CSI201
Lecture 03-04
Feb 5-7, 2013
I just figured out a better way to tell if I need to move the whiteboards so they don't obscure the view for anybody. I can stand at the screen and see if I can see everybody. If not then somebody's view is obscured and I have to move a whiteboard.

This an improved way to figure something out. Computers figure things out. But somebody has to figure out first how the computer will figure a thing out. There are two levels of figuring: First, a person figures out what to program. Second, after the program is written and when the computer runs it, the computer figures something out by following the programmed steps.

Computer science is about figuring out better ways for computers to figure something out. It's harder and more interesting because, unlike in most human activities, you don't just figure out or accomplish something and you're done.
We need to distribute a handout to everyone in the lecture room.

We (humans) can rely on our ability to figure out some way to achieve a goal when we have a goal. We do that by ourselves, we can figure out strategies. We are different from computers that way because computers cannot figure out their own strategies. They must be programmed ahead of time with exactly what to do.

We also asked you to develop jointly and individually strategies for distributed processing. Distributed processing is what computer scientists call processing for one task that is carried out by many computing entities at the same time.
• Turtle graphics and programming
  – Great intro to **sequential processing**: you see the effects of different orders.
  – Turtle (animal) only moves forward or turns, sketches its path with a pen

• Object oriented programming: a modern way to think
  – inside the computer there are real, concrete, heavy, kind of alive “objects” with potential behaviors
  – The code YOU write will command an object to actually act out a behavior.

• Lab01-02, Proj01: You write a sequence of commands to tell a Turtle object to turn or go forward.
English program of sequential steps:

1. Display a message on the screen: “How many hours did you work?”
2. Allow the user to enter the number of hours worked.
3. Once the user enters a number, store it in memory.
4. Display a message on the screen: “How much do you get paid per hour?”
5. Allow the user to enter an hourly pay rate.
6. Once the user enters a number, store it in memory.
7. Once both the number of hours worked and the hourly pay rate are entered, multiply the two numbers and store the result in memory.
8. Display a message on the screen that shows the amount of money earned. The message must include the result of the calculation performed in Step 7.
Link to video of Papert, his turtles and kids
http://www.youtube.com/watch?v=xMzojQFyMo0

What did he say that you think is WRONG? That's an opinion question. Response from a student: “Long division is useless.” The student told us a couple of examples of using division.

My opinion: Papert said that problem solving strategies like breaking a hard problem up into easier subproblems is something “everybody knows”. I don't thing everybody knows that! (Maybe at MIT they do, but not everywhere.) This kind of strategic problem solving is something we want you to become able to do from taking this course.

Once a problem is broken into easier subproblems, Papert says the solution to each subproblem can be expressed in a separate procedure or sub procedure. The kids explain how they did that for the particular Turtle artwork they programmed.
Rough timeline
“method” used today means the same thing as the words “procedure”, “subprocedure”, “subroutine”, “function”
Lab01-02 and Project 01 facts

```java
public class AProgram {

    public static void main(String[] a) {
        //This comment is IN the BODY of
        // the method named main
    }

    Must be saved in a file named AProgram.java
}
```

Rough meaning, good for now:
A **class** is a container (like a trash can) for a complete program.
facts

- You must write the SEQUENCE of commands that the computer will do INSIDE the BODY of the main method.

- G&E Book Classes must be unzipped AND located by extra class path resource preference you can edit in DrJava.

- Code to make one World and one Turtle in it:

```java
World wRef = new World();
Turtle tRef = new Turtle(wRef);
```
Turtle geometry math fact

To make a regular polygon:

- Repeat for the right number of times, the two steps in the given sequence:
  - go forward some distance,
  - turn some angle.

  Always go forward the SAME distance

  Always turn the SAME angle.

- The angle is related to the number of sides
  (you'll recall or figure that out in Lab02.)
Let's demonstrate that with LIVE CODING!

