Range of what you can DO to or with variables.

Assign \( <\text{var}> = \ldots \)

Meth Param. Fun(\( <\text{var}> \)) (say print( ) )

\( <\text{var}>++ \)

Range of how you can specify WHICH variable.

simple, lonely variable.

\( x \)

one array element \( \text{votes}[\text{any expression of an int}] \)
This is how computer scientists think..

\[ x = 1; \quad \text{print}(X); \quad x++; \]

\[ v[\text{pk}] = 1; \quad \text{print}(v[\text{pk}]); \quad v[\text{pk}]++; \]

context: \text{int pk}; \text{ inside a loop body, the value of “pick” pk is a candidate number.}

Outside the loop: \text{int v[ ] = new int[6];}
Every choice in the y dimension can be combined with every choice in the x dimension.

This is how computer scientists think.
Every choice in the y dimension can be combined with every choice in the x dimension. This (really nice!) feature in Java language DESIGN is called ORTHOGONALITY.

This is how computer scientists think..
(really nice!) feature: Where an int variable is located
-----local to a method, array element, instance variable (stay tuned!)-----
is ORTHOGONAL (at right angles) to:
What you can make the computer do
to the variable.
Variables inside objects, also known as fields, can be worked with the same as variables anywhere else!

```java
public class Album {
    private int nPicts;
    private pictArray[];
    public boolean removeLast( )
    {
        if( this.nPicts == 0 ) { return false; }  
        else
        {
            this.nPicts = this.nPicts – 1;
            return true;
        }
    }
    /* other stuff */  
}
```
One Album object has two variables inside it, laid out by the blueprint expressed by the class definition. The variable named nPicts has type int.
Variables inside objects, also known as fields, can be worked with the same as variables anywhere else!

```java
public class Album {
    private int nPicts;
    private pictArray[ ];
    public boolean removeLast( )
    {
        if( this.nPicts == 0 ) { return false; } 
        else
        {
            this.nPicts = this.nPicts – 1;
            return true; }
    }

    /* other stuff */
```
<table>
<thead>
<tr>
<th>x = 1;</th>
<th>print(X);</th>
<th>x++;</th>
</tr>
</thead>
<tbody>
<tr>
<td>v[pk]=1;</td>
<td>print(v[pk]);</td>
<td>v[pk]++;</td>
</tr>
<tr>
<td>this.</td>
<td>print(</td>
<td>this.</td>
</tr>
<tr>
<td>nPicts</td>
<td>this.</td>
<td>nPicts++;</td>
</tr>
<tr>
<td>= 1;</td>
<td>nPicts</td>
<td></td>
</tr>
</tbody>
</table>

Fields (variables inside objects) are variables too!