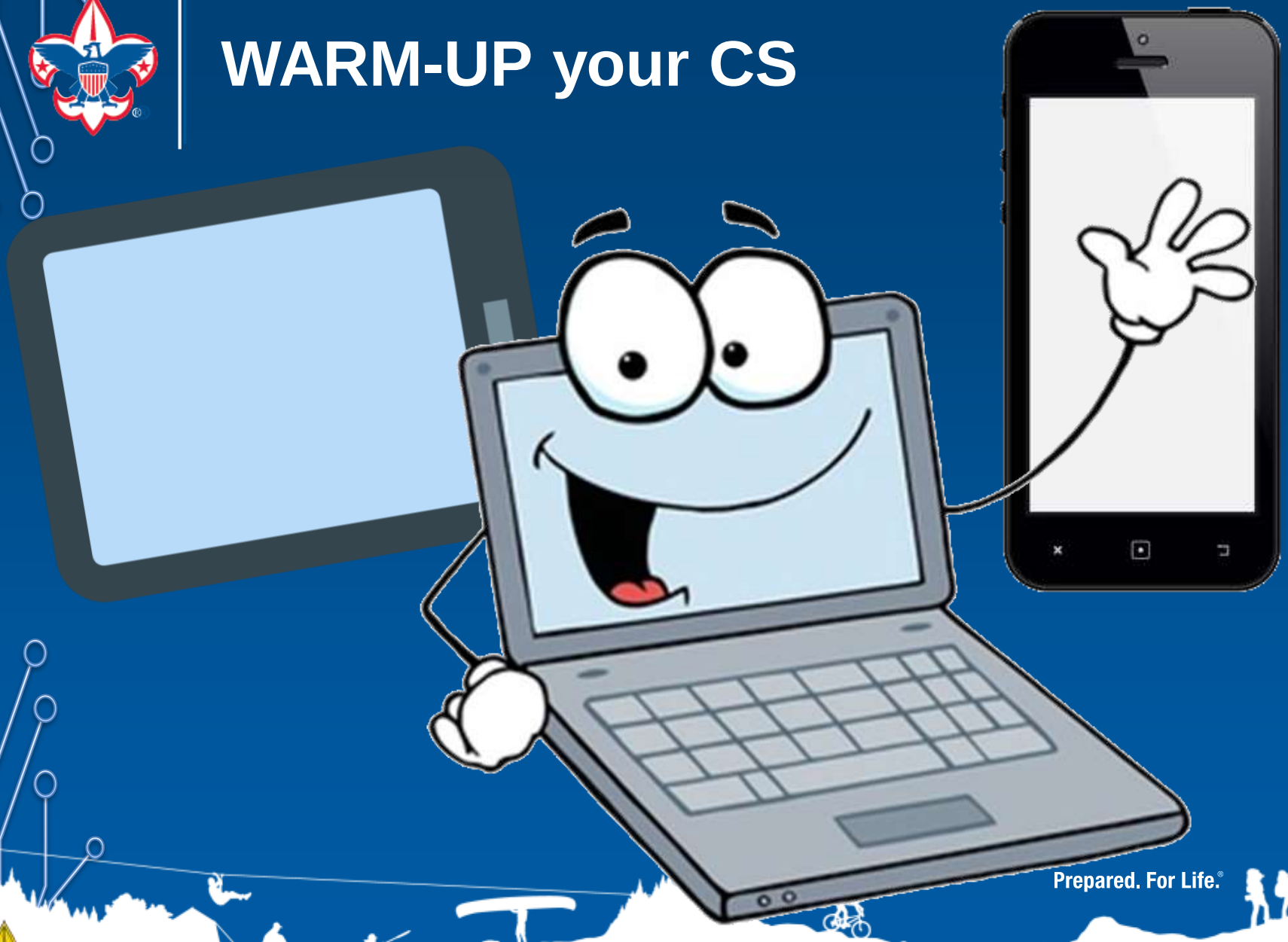
per second

Every human on Planet Earth
doing **1+1** in their head
without stopping for **8 years!**

Prepared. For Life.®

SCOUTS | BSA

WARM-UP your CS



Prepared. For Life.®

SCOUTS | BSA

WARM-UP your CS

Computer programs will

do something

based on a condition

What is a condition?

(we call them conditional expressions in cs)