- Course goal: Become able to do yourself, figuring out each step, fixing each oversight, the kind of thinking, typing, programming, we show you. I'll try to think aloud as I program!
  - Practice smaller problems in lab and MyProgLab.
  - Practice and perfect bigger problems in Projects.
  - Demonstrate smaller problems in EXAMS!
  - Do and talk about a programming problem AT A WHITEBOARD during a JOB INTERVIEW!!
    - (well, maybe in 3-4 semesters!)
Make a regular pentagon
Math fact: 5 sides, the turn angle is 72 degree.

Math observation: 5*72 equals 360
360 degrees is a full circle!
public class Demo
{
    public static void main(String[] a)
    {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef);

        tRef.forward( 100 );
        tRef.turn(72);

        tRef.forward( 100 );
        tRef.turn(72);

        tRef.forward( 100 );
        tRef.turn(72);

        tRef.forward( 100 );
        tRef.turn(72);

        tRef.forward( 100 );
        tRef.turn(72);
    }
}
Now make a pentagon-angled broken line spiral

- Each turn still 72 degrees.
- Make the distance forward increase for each side.
- We will write some badly written code to code the sequence of steps.
  - Each different distance is coded by a separate literal (number).
- Start with distance of 20 and increase it by 4 each time: 20, 24, 28, 32, 36, 40, 44, 48, etc.
- Copy, paste and edit is great for programming in such a dumb way!
Really really TERRIBLE solution:

```java
public class Demo {
    public static void main(String[] a) {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef);

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(120); // People figuring out these numbers
        tRef.turn(72);

        tRef.forward(140); // have a hard time getting it right
        tRef.turn(72);

        tRef.forward(180); // This is a bad, tedious, boring,
        tRef.turn(72);

        tRef.forward(200); // ERROR-prone, inflexible
        tRef.forward(72);

        tRef.forward(220); // style of programming
        tRef.turn(72); // DON'T DO IT!!
    }
}
```
It is a much better idea to use a VARIABLE instead of 100, 120, 140, 160, etc.

But: The solution teaches the idea of a regular patterned of a sequence of operations, expressed in straight line code, makes the computer do a regularly patterned task.

Many computation tasks have regular patterns: Similar behaviors are performed repeatedly.
**Figure 1-3** Memory bytes and their addresses

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>72</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</tbody>
</table>

Each **frame** represents a location in memory.

**length**
Last lect: given an address, we can READ OR CHANGE the data stored in memory at that address.

On request, we erased potato from location 3 and rewrote potato over it!
That is OK! That is how computer memory does work!
Better solution, still bad (but not TERRIBLE).

```java
public class Demo {
    public static void main(String[] a) {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef);
        int length;
        length = 100;

        tRef.forward(length);
        length = length + 20;
        tRef.turn(72);

        tRef.forward(length);  //Three lines were
        length = length + 20;   //copied and pasted
        tRef.turn(72);          //without editing
        length = length + 20;   //many times.
        tRef.turn(72);          //No mistakes were made!
        tRef.forward(length);
        length = length + 20;
        tRef.turn(72);
    }
}
```
What each line of code commands the computer to do [Lines 1 and 2]

```java
int length; //"Declare length" which //means set up, in RAM (memory) one //VARIABLE named length for storing, //retrieving, adding to, //and rewriting whole numbers

length = 100; //Makes the computer //store number 100 in that variable.

tRef.forward( length );
length = length + 20;
tRef.turn(72);
```
What each line of code commands the computer to do [Line 3]

tRef.forward( length );

//TWO DIFFERENT THINGS! Read it
//Right ( --> ) to Left ( <= )
//(backwards).

//First: Retrieve the value stored
//in memory (in that variable)
//Second: Command the Turtle to do
//its forward behavior, going
//forward 100 (pixel) steps.

length = length + 20;
tRef.turn(72);
What each line of code commands the computer to do [Line 4]

tRef.turn( 72 );
//This line commands the Turtle
//referred to by tRef
//to actually do its turning behavior
//turning 72 degrees clockwise.

//My purpose for coding Line 4 is to
//prepare the Turtle
//to draw its next line in the
//direction we want.
The 3 lines together

tRef.forward( length );
length = length + 20;
tRef.turn(72);

//Three lines were
//copied and pasted
//many times
//without editing.
//No mistakes were made!
We programmers imagine (1) what a line of code WILL MAKE the computer do; and (2) the PURPOSE we or another programmer had for wanting the computer to do that.

