ENGINEERING
TEXAS A&M UNIVERSITY

Week 2

Sequential Steps, Variables, Assignment and Data Types

Learning Objectives

ENGINEERING
TEXAS A&M UNIVERSITY

Sequential Steps, Variables, Assignments and Data Types

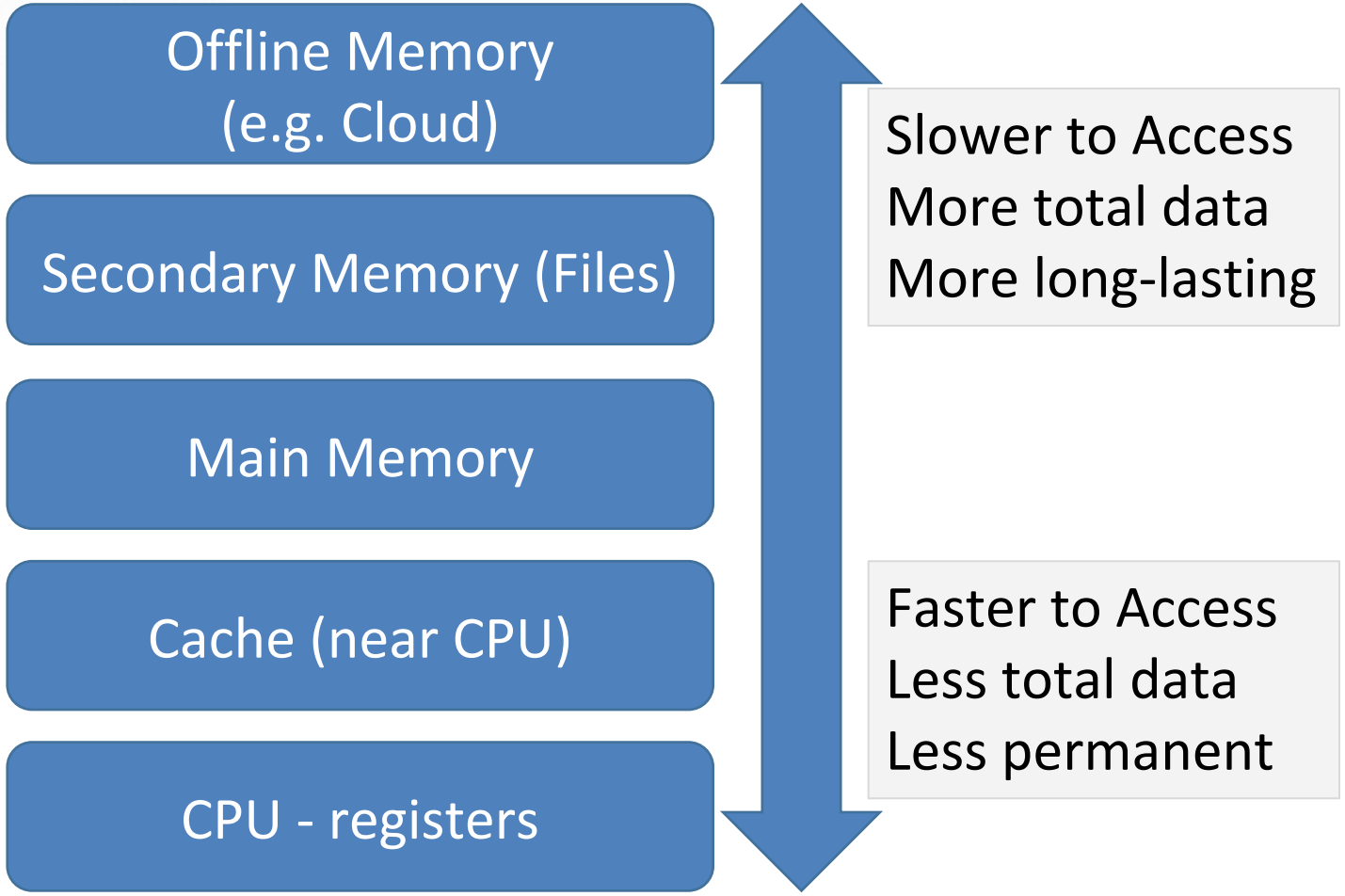
- Utilize variables in Python programming to store data
- Apply Python naming rules to variables
- Identify necessary programming variables from a given problem statement
- Assign or reassign values to variables
- Use the special assignment operators (+=, -=, *=, /=)
- Write a sequence of steps to successfully complete a complex task
- Follow a sequence of steps to understand the complex task that was performed
- Identify the integer, floating-point number, Boolean and string data types in Python
- Know common usages of these data types in programming
- Create, read and use each data type within a Python program
- Use the escape character (\) in Python to define special characters in strings