Prepared. For Life.®

SCOUTS | BSA

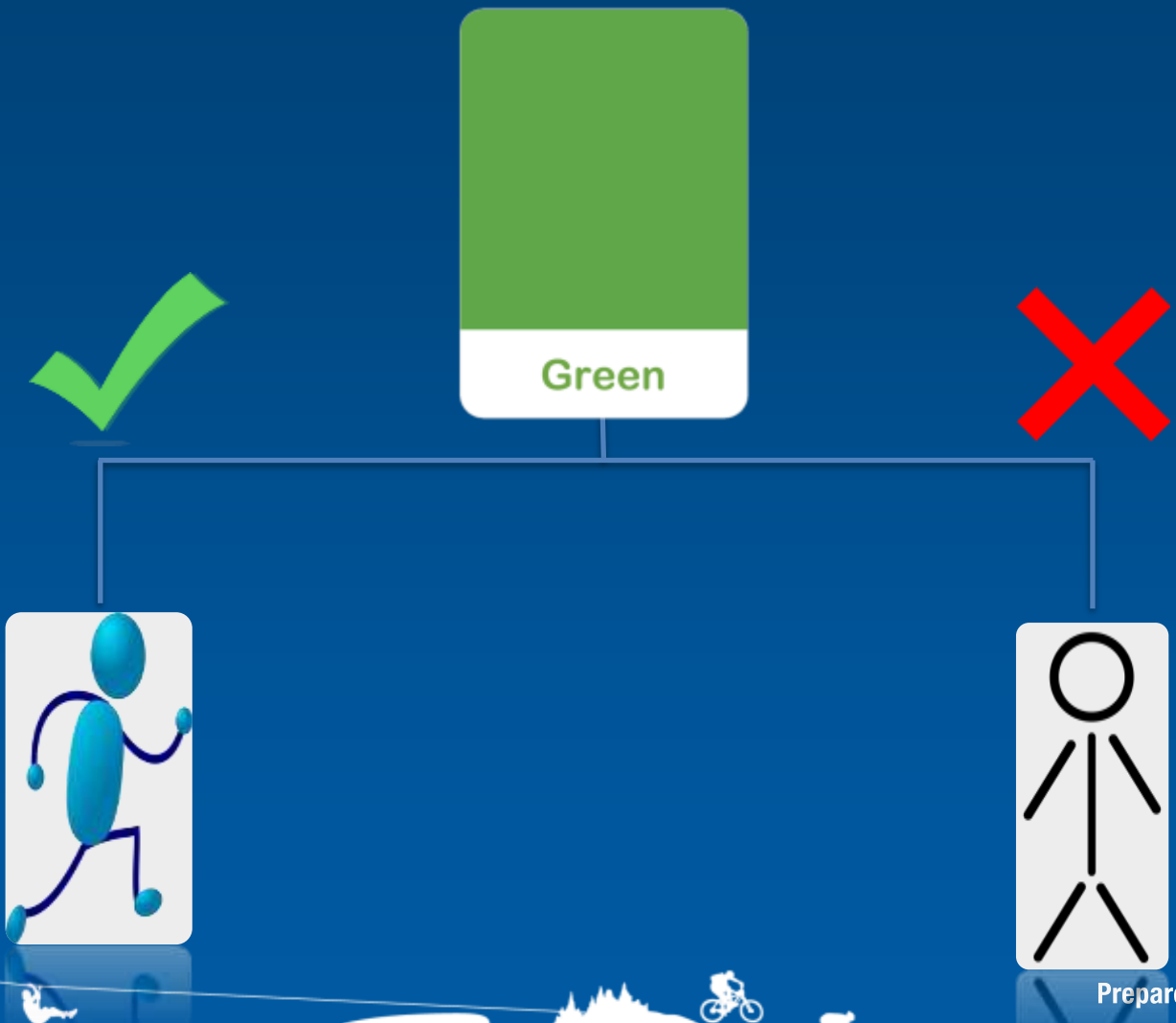
WARM-UP your CS

Let's try it!

You = computer

Me = the programmer

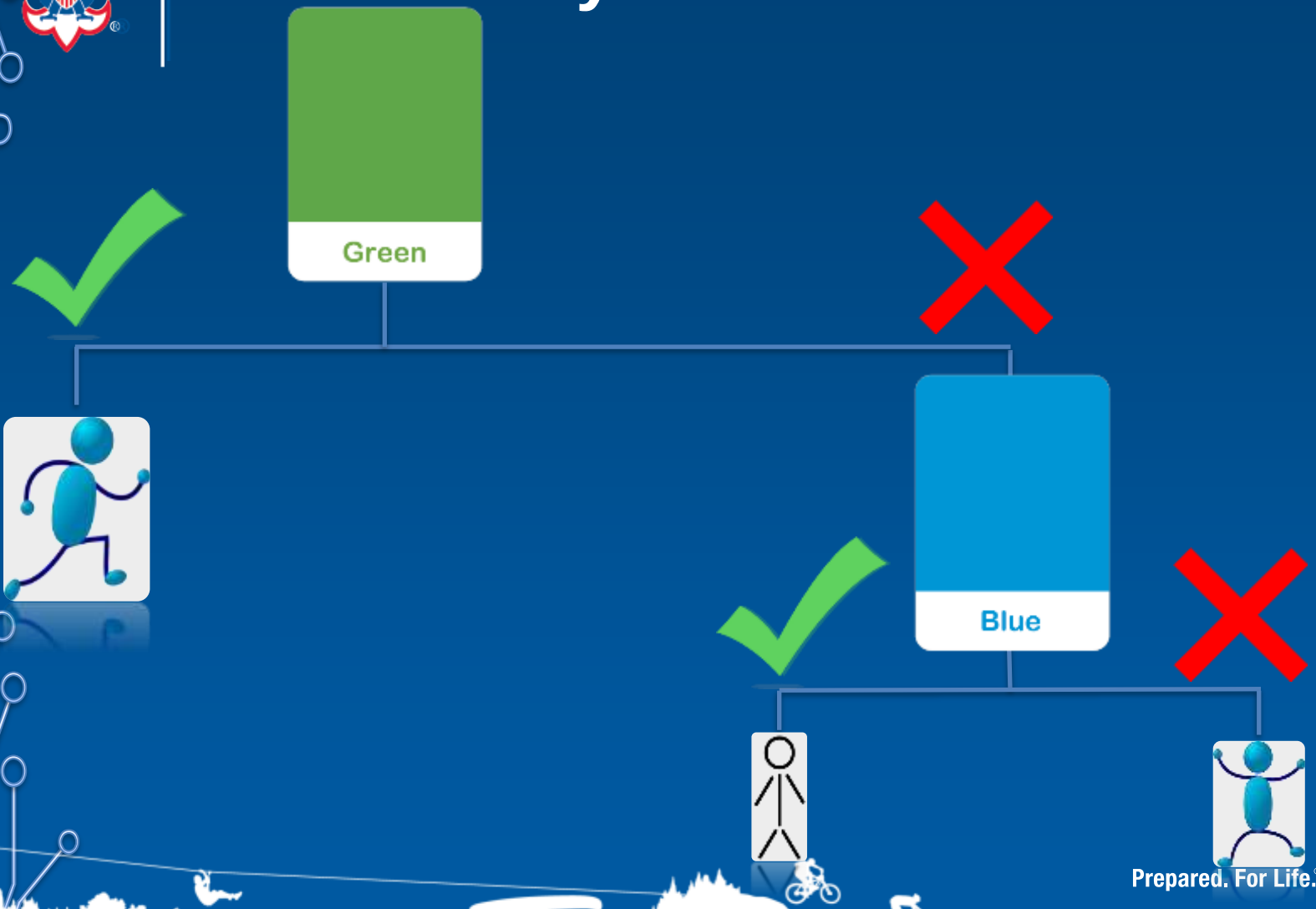
WARM-UP your CS



Prepared. For Life.®

SCOUTS | BSA

WARM-UP your CS



Programming Merit Badge

Requirements

1. **Safety**
   a. Cyber Chip
   b. First aid and injury prevention
2. **History**
   a. Programming and programming languages
   b. Evolution of programing languages
3. **General knowledge**
   a. Describe popular languages
   b. Programmed devices
4. **Intellectual property**
   a. Four types
   b. Licensing vs. Ownership
   c. Freeware vs. Open-source vs. Commercial
5. **Programming projects**
   a. 3 code projects: write or modify, debug, and demonstrate – different languages, different industries
   b. Explain how your code works
6. **Careers**
   a. Learn about 3 career paths

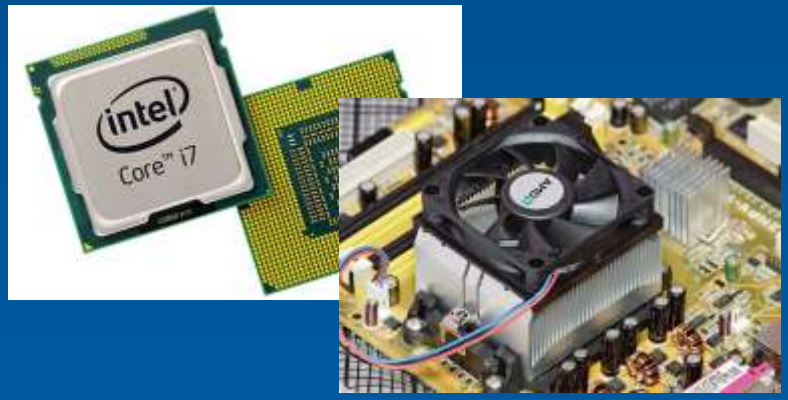
What is programming?

First, what is a program?