Programming students must learn to use their own mental imaginations that way—OR ELSE: Get C or worse in CS major courses.
"Memory" song from Cats musical http://www.youtube.com/watch?v=4-L6rEm0rnY
Figure 1-3 Memory bytes and their addresses

The represents a location in memory

[Table of memory bytes and addresses]

length
What a variable really is.

Its name is length.

Here is a variable

| length |

120 is the current value.
What a variable really is.

It's called a variable because its value can change.
What a variable really is.

\[ \text{length} = \text{length} + 60; \]

22, 23, 24, 25 are memory addresses of 4 different variables located near each other in computer memory hardware. In Java, by its design for security, numeric addresses cannot be used to refer to variables. But in C and assembly languages (CSI333 in UA) they can. Multiple names for referring to the same thing are called aliases.
C03: live code a pentagon angled spiral with a variable for the distance.
It is a much better idea to use a VARIABLE. According to Gaddis, in the context of programming (not math)

A) A variable is a named storage location in the computer's memory.
B) A variable is something you can solve for in a math problem.
C) A variable is an unknown quantity.
D) A variable is a name for a value (WRONG!!!
According to Gaddis, in the context of programming (not math)

A) A variable is a named storage location in the computer's memory.

B) A variable is something you can solve for in a math problem.

C) A variable is an unknown quantity. Its value is stored in computer memory. It is not unknown.

D) A variable is a name for a value (WRONG!!!)

“Thirtyfive” is the name for a mathematical value, so is XXXV, that value expressed in Roman numerals XXXV, but these are not variables.
Variables (usually) have names.

Gaddis Code 1-1 has programmer defined names hours, payRate, grossPay
They are names of VARIABLES

To make your program have a variable to use, you must code a command to declare it.

In Java declare a variable by writing

<Type of data it will hold>

    <Name you make up> ;

usually on one line (good style)
Declaring & assigning a variable (shown in live coding)

```java
int length; //DECLARE one variable
//GIVING it the name length and
//setting up the MEMORY for use.

length = 100; //ASSIGN: Write
//the data value 100 into the
//variable in MEMORY named by the
//name "length"

//Assigning a value the first time
//is sometimes called defining or
//initializing.
```
Using & Changing a variable (shown in live coding)

tRef.forward ( length );
//USING the value of length to tell
//the Turtle what distance to go
//forward

length = length + 20;
//CHANGE the value stored in MEMORY
//from 100 to 120.
//The computer FIRST retrieves 100,
//SECOND adds 20 to get 120 in its
//calculator, THIRD overwrites
//the old value 100 with 120.
Changing is really 3 steps! (shown in live coding)

Before: 100

length = length + 20;
//CHANGE the value stored in MEMORY
//from 100 to 120.
//The computer FIRST retrieves 100, SECOND adds 20 to get 120 in its calculator, THIRD overwrites the old value 100 with 120.

After: 120
More about Variables

- They are just like the grid boxes pre-printed on ledger paper used by accounting students (or the boxes in computer spreadsheets!)

- A variable can store a number, like the “memory” in a calculator use with the M+, M- keys.

- Not just numbers: Some variables store LOCATIONS of OBJECTS like Turtles
  - A Turtle object is NOT a real little animal. It IS a region of computer memory in which information about the particular one is stored.
  - Java objects are located by a memory address, a number you ordinarily never see.
• Don't worry about the point below yet.

• It explains what's really involved with code like
  – World wRef = new World( );
  – Turtle tRef = new Turtle( wRef );

• Not just numbers: Some variables store LOCATIONS of OBJECTS like Turtles
  – A Turtle object is NOT a real little animal. It IS a region of computer memory in which information about the particular one is stored.
  – Java objects are located by a memory address, a number you ordinarily never see.
I just figured out a better way to tell if I need to move the whiteboards so they don't obscure the view for anybody. I can stand at the screen and see if I can see everybody. If not then somebody's view is obscured and I have to move a whiteboard.

This an improved way to figure something out. Computers figure things out. But somebody has to figure out first how the computer will figure a thing out. There are two levels of figuring: First, a person figures out what to program. Second, after the program is written and when the computer runs it, the computer figures something out by following the programmed steps.

