Variable

That means “a named storage location [located in your computer's RAM] in the computer's memory”

“The data stored in a variable may change while the program is running (hence ... 'variable').

That DATA is called the value.

The value only changes when programmed assignment statements like $X = 202013$; are actually run (“executed”).
Project02: Due Thurs, in 9 days.

- Straight-line code: A for Proj01 and Prof. Papert's kids. E (FAILING) for Proj02 and beyond!

- Design, implement and demonstrate your own parametrized methods you add to\ ArtisticTurtle
  - define (code) method names/bodies in\ ArtisticTurtle.java Code commands this Turtle.
  - call (use) those methods in GraffitiApp's main.

- Same for\ PaintablePicture extends\ Picture
  - You make parametrized methods that change colors of various numbers of\ Pixels in this Picture.
  - Those methods must code LOOPS—You've outgrown SLC.
The gradeArray (Albany, G&E 4.1)

double gradeArray[ ] = {80, 90.5, 88};

This array really is 3 (yes, THREE) different VARIABLES!

Each one variable (of the 3) is LOCATED by TWO things:

(A) the array (ref.) name gradeArray

AND

(B) a number

The truth which Java's design sometimes hides.
Variable: 1\textsuperscript{st} Example

```c
int length;
length = 100; /* 100 is a literal(value). */

\texttt{tRef.forward(length);} \\
\texttt{tRef.turn(72);} /* 72 is another literal */ \\
length = length + 20;

\texttt{tRef.forward(length);} \\
\texttt{tRef.turn(72);} \\
length = length + 20;

\texttt{tRef.forward(length);} \\
\texttt{tRef.turn(72);} \\
length = length + 20;
```
Variables: 2\textsuperscript{nd} example “tickets”

They hold the starting numbers, the intermediate values, and, eventually the answer, for and from Euclid's algorithm (subtractive version).

8. Subtract the OTHER picked number from the number your thumb is on. (The difference should be 0 if they are the same, and positive if they are different.)

9. Erase (or cross out) the number under your thumb and overwrite it with the difference, the number you got from the subtraction you did in step 8.

10. Now, go back to step (4). You will repeat steps (4) – (9) over and over until you get two zeros.
Variables: 3rd Example

Hidden, inside (you can't see them) one Pixel are THREE (3) (separate) int *variables*

- `redIntensity` 0
- `greenIntensity` 255
- `blueIntensity` 0
In Lab04, this Java code:

```java
Color purple = new Color( 175, 0, 175 );
Pixel pixRef;
pixRef = pRef.getPixel( 0, 0 );
pixRef.setColor( purple );
```

commands the computer to CHANGE the values of these 3 variables:

- redIntensity: 0
- greenIntensity: 255
- blueIntensity: 0

The values are changed to:

- redIntensity: 175
- greenIntensity: 0
- blueIntensity: 175
This Pixel has almost max intensity in all 3 colors: It looks white.
This Pixel is LOCATED in location [0] of the 1-dimensional array of Pixels. It's LOCATED at x,y location (0,0) in the 2-dim. Picture (a matrix!)
To change the value in a single Java variable:

(1) LOCATE the variable with its name.

\[ \text{length} = 100; \]

(2) Do the copy ("assign") operation.

To change the values in the 3 color variables of a Pixel:

(1) LOCATE and get a ref. to the Pixel with its x,y location.

\[ \text{pixRef} = \text{pRef.getPixel(5,0)}; \]

\[ \text{pixRef.setColor(purple);} \]

(2) Do (call) the \text{setColor} method.
```java
Code in main
int length;
length = 100;
tRef.forward(length);
length = length + 20;

Code in main
tRef.spiral(100);

Code in ArtisticTurtle.java
public void void spiral(int length)
{
    this.forward(length);
    length = length + 20;
    this.turn(72);
    //Repeat several times.
}
```
Variable: 4\textsuperscript{th} Example

Parameter variable

Code in \texttt{ArtisticTurtle.java}

\begin{verbatim}
public void spiral(int length)
{
    this.forward(length);
    length = length + 20;
    this.turn(72);
    //Repeat several times.
}
\end{verbatim}

method parameter's initial value comes from each method call.

Code in \texttt{main}

\texttt{tRef.spiral(100);}
Why was `tRef` replaced by `this`?

```java
public class ArtisticTurtle extends Turtle {
    public void spiral(int length) {
        this.forward(length);
        length = length + 20;
        this.turn(72);
        //Repeat several times.
    }
}
```

Method parameter's initial value comes from each method call.

Code in `main`

```java
tRef.spiral(100);
```
Why was tRef replaced by this?

Within the main method, an ArtisticTurtle is made. Main's variable tRef stores data to locate that object, so its methods can be called. Those methods code potential behaviors that only make sense for ArtisticTurtles. That code calls other Turtle methods, like turn and forward. In that code, this refers to the particular Turtle that those other methods are called on.

The value of this (within spiral) is a copy of the value of tRef. So forward, etc. is called on the same ArtisticTurtle that was made by main.

Code in main

```
tRef.spiral(100); // tRef is the variable storing the ArtisticTurtle
```

Code in spiral

```
this.forward(length); // this is the particular ArtisticTurtle that was made by main
```
Why was `tRef` replaced by `this`?

The above explanation is complete one I attempted to give with limited jargon.

A good partial way to think about this: *NOWHERE* in `ArtisticTurtle.java` is `tRef` declared. So `tRef` cannot be used.

Meanwhile, `this` has the special meaning of holding the reference (location!) of the object on which the method (where this appears) is called. `this` isn't and must not be declared because “`this`” is a Java keyword. It cannot be the name of a variable.

Code in `main`

```java
  tRef.spiral(100);
```

Code in `spiral`

```java
  this.forward(length);
```
Don't worry about this, but do worry about methods..
The subtlety of what “this” is will be taught several weeks from now..

But for now, the easy and important concepts are what a method is and what are methods good for.

The iClicker question below tests whether you understand the main idea of methods. That is a big step up from straight line programming.

Understanding and using methods is absolutely critical for success in the rest of this course.
iClicker: Can different calls to the same spiral method, to draw different spirals, give different initial values to the param. variable whose name is length?

A) YES! That's a great way to draw many spirals of different sizes!

B) NO. A parameter variable can have its initial value set only once. Therefore, instructions you write once in a method, like straight-line code, can only be run once.
Project02: Due Thurs, in 9 days.

- Straight-line code: A for Proj01 and Prof. Papert's kids. E (FAILING) for Proj02 and beyond!

- Design, implement and demonstrate your own parametrized methods you add to ArtisticTurtle
  
  - define (code) method names/bodies in ArtisticTurtle.java Code commands this Turtle.
  
  - call (use) those methods in GraffitiApp's main.

- Same for PaintablePicture extends Picture
  
  - You make parametrized methods that change colors of various numbers of Pixels in this Picture.
  
  - Those methods must code LOOPS-You've outgrown SLC.
Plan

• Rest of today: ArtisticTurtle methods. (Linked separately from the main web page.)
• Thursday: Turtle moves meet up with LOOPs.
• Thursday: After you finish Lab04: Methods added to PaintablePicture and LOOP to recolor lots of Pixels.