- A set of **instructions** to tell a computer *exactly what to do.*

```
if (dataSupOrg && dataSupOrg.length > 0) { // If one or more
    // results found
    supervOrg.value = dataSupOrg[0].innerHTML;
    supervisorOrg_saved.value = dataSupOrg[0].innerHTML2;
} else {
    // If no supervisory org is found, leave field blank and
    // disable:
    supervOrg.value = "";
    supervisorOrg_saved.value = "None";
}
```

- The computer is a **processor** that performs specific logical tasks.



What is programming?

OK, now… *what is a programming*?

- It's how humans **communicate** with computers.
- Code converts our **ideas** into **instructions** for computers to follow.
- Many different language exist today for writing code (around 700!), and they constantly evolve – just like human languages!
- This merit badge will cover several important aspects of programming and you will gain **experience doing real coding**.

Safety First… even when coding

Human – computer interaction is *not natural*.

- Programming typically occurs with a computer that requires electricity.
  - ***Important***: make sure all power cords are not frayed and keep liquids far away to prevent electric shock (or a fried computer).

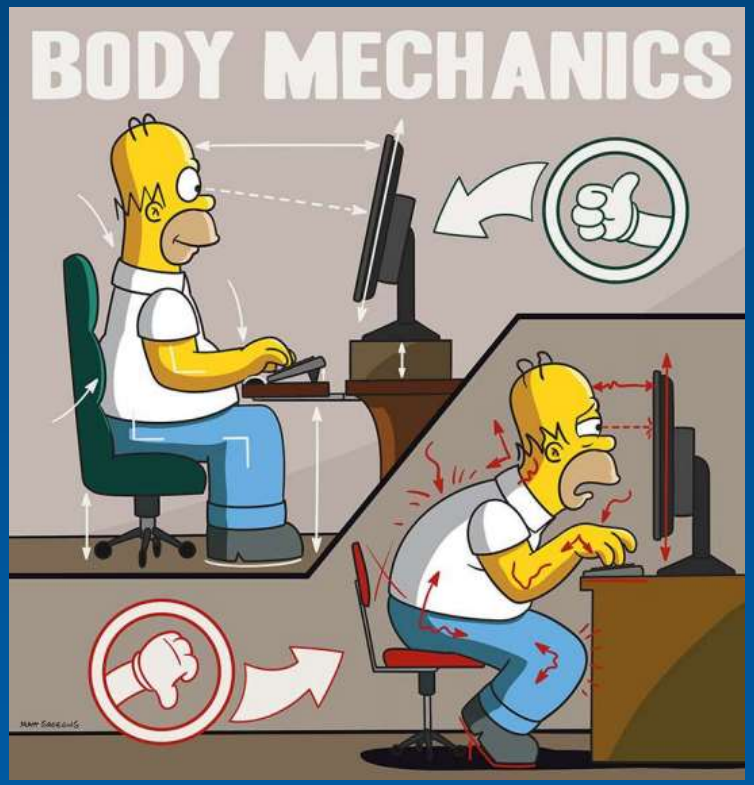
Safety First… even when coding

Human – computer interaction is *not natural*.

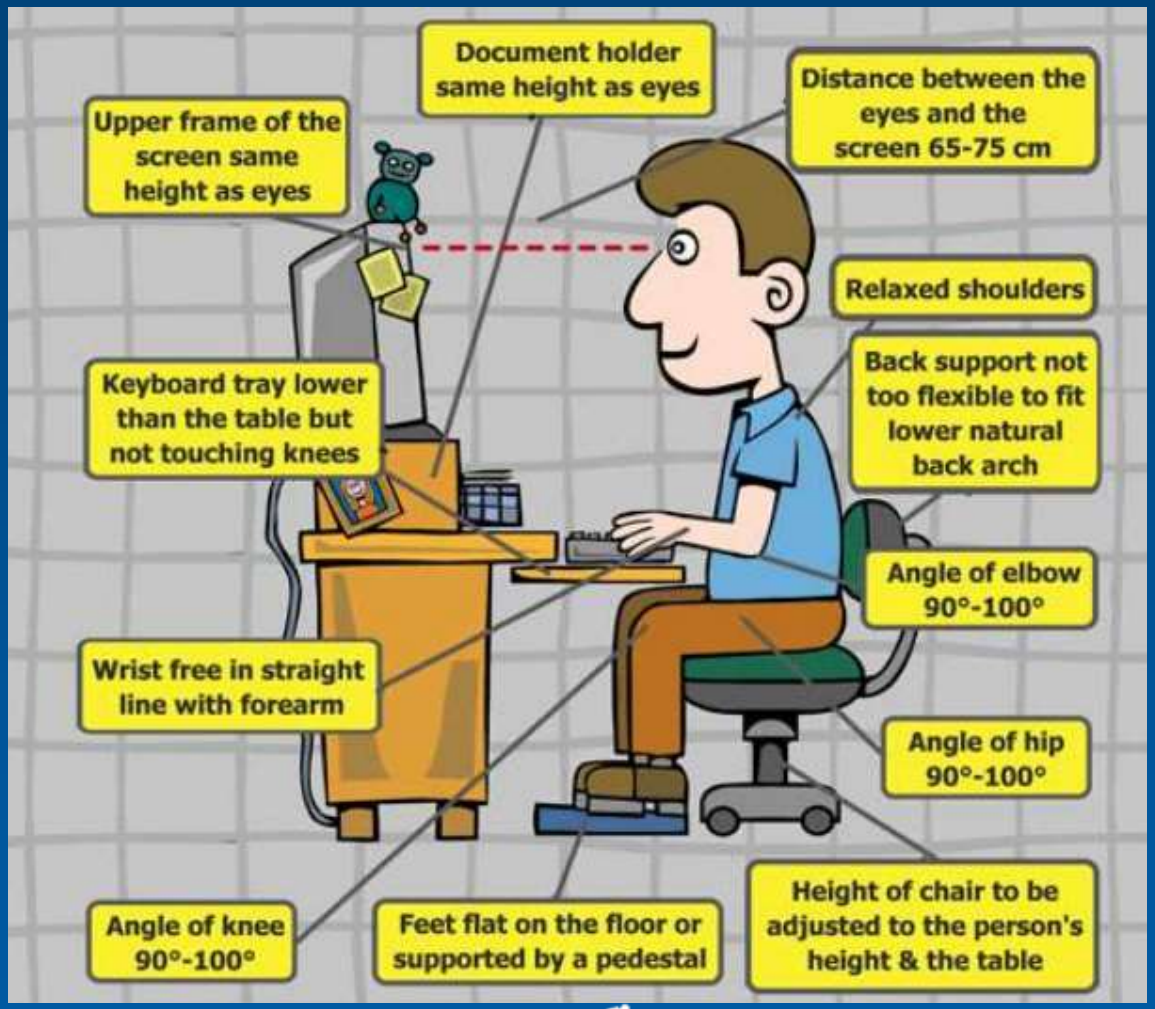
- RSI – Repetitive Stress Injury
  - Caused by typing for long periods of time, resulting in wrist and hand **pain**.
  - How can RSI be prevented?