Computer science is about figuring out better ways for computers to figure something out. It's harder and more interesting because, unlike in most human activities, you don't just figure out or accomplish something and you're done.
Warmup remarks...C04

We need to distribute a handout to everyone in the lecture room.

We (humans) can rely on our ability to figure out some way to achieve a goal when we have a goal. We do that by ourselves, we can figure out strategies. We are different from computers that way because computers cannot figure out their own strategies. They must be programmed ahead of time with exactly what to do.

We also asked you to develop jointly and individually strategies for distributed processing. Distributed processing is what computer scientists call processing for one task that is carried out by many computing entities at the same time.
• Turtle graphics and programming
  – Great intro to **sequential processing: you see the effects of different orders.**
  – Turtle (animal) only moves forward or turns, sketches its path with a pen
• Object oriented programming: a modern way to think
  – inside the computer there are real, concrete, heavy, kind of alive “objects” with potential behaviors
  – The code YOU write will command an object to actually act out a behavior.
• Lab01-02, Proj01: You write a sequence of commands to tell a Turtle object to turn or go forward.
Suppose steps 1 and 2 were programmed in the wrong order. First, the computer would wait, with a blank screen, for your client or user to enter a number. The client will not know what to do. But if the user did guess to enter a number, the computer would only then display the prompt “How many hours did you work!”

Your client will justifiably think that your software is no good.
What did he say that you think is WRONG? That's an opinion question. Response from a student: “Long division is useless.” The student told us a couple of examples of using division.

My opinion: Papert said that problem solving strategies like breaking a hard problem up into easier subproblems is something “everybody knows”. I don't thing everybody knows that! (Maybe at MIT they do, but not everywhere.) This kind of strategic problem solving is something we want you to become able to do from taking this course.

Once a problem is broken into easier subproblems, Papert says the solution to each subproblem can be expressed in a separate procedure or sub procedure. The kids explain how they did that for the particular Turtle artwork they programmed.
Rough timeline
“method” used today means the same thing as the words “procedure”, “subprocedure”, “subroutine”, “function”

scan of sketch here..
Lab01-02 and Project 01 facts

```java
public class AProgram {
    public static void main(String[] a) {
        //This comment is IN the BODY of
        // the method named main
    }
}
```

Must be saved in a file named `AProgram.java`

Rough meaning, good for now:
A class is a container (like a trash can) for a complete program.
facts

- You must write the SEQUENCE of commands that the computer will do INSIDE the BODY of the main method.

- G&E Book Classes must be unzipped AND located by extra class path resource preference you can edit in DrJava.

- Code to make one World and one Turtle in it:

  ```java
  World wRef = new World();
  Turtle tRef = new Turtle( wRef );
  ```
Turtle geometry math fact

To make a regular polygon:

- Repeat for the right number of times, the two steps in the given sequence:
  - go forward some distance,
  - turn some angle.
    
    Always go forward the SAME distance
    Always turn the SAME angle.

- The angle is related to the number of sides
  (you'll recall or figure that out in Lab02.)
Let's demonstrate that with LIVE CODING!

- Course goal: Become able to do yourself, figuring out each step, fixing each oversight, the kind of thinking, typing, programming, we show you. I'll try to think aloud as I program!
  - Practice smaller problems in lab and MyProgLab.
  - Practice and perfect bigger problems in Projects.
  - Demonstrate smaller problems in EXAMS!
  - Do and talk about a programming problem AT A WHITEBOARD during a JOB INTERVIEW!!
    - (well, maybe in 3-4 semesters!)
Make a regular pentagon
Math fact: 5 sides, the turn angle is 72 degree.