Computer Memory

- If we can't remember things, we can't actually do very much
- Computers have memory – the ability to remember information

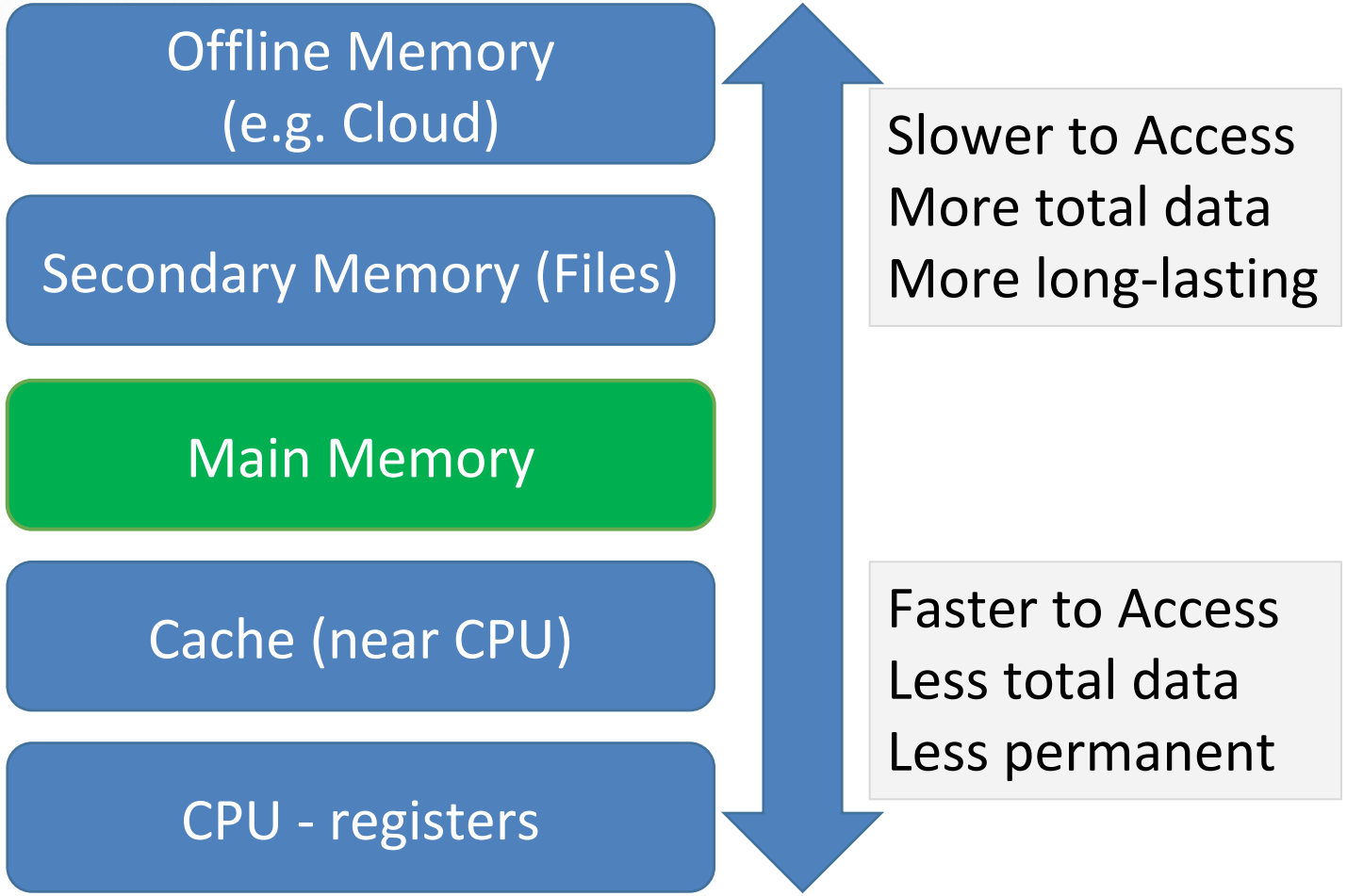
Computer Memory

- Computer memory is organized in a hierarchy
- Higher in the hierarchy, memory is slower to access, but you get more, and it is more permanent