Safety First… even when coding

Human – computer interaction is *not natural*.



Safety First… even when coding



History of Programing

Programming *before* computers?

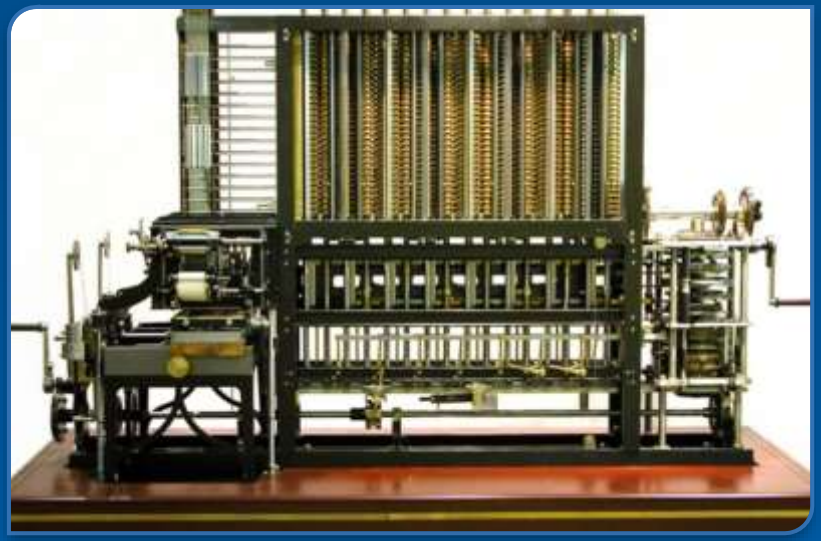
- Mechanical devices used in factories – long before the 1900s – were automated with programming.
- **1804**: The Joseph Jacquard Loom used hole-punched cards to “program” patterns into fabric.
- Jacquard’s tech was based on earlier work from **1725** that originated the ideas of punch cards for controlling a loom.

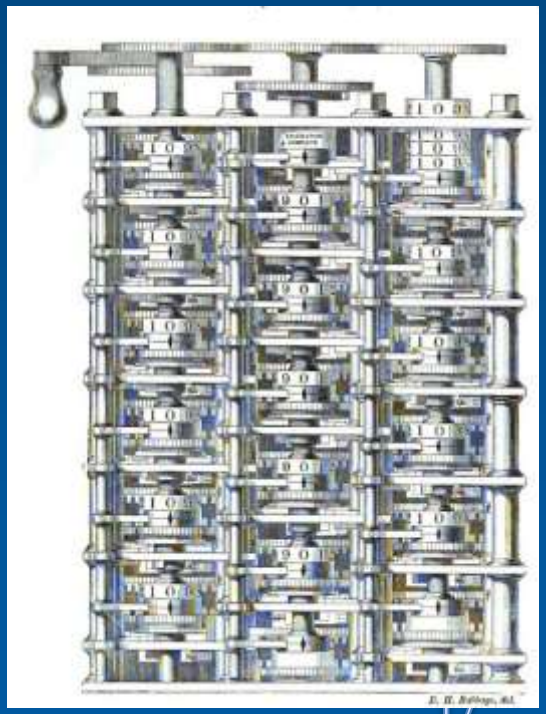
History of Programing

History of Programing

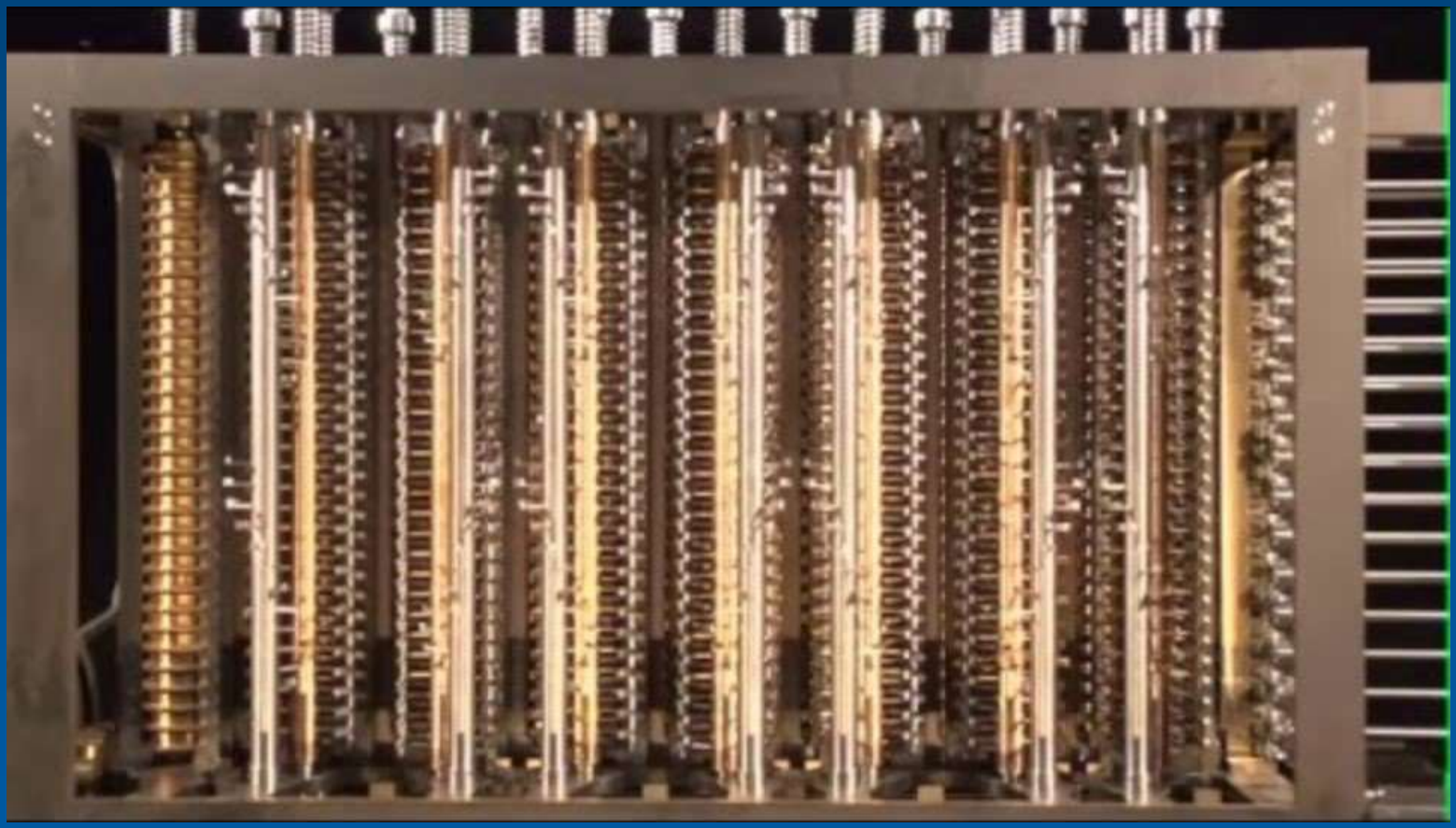
Programming *before* computers?