Math observation: $5 \times 72$ equals 360
360 degrees is a full circle!
public class Demo
{
    public static void main(String[] a)
    {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef);

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(100);
        tRef.turn(72);
    }
}
Now make a pentagon-angled broken line spiral

- Each turn still 72 degrees.
- Make the distance forward increase for each side.
- We will write some badly written code to code the sequence of steps.
  - Each different distance is coded by a separate literal (number).
- Start with distance of 20 and increase it by 4 each time: 20, 24, 28, 32, 36, 40, 44, 48, etc.
- Copy, paste and edit is great for programming in such a dumb way!
Really really TERRIBLE solution:

```java
public class Demo {
    public static void main(String[] a) {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef); //DON'T DO IT!!

        tRef.forward(100);
        tRef.turn(72);

        tRef.forward(120); //People figuring out these numbers
        tRef.turn(72);

        tRef.forward(140); //have a hard time getting it right
        tRef.turn(72);

        tRef.forward(180); //This is a bad, tedious, boring,
        tRef.turn(72);

        tRef.forward(200); //ERROR-prone, inflexible
        tRef.forward(72);

        tRef.forward(220); //style of programming
        tRef.turn(72);    //DON'T DO IT!!
    }
}
```
But: The solution teaches the idea of a regular patterned of a sequence of operations, expressed in straight line code, makes the computer do a regularly patterned task.

Many computation tasks have regular patterns: Similar behaviors are performed repeatedly.

It is a much better idea to use a VARIABLE instead of 100, 120, 140, 160, etc.
Figure 1-3  Memory bytes and their addresses

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
</tbody>
</table>

Each represents a location in memory

length
Last lect: given an address, we can READ OR CHANGE the data stored in memory at that address.

<table>
<thead>
<tr>
<th>Address</th>
<th>potato lacta live</th>
<th>color Horton culture</th>
<th>legend dairy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>potato lacta live</td>
<td>color Horton culture</td>
<td>legend dairy</td>
</tr>
</tbody>
</table>

On request, we erased potato from location 3 and rewrote potato over it!

That is OK! That is how computer memory also works!
Better solution, still bad (but not TERRIBLE).

```java
public class Demo {
    public static void main(String[] a) {
        World wRef = new World();
        Turtle tRef = new Turtle(wRef);
        int length;
        length = 100;

        tRef.forward(length);
        length = length + 20;
        tRef.turn(72);

        tRef.forward(length);
        length = length + 20;
        tRef.turn(72);
        //Three lines were copied and pasted
        length = length + 20;  //copied and pasted
        tRef.turn(72);          //without editing
        //many times.
        //No mistakes were made!
        tRef.forward(length);
        length = length + 20;
        tRef.turn(72);
    }
}
```
int length; //"Declare length" which
//means set up, in RAM (memory) one
//VARIABLE named length for storing,
//retrieving, adding to,
//and rewriting whole numbers

length = 100; //Makes the computer
//store number 100 in that variable.

tRef.forward( length );
length = length + 20;
tRef.turn(72);
What each line of code commands the computer to do [Line 3]
tRef.forward( length );
//TWO DIFFERENT THINGS! Read it
//Right ( → ) to Left ( ← )
//(backwards).
//First: Retrieve the value stored
//in memory (in that variable)
//Second: Command the Turtle to do
//its forward behavior, going
//forward 100 (pixel) steps.

length = length + 20;
tRef.turn(72);
What each line of code commands the computer to do [Line 4]

tRef.turn( 72 );
//This line commands the Turtle
//referred to by tRef
//to actually do its turning behavior
//turning 72 degrees clockwise.

//My purpose for coding Line 4 is to
//prepare the Turtle
//to draw its next line in the
//direction we want.
The 3 lines together

tRef.forward( length );
length = length + 20;
tRef.turn(72);
//Three lines were
//copied and pasted
//many times
//without editing.
//No mistakes were made!
We programmers imagine (1) what a line of code WILL MAKE the computer do; and (2) the PURPOSE we or another programmer had for wanting the computer to do that.

Programming students must learn to use their own mental imaginations that way—OR ELSE: Get C or worse in CS major courses.

Semantics: What a line of code WILL MAKE the computer do.
Logic: the PURPOSE we or an other programmer had for making the computer do that.
Syntax: the rules of grammar for structuring code to make it acceptable to the compiler.