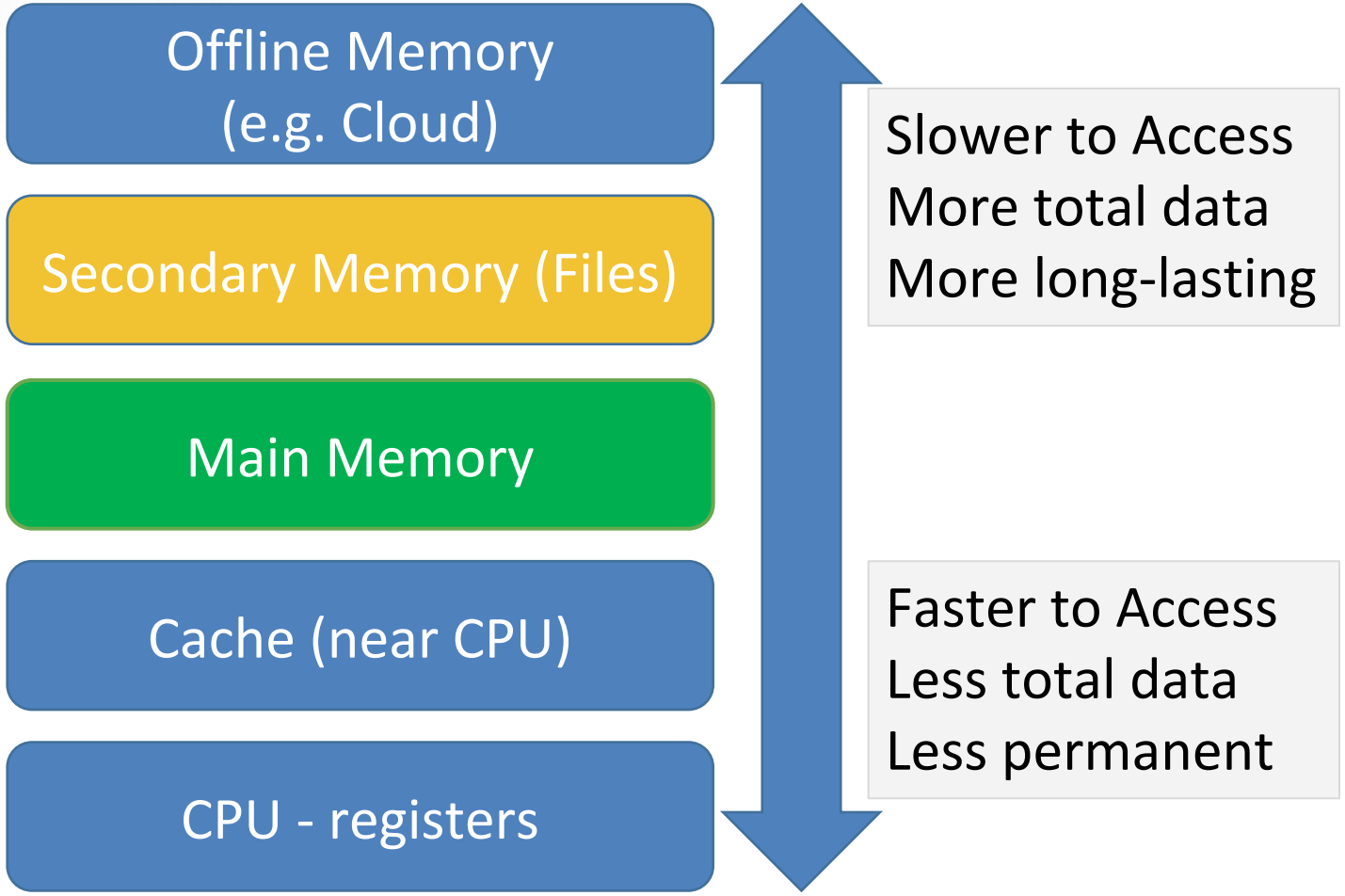
Computer Memory

- When we say “memory”, we’ll be referring to “main memory”