- **1822**: The Difference Engine by Charles Babbage
  - Used a hand crank to move complex series of gears configured to compute values – all based on subtraction (no multiplication/division required)!





History of Programing



History of Programing

Programming *before* computers?

- **1843**: Ada Lovelace – the first programmer
  - Babbage developed a new machine, called the Analytical Engine
  - Ada wrote an algorithm -- the logic for the machine to perform its calculations.

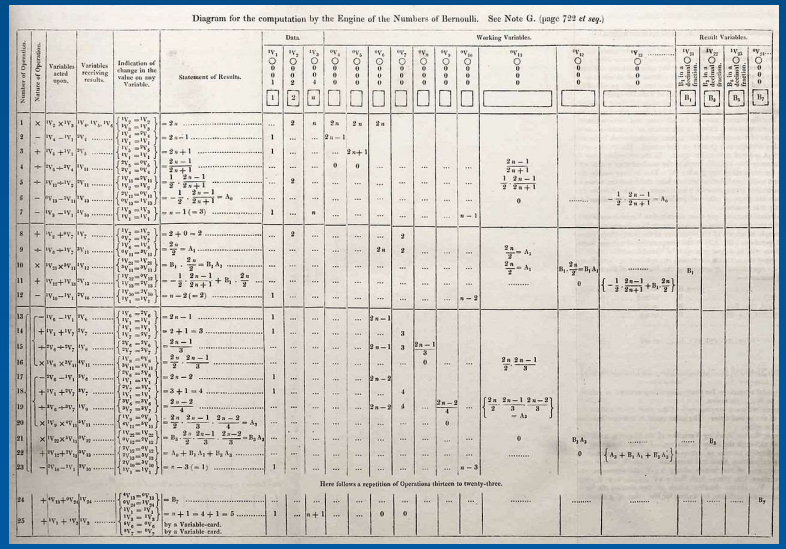


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

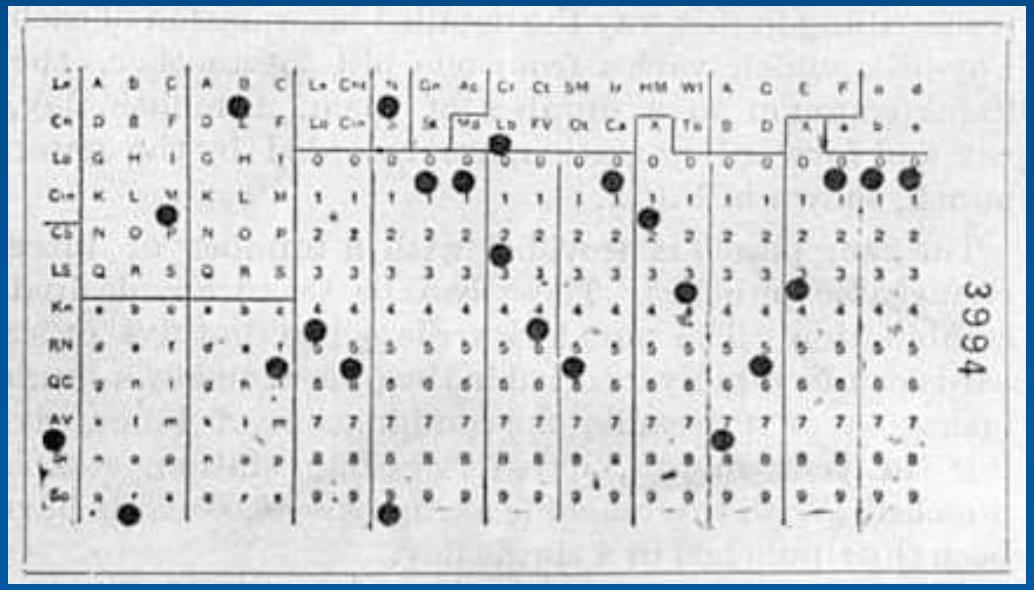
Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.									Result Variables.			
						$^1V_1$ 0 0 0 1 [1]	$^1V_2$ 0 0 0 2 [2]	$^1V_3$ 0 0 0 4 [n]	$^0V_4$ 0 0 0 0 [ ]	$^0V_5$ 0 0 0 0 [ ]	$^0V_6$ 0 0 0 0 [ ]	$^0V_7$ 0 0 0 0 [ ]	$^0V_8$ 0 0 0 0 [ ]	$^0V_9$ 0 0 0 0 [ ]	$^0V_{10}$ 0 0 0 0 [ ]	$^0V_{11}$ 0 0 0 0 [ ]	$^0V_{12}$ 0 0 0 0 [ ]	$^0V_{13}$ ..... 0 0 0 0 [ ]	$^1V_{21}$ B_1 in a decimal fraction. [B_1]	$^1V_{22}$ B_3 in a decimal fraction. [B_3]	$^1V_{23}$ B_5 in a decimal fraction. [B_5]	$^0V_{24}$ ..... 0 0 0 0 [B_7]
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \dfrac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\dfrac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2+0=2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \dfrac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\dfrac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} {}^1V_{21} = {}^1V_{21} \\ {}^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \dfrac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\dfrac{2n}{2} = A_1$	$B_1 \cdot \dfrac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} {}^1V_{12} = {}^0V_{12} \\ {}^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} + B_1 \cdot \dfrac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} + B_1 \cdot \dfrac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} {}^1V_{10} = {}^2V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} {}^1V_6 = {}^2V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2+1=3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} {}^2V_6 = {}^2V_6 \\ {}^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \dfrac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\dfrac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} {}^1V_8 = {}^0V_8 \\ {}^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \dfrac{2n}{2} \cdot \dfrac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\dfrac{2n}{2} \cdot \dfrac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} {}^2V_6 = {}^3V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} {}^2V_7 = {}^3V_7 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3+1=4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} {}^3V_6 = {}^3V_6 \\ {}^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \dfrac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\dfrac{2n-2}{4}$	...	$\left\{\dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{3} = A_3\right\}$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} {}^1V_9 = {}^0V_9 \\ {}^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} {}^1V_{22} = {}^1V_{22} \\ {}^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} {}^2V_{12} = {}^0V_{12} \\ {}^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$			
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} {}^2V_{10} = {}^3V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							
					Here follows a repetition of Operations thirteen to twenty-three.																	
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} {}^4V_{13} = {}^0V_{13} \\ {}^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_3 = {}^1V_3 \\ {}^5V_6 = {}^0V_6 \\ {}^5V_7 = {}^0V_7 \end{matrix}\right\}$	$= n+1 = 4+1 = 5$ ..... by a Variable-card. by a Variable-card.	1	...	$n+1$	...	...	0	0										

History of Programing

Programming *before* computers?