The most effective way to solve programming problems is to image what happens to the data stored in computer memory.
It seems hard to teach people to imagine computing this way.
Paper/whiteboard and pen is very effective for people to greatly extend how much they can imagine in their brains alone.
Memory

"Memory" song from Cats musical http://www.youtube.com/watch?v=4-L6rEm0mY

Whoops. Wrong course.
Figure 1-3  Memory bytes and their addresses

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
</tbody>
</table>

Each represents a location in memory

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
</tbody>
</table>

length
What a variable really is.

Its name is length

Here is a variable

120 is the current value
What a variable really is.

length = length + 60;

Its name is length

Here is a variable 180

180 is the current value

It's called a variable because its value can change.
What a variable really is.

length = length + 60;

<table>
<thead>
<tr>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>a variable</td>
<td>180</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Its name is length

80 is the current value

22, 23, 24 25 are memory addresses of 4 different variables located near each other in computer memory hardware. In Java, by its design for security, numeric addresses cannot be used to refer to variables. But C and assembly languages (CSI333 in UA) they can. Multiple names for referring to the same thing are called aliases.
C03: live code a pentagon angled spiral with a variable for the distance.
It is a much better idea to use a VARIABLE. According to Gaddis, in the context of programming (not math)

A) A variable is a named storage location in the computer's memory.
B) A variable is something you can solve for in a math problem.
C) A variable is an unknown quantity.
D) A variable is a name for a value (WRONG!!!)
According to Gaddis, in the context of programming (not math)

A) A variable is a named storage location in the computer's memory.

B) A variable is something you can solve for in a math problem.

C) A variable is an unknown quantity.
   Its value is stored in computer memory. It is not unknown.

D) A variable is a name for a value (WRONG!!!)
   “Thirtyfive” is the name for a mathematical value, so is XXXV, that value expressed in Roman numerals XXXV, but these are not variables.
Variables (usually) have names.
Gaddis Code 1-1 has programmer defined names hours, payRate, grossPay
They are names of VARIABLES

To make your program have a variable to use, you must code a command to declare it.

In Java declare a variable by writing

<Type of data it will hold> <Name you make up> ;

usually on one line (good style)
Declaring & assigning a variable (shown in live coding)

```java
int length; //DECLARE one variable
//GIVING it the name length and
//setting up the MEMORY for use.

length = 100; //ASSIGN: Write
//the data value 100 into the
//variable in MEMORY named by the
//name “length”

//Assigning a value the first time
//is sometimes called defining or
//initializing.
```
Using & Changing a variable (shown in live coding)

```javascript
letRef.forward ( length );
// USING the value of length to tell
// the Turtle what distance to go
// forward

length = length + 20;
// CHANGE the value stored in MEMORY
// from 100 to 120.
// The computer FIRST retrieves 100,
// SECOND adds 20 to get 120 in its
// calculator, THIRD overwrites
// the old value 100 with 120.
```
Changing is really 3 steps! (shown in live coding)

Before: 100

```c
length = length + 20;
//CHANGE the value stored in MEMORY
//from 100 to 120.
//The computer FIRST retrieves 100,
//SECOND adds 20 to get 120 in its
//calculator, THIRD overwrites
//the old value 100 with 120.
```

After: 120

In assembly language, taught in UA ICSI333, these 3 steps are coded in 3 separate lines of code. Those 3 lines of assembly code get translated into 3 separate machine instructions. (On some machines, more powerful instructions that combine some steps can be used.)
More about Variables

- They are just like the grid boxes pre-printed on ledger paper used by accounting students (or the boxes in computer spreadsheets!)

- A variable can store a number, like the “memory” in a calculator use with the M+, M- keys.

- Not just numbers: Some variables store LOCATIONS of OBJECTS like Turtles
  - A Turtle object is NOT a real little animal. It IS a region of computer memory in which information about the particular one is stored.
  - Java objects are located by a memory address, a number you ordinarily never see.
• Don't worry about the point below yet.
• It explains what's really involved with code like
  − World wRef = new World( );
  − Turtle tRef = new Turtle( wRef );
• Not just numbers: Some variables store LOCATIONS of OBJECTS like Turtles
  − A Turtle object is NOT a real little animal. It IS a region of computer memory in which information about the particular one is stored.
  − Java objects are located by a memory address, a number you ordinarily never see.