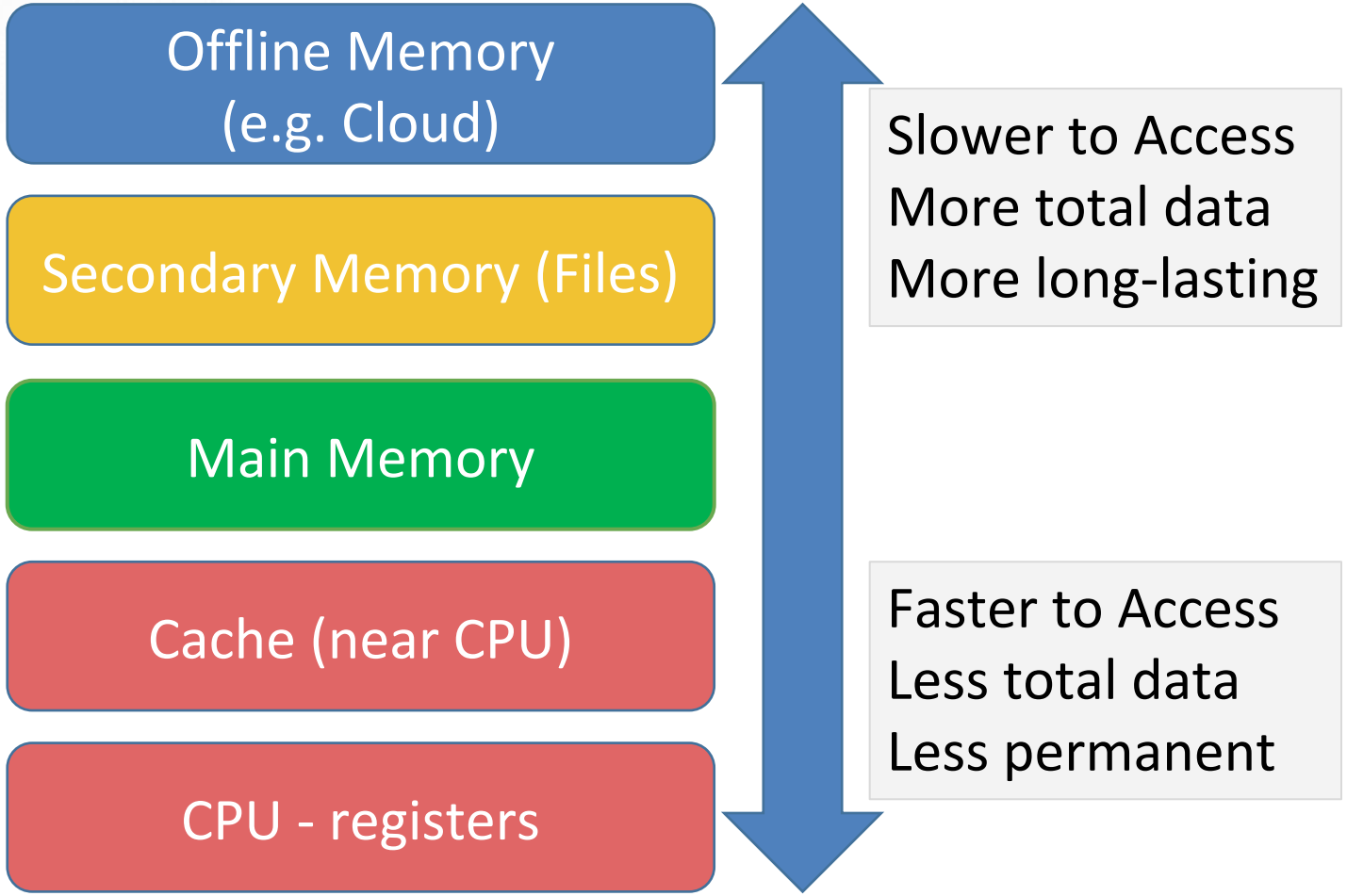
Computer Memory

- Later in the course, we'll talk about handling files