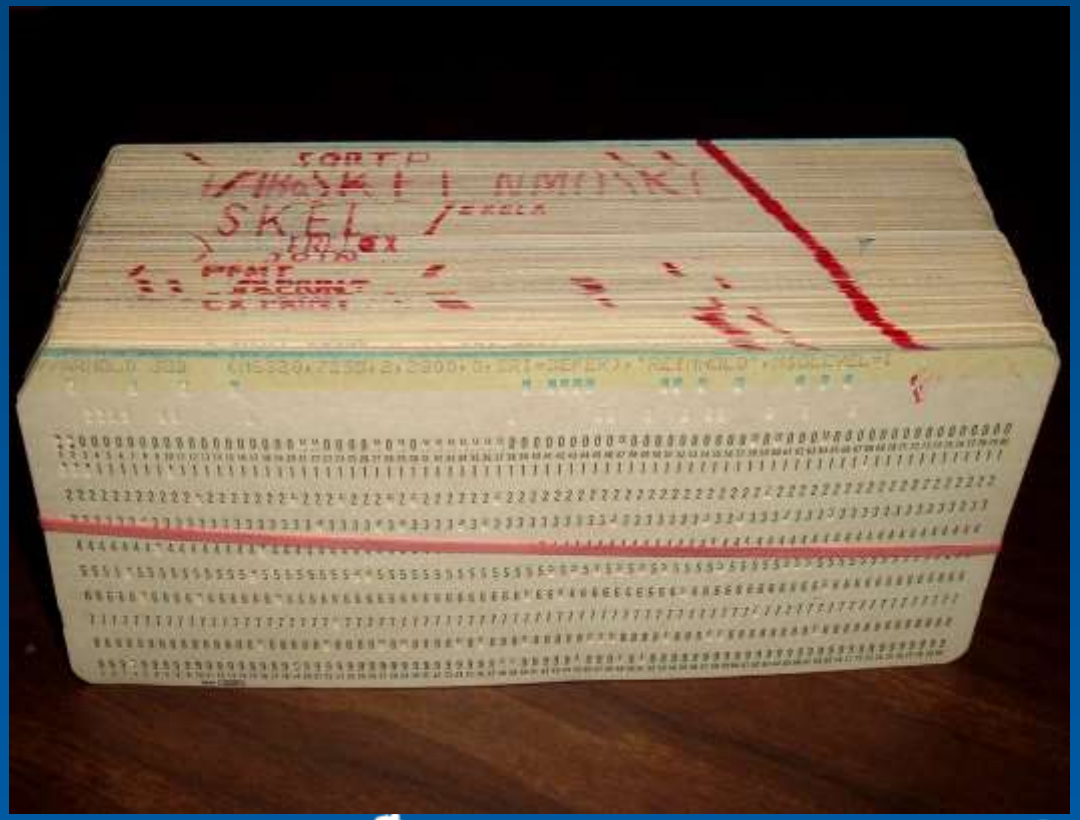
- Continued development with Punch Cards
  - Used for the 1890 Census
  - Became a standard in business computing by the **1950**s.



History of Programing

A computer program in 1969

- A deck of punch cards for an IBM 360.



Prepared. For Life.®

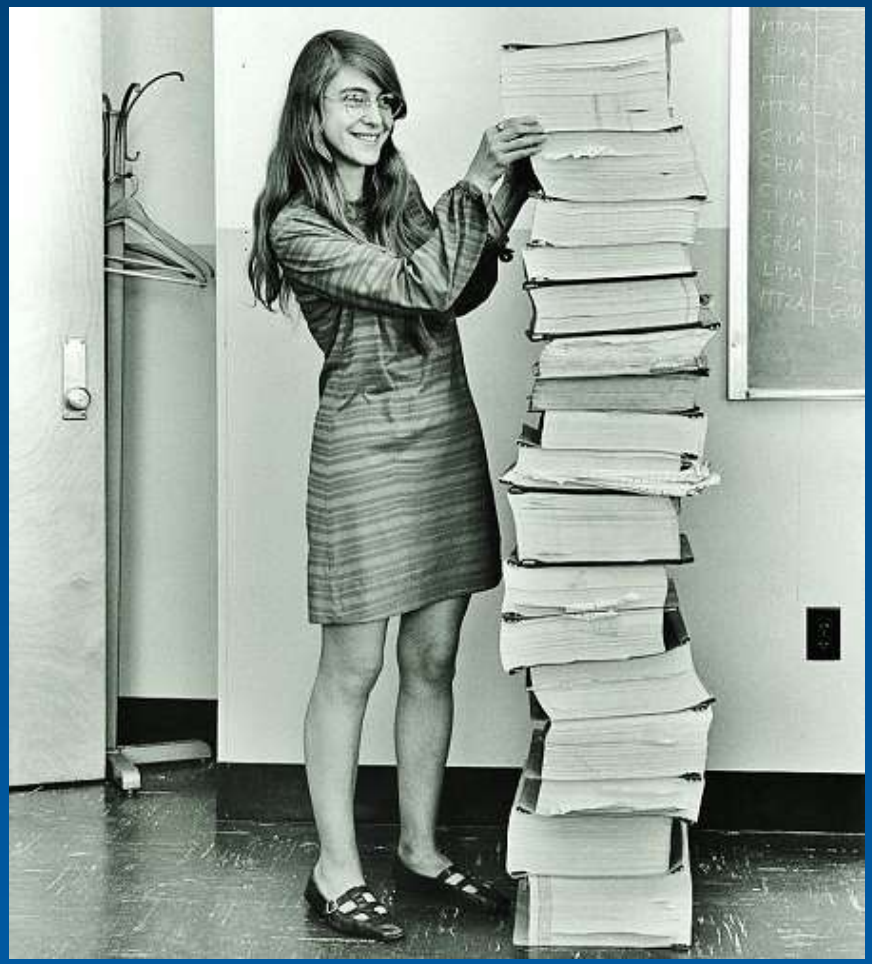
SCOUTS | BSA

History of Programing

Another computer program from 1969

- Margaret Hamilton stands next to a stack of program listings from the Apollo (11) Guidance Computer in a photograph taken in 1969. Wikimedia Commons

- They only had **72 kB** of computer memory to work with to land *humans on the Moon!*



History of Programing

John von Neumann (1903 – 1957) a programming pioneer

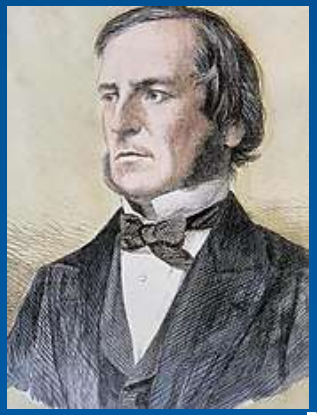
- Developed the idea of *conditional control*
  - Exactly what we did in our warm-up!
  - If/then statements

- Instead of writing one long piece of code, programmers began creating “blocks” of code that could loop (repeat) or be reused over and over by other parts of the code (called libraries).

