A programming language is a **formal language** (vs. natural language) that communicates a set of instructions to a machine – computer for example.

Programs tell the computer what to do (even the operating system is a big, complicated program!), express and implement algorithms.

A set of instructions a machine can understand.
• The first “programmable machines” precede modern computers (remember last week’s talk!).
• The first modern computer languages – early 1950s.
• First programs used machine language – directly understood by the CPU.
• The first compiler – 1952 (Grace M. Hopper).
• Compiler – “translates” programs into machine language (itself a program!)

<table>
<thead>
<tr>
<th>Assembly Language</th>
<th>Machine Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>add $t1, $t2, $t3</td>
<td>04CB: 0000 0100 1100 1011</td>
</tr>
<tr>
<td>addi $t2, $t3, 60</td>
<td>16BC: 0001 0110 1011 1100</td>
</tr>
<tr>
<td>and $t3, $t1, $t2</td>
<td>0299: 0000 0010 1001 1001</td>
</tr>
<tr>
<td>andi $t3, $t1, 5</td>
<td>22C5: 0010 0010 1100 0101</td>
</tr>
<tr>
<td>beq $t1, $t2, 4</td>
<td>3444: 0011 0100 0100 0100</td>
</tr>
<tr>
<td>bne $t1, $t2, 4</td>
<td>4444: 0100 0100 0100 0100</td>
</tr>
<tr>
<td>j 0x50</td>
<td>F032: 1111 0000 0011 0010</td>
</tr>
<tr>
<td>lw $t1, 16($s1)</td>
<td>5A50: 0101 1010 0101 0000</td>
</tr>
<tr>
<td>nop</td>
<td>0005: 0000 0000 0000 0101</td>
</tr>
<tr>
<td>nor $t3, $t1, $t2</td>
<td>029E: 0000 0010 1001 1110</td>
</tr>
<tr>
<td>or $t3, $t1, $t2</td>
<td>029A: 0000 0010 1001 1010</td>
</tr>
<tr>
<td>ori $t3, $t1, 10</td>
<td>62CA: 0110 0010 1100 1010</td>
</tr>
<tr>
<td>ssl $t2, $t1, 2</td>
<td>0455: 0000 0100 0101 0101</td>
</tr>
<tr>
<td>srl $t2, $t1, 1</td>
<td>0457: 0000 0100 0101 0111</td>
</tr>
<tr>
<td>sw $t1, 16($t0)</td>
<td>7050: 0111 0000 0101 0000</td>
</tr>
<tr>
<td>sub $t2, $t1, $t0</td>
<td>0214: 0000 0010 0001 0100</td>
</tr>
</tbody>
</table>
Why Not Machine Language, then?

- See previous slide...
- People are not good at machine language
- Modern (high level languages) are the middle ground between human and computer languages
- They are easier to understand, to debug and to move between different computers

```java
public void bubbleSortA(int[] nums, int size)
{
    for(last = size - 1; last > 0; last = last - 1)
    {
        for(current = 0; current < last; current = current + 1)
        {
            if (nums[current] < nums[current + 1])
            {
                temp = nums[current];
                nums[current] = nums[current + 1];
                nums[current + 1] = temp;
            }
        }
    }
}
```
Why Not Natural Language, then?

- All natural languages are believed to be Turing complete
- That is – they can express anything a Turing machine can
- Most programming languages are Turing complete too.
- It follows – all human languages are equivalent to one another (in the Computability Theory sense)
- But... human languages have context, expressions and ambiguity:
  - Time flies like an arrow
  - I have been chasing my own tail
  - Give me the key
- Not everyone speaks the same language/dialects
- Machine translation between natural languages can be atrocious...
A computer language is also a language.
It has semantics and syntax and a vocabulary.
You already speak at least one language... so you should not be afraid of programming languages.
It’s actually like learning another, much simpler language.
A program is simply a list of unambiguous, clear instructions given to a machine that cannot understand context.
Chocolate Cake Example

Input

Algorithm

Output

Recipe
CHOCOLATE CAKE
4 oz. chocolate
3 eggs
1 cup butter
1 tsp. vanilla
2 cups sugar
1 cup flour

Melt chocolate and butter. Stir sugar into melted chocolate. Stir in eggs and vanilla. Mix in flour.
Spread in greased pan. Bake at 350° for 40 minutes or until inserted fork comes out almost clean. Cool in pan before eating.

Program Code
Declare variables:
chocolate
eggs
mix
butter
vanilla
sugar
flour
mix = melted (chocolate + butter)
mix = mix + (2*eggs)
mix = mix + flour
spread (mix)
While not clean (fork)
bake (mix, 350°)

Mix Ingredients
Spread in Pan
Bake at 350°
Remove from Oven
Let Cool
Eat
Chocolate Cake Example

- Must be unambiguous, solve the problem and terminate.
- One starting point and (one or more) end point(s).
- Input and output (both optional).
- Control flow – we follow the instructions (not necessarily in the order of their appearance) and at any stage we’re at some ”block”.
- Conditional branching – if...then... (else...).
- Loops – repeat some actions for a certain number of times or until some condition is filled.
More Relevant Example, 3-Way Maximum

- **Problem:** Given 3 numbers X, Y, Z – find the largest and print on the screen.
- **Input:** 3 numbers
- **Algorithm:** Calculate the maximum of 3 numbers
- **Output:** The largest number
Input: X, Y, Z

1. Not knowing better, assume X is the largest and denote it ”biggest” (the largest we have so far).
2. Compare X and Y. If Y is larger than X, then now Y is the ”biggest”.
3. Otherwise, do nothing (leave X as ”biggest”).
4. Compare Z and ”biggest”.
5. Return the largest of the two as the maximum.

Try to run it in your head with an actual example (say – 17,8,29) to convince yourselves that it works. Remember that the computer is a machine and that the program should be able to handle any 3 numbers, not just this specific example.
3-Way Maximum as Flowchart

Begin
Read X, Y, Z
BIGGEST = X

Y > BIGGEST
BIGGEST = Y

Z > BIGGEST
BIGGEST = Z

PRINT BIGGEST
End
void max3(int x, int y, int z) {
    int biggest;
    if (x > y) biggest = x;
    else biggest = y;
    if (z > biggest) biggest = z;
    System.out.println(biggest);
}
Algorithm 1 Sort (list $L$ of $N$ numbers)

1: Repeat steps (2-4) $N$ times:
2: $m = \text{Minimum}(L)$
3: print($m$)
4: Remove $m$ from $L$

Algorithm 2 Minimum(list $L$ of $M$ numbers)

1: $tmpmin = \text{first element in } L$
2: for Each of the remaining elements in $L$ do
3: \hspace{1em} $i = \text{Next element in } L$
4: \hspace{1em} if $i < tmpmin$ then
5: \hspace{2em} $tmpmin = i$
6: \hspace{1em} end if
7: end for
8: return $tmpmin$