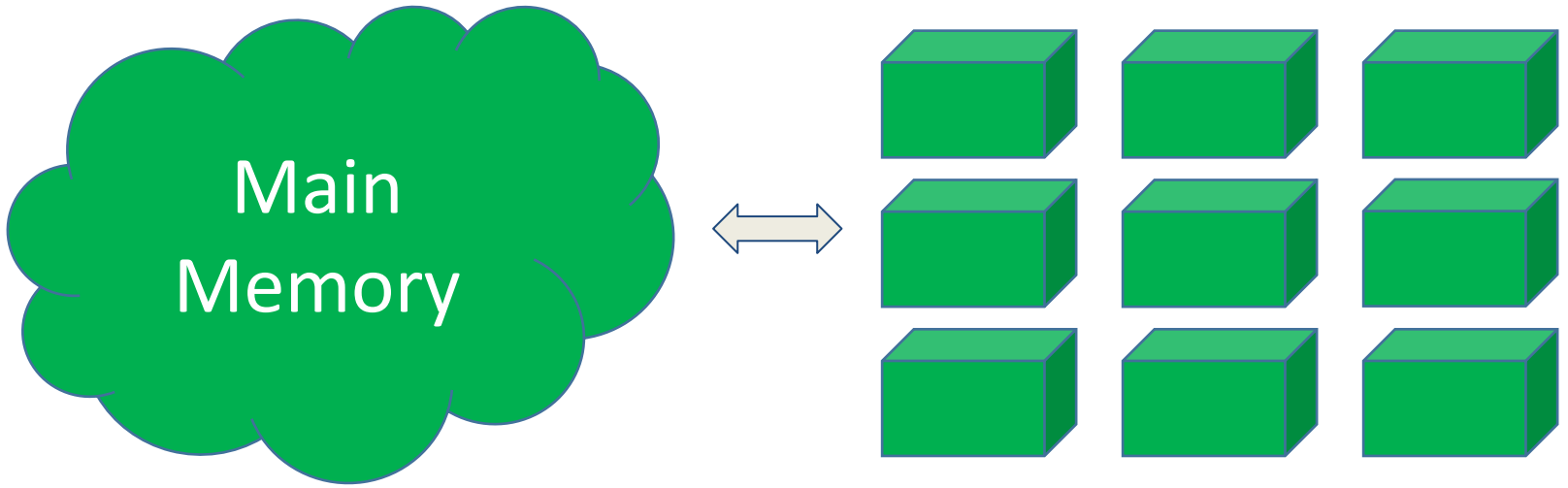
Computer Memory

- We'll ignore the lowest levels of memory in this class.