History of Programing

The native language of a computer:
Binary – 1s and 0s

- A modern computer is based on the idea of turning electric current "on" or "off" – can be represented in writing (for us non-digital humans) as a 1 or 0.
- First developed in **1689** by Gottfried Leibniz, George Bool developed a system of algebra with 1s and 0s in **1847**.



History of Programing

The native language of a computer:
Binary – 1s and 0s

- A binary number as a “byte” of information (8 positions).
- Each position – called a “bit” – can be a 1 or a 0 *only*.
- Each bit represents a value that doubles at each position (right to left).
- The values are added *only* if that position is “on” or “1”

0	0	0	0	0	0	0	0
128	64	32	16	8	4	2	1

00000000 = 0
00000001 = 1
00000010 = 2
01000000 = 64
00001001 = 9
00101001 = 41

History of Programing

How can we write code in binary?!

- This is extremely hard (and punch cards were the first solution).

- **Assembly Language** – the beginning of modern programming
  - "names" and "characters" typed by a human are *translated* into 0s and 1s for a computer to read – called a **compiler**.

- The first compiler developed in 1952 by Grace Hopper (1906 – 1992).
  - Also helped build the first general purpose computer, UNIVAC I (UNIVersal Automatic Computer I).

History of Programing

How can we write code in binary?!

- Assembly Language is also very difficult to read and write.

- A flood of new programming languages were developed during the 1950s and 1960s (and ever since).
  - All are still based on the idea of *translating* human computational thinking into binary.

- Early evolution of coding languages improved:
  - Code portability between computer systems.
  - Easier to write, read, and debug code.
  - Allowed for new concepts (functions, classes, objects, OOP).
  - Explored new fields (engineering, math, biology, data science, business).

History of Programing

How can we write code in binary?!

1950s

- FORTRAN (1957) – Formula Translating System
- LISP (1958) – List Processor
- COBOL (1959) – Common Business Oriented Language

1960s – 1970s

- BASIC (1964) - Beginners' All-purpose Symbolic Instruction Code
- Pascal (1970)
- C (1972)

History of Programing

How many languages do you recognize?

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

History of Programing

How many languages do you recognize?

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

Introduction to Programming

Different Language – Same Purpose

- Human languages may *look* and *sound* different, but they all have the same purpose. **What is it?**

- All computer languages might *look* and are *structured* differently, but they all the same purpose. **What is it?**

Introduction to Programming

“Hello, World!”

- The traditional first programming code to write when learning a new language. (First presented in the handbook for the C programming language in 1978.)

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);
   }
}
```

Introduction to Programming

"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
```

Introduction to Programming

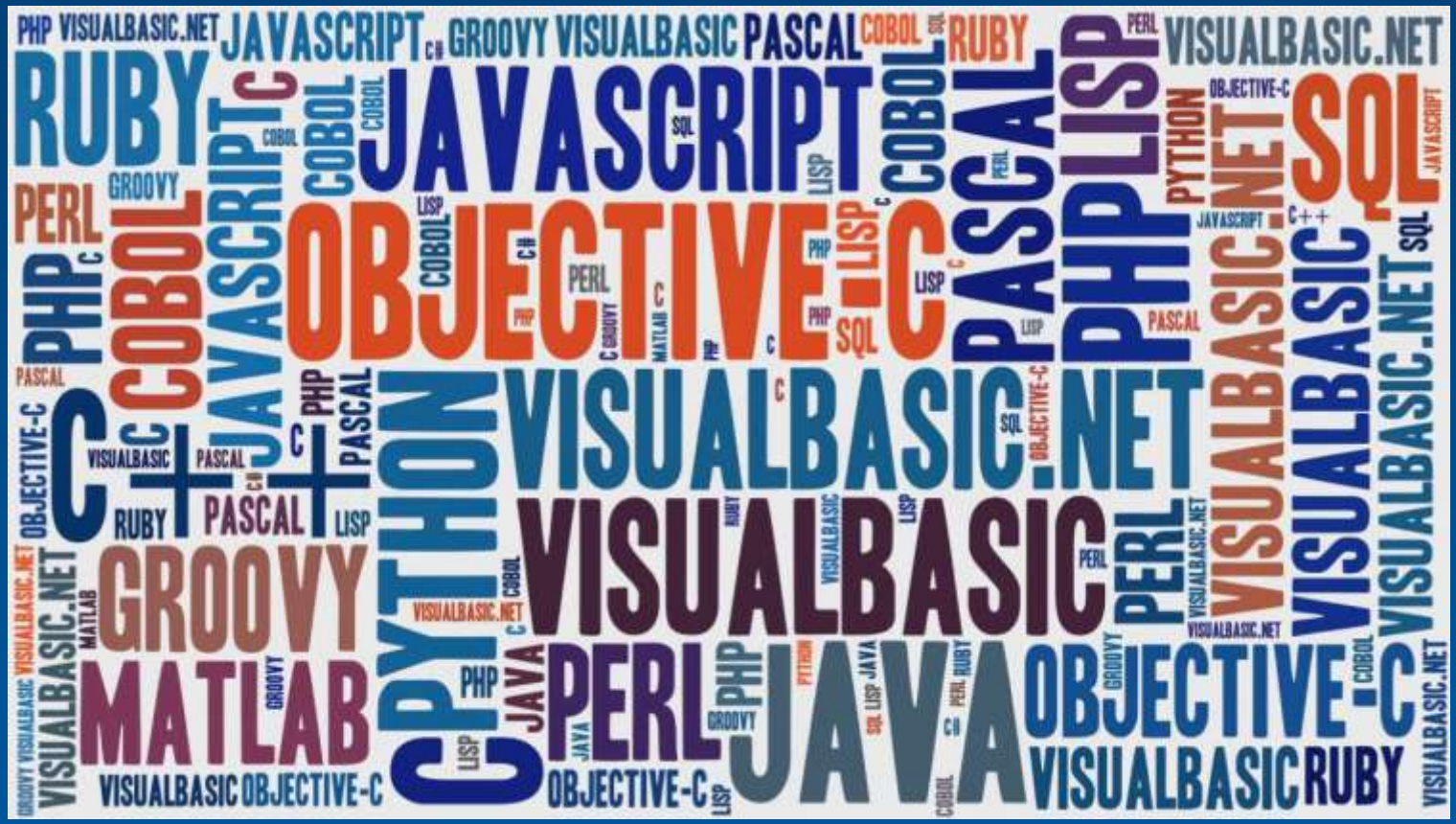
“Hello, World!”