Main Memory

- Think of main memory as being a bunch of “boxes” that hold information.
- We will call these boxes of memory **variables**



Variables and Names

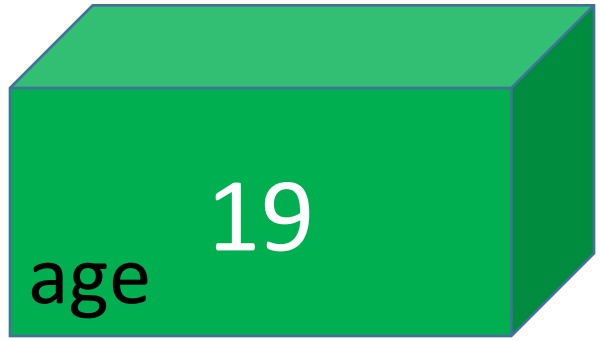
A variable (a box of memory) can hold some unit of information

- e.g. a number, like the number 19

We need a way to refer to a particular variable, to distinguish it from the other boxes of memory. This will be the variable **name**.

- e.g. age

Think of the name as the label for the box



The (Python) rules for naming

The name can **contain** letters (lowercase or uppercase), numbers, and underscore characters.

- But, names can only **start** with a letter or an underscore (_).
- Generally you shouldn't start a variable name with the underscore since it implies certain things about the variable

The name cannot be a reserved keyword

- These are special commands reserved for the language itself. They include words like "for" and "while".
- You will learn these as you learn to program.

Naming Variables and Constants

So you could name all your variables to minimize typing

- e.g. a, b, c, …
- But this would result in confusion later – what do these mean?

A better way of thinking about variable (and other) names you choose in code is to think of communicating with other people

- You are communicating with anyone you are sharing the code with, including anyone who has to take over your code
- But most importantly to you, **you are communicating with the future you** – the you that will have forgotten many of the details of the code you are going back to after doing other things

Are these names valid and good?

`abcde`	
`Age`	
`my_name`	
`My_Name`	
`MyName`	
`2nd_name`	
`name_2nd`	
`Winner!`	
`F0rTheW1n`	
`_density`	
`Gig'Em`	
`Gig Em`	
`Gig-Em`	
`Gig_Em`	

Are these names valid and good?

`abcde`	Valid but not good (meaningless)
`Age`	Valid and good
`my_name`	Valid and good
`My_Name`	Valid and good (and different from previous)
`MyName`	Valid and good (and different from previous)
`2nd_name`	Not valid (starts with a number)
`name_2nd`	Valid, and probably good
`Winner!`	Not valid (contains !)
`F0rTheW1n`	Valid, but not good (digits are confusing)
`_density`	Valid, probably not good (begins with _)
`Gig'Em`	Not valid (contains ‘)
`Gig Em`	Not valid (contains a space)
`Gig-Em`	Not valid (contains a -)
`Gig_Em`	Valid, but probably not good (not clear what it contains)

More about naming

Generally, pick descriptive names

- Volume might be a better choice than V

Not too long, though

- Volume_of_the_sphere is probably too long
- V_sphere, or VolSphere, or Vsphere, etc. might be better names

There are a few conventions people use:

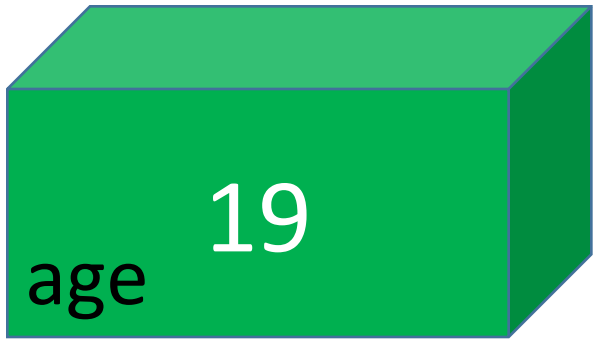
- Constants (that