JavaScript

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

Python

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



Prepared. For Life.®

SCOUTS | BSA

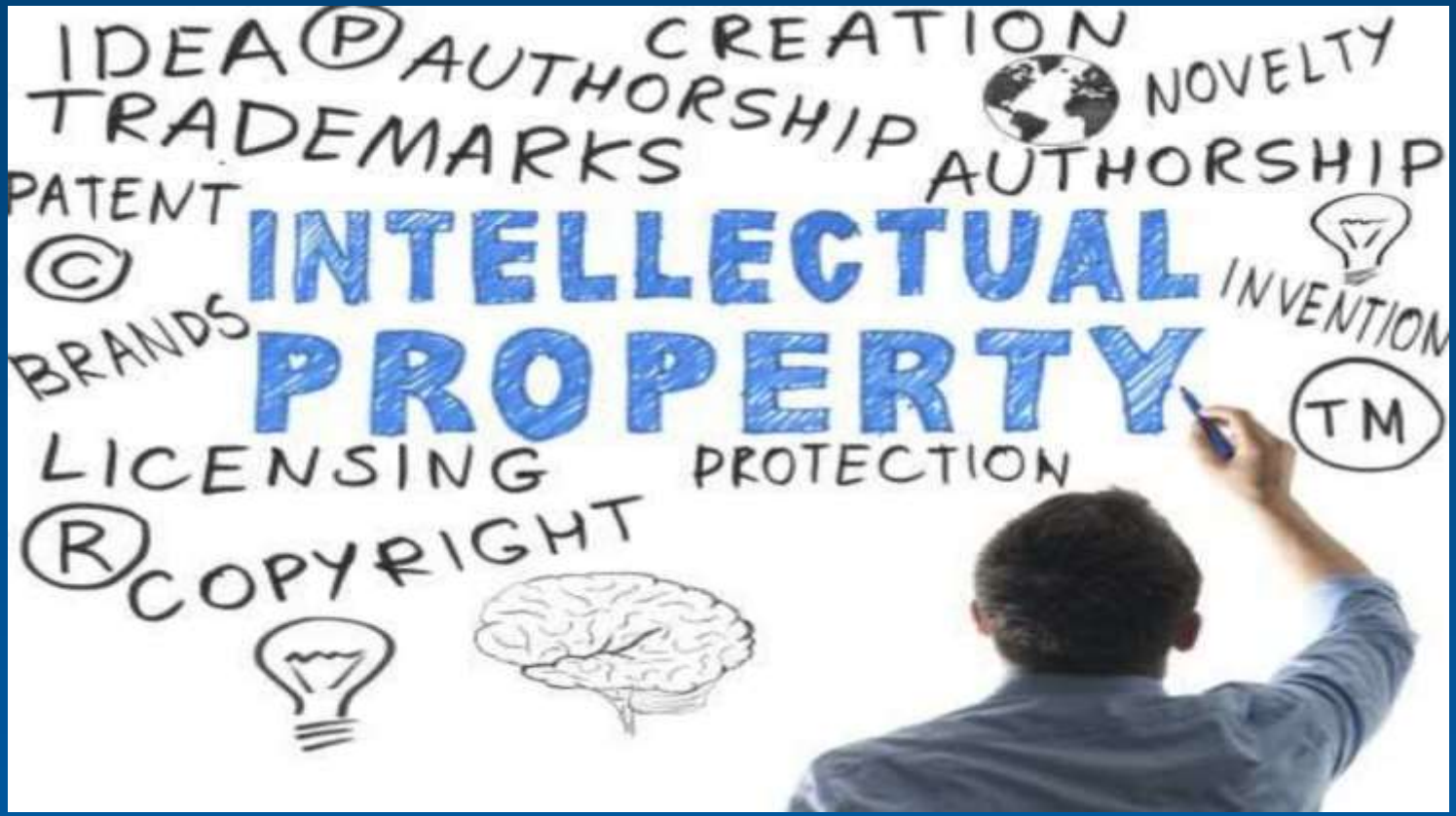
Programming and Devices

Code lives inside so many devices that you interact with every day.

- Mobile phones
- Tablets
- DVD / Blu-ray players
- Game consoles
- Alarm clock
- Smart TVs
- Doorbells

- Cars
- Smart watches
- Fitness trackers
- GPS
- Coffee pots
- Microwaves
- Wi-Fi routers

- **What else? Where else?**


IDEA
AUTHORSHIP
CREATION
NOVELTY
TRADEMARKS
AUTHORSHIP
PATENT
INTELLECTUAL
INVENTION
BRANDS
PROPERTY
TM
LICENSING
PROTECTION
COPYRIGHT

Intellectual Property

Copyright

- Protects expressions of an **idea**.
- Automatically available the **moment** an idea is created and physically stored.
- Protects from others making **copies** of text, images… and *code*!

Intellectual Property

Patent

- Protects innovations and ideas that are **demonstrated** to be useful:
  - Processes and methods
  - Machines and manufactured items
  - “Compositions of matter” (like, a mathematical algorithm)
- Not automatically awarded. Patents must be applied for with the US Patent and Trademark Office --- and world-wide patent offices!
- A lengthy and expensive process: typically requires patent attorneys or agents (and printed paper and money).

Intellectual Property

Trademark

- Protects words, phrases, symbols, or sounds that **identify** and distinguishes the source of a product of service.
  - What are trademarks you are familiar with?
- A more involved process and cost compared to copyrights.
- May need to defend your trademark from other similar uses.

Intellectual Property

Trade Secrets

- Protects **commercially valuable** information.
- The owner of the “secret idea” must take strong measures to maintain secrecy.
- No protections against the use of the “idea” (or code) if it is developed independently or if the information is acquired legitimately.

Intellectual Property

How code can be distributed via licensing:

- **OPEN SOURCE**
  - A community of developers.
  - FREE with support from sponsors or the community.
  - Source code is available to the public, modifiable, and may be redistributed.
- **FREEWARE**
  - FREE! – the developer choose free distribution.
  - But, the source code is not available to others.
- **SHAREWARE**
  - FREE! For a limited trial period.
  - Developer owns the rights, and source code is not available.
  - No collaborative community.

Careers in Programming

Many, many, many, many, many jobs are available today, tomorrow, and after you graduate in computer science and programming!

• Computer Scientist • Mobile App Dev • Application Dev • UI / UX Engineer • Software Engineer • Game Engine Dev • Gameplay Dev • IOS Developer • Robotics Engineer	• Database Engineer • Hardware Engineer • Computer Engineer • Sysadmin • White Hat Hacker • Frontend Web Dev • Backend Web Dev • Full-stack Dev • Embedded Software Engineer	• Systems Analyst • Software QA • Business Intelligence Analyst • Network Admin • Data Engineer • Machine Learning Engineer • Solutions Architect

Introduction to Programming

Let's code!

- There are so many resources available online.

https://**scoutlife.org/programming** – *many* sample programs for Requirement #5

https://**scratch.mit.edu** – block programming (JavaScript)

https://**colab.research.google.com** – Google Colaboratory (Python)

https://**codepen.io/pen** – front-end web development (HTML/CSS/JavaScript)

https://**vscode.dev** – powerful IDE available in the web browser

https://www.khanacademy.org Computer Programing – learn and write code

https://www.w3schools.com – so many great tutorials

https://www.freecodecamp.org – get ready to be a pro

Continuing to Code

- Complete “assignments” – use workbook as a guide.
- Requirement #5 – code in three languages.
- Schedule Zoom sessions with Mr. Dearing.
- CC: Ms. Dalia Israel, on emails.

Prepared. For Life.®

SCOUTS | BSA

- ...
  - ...
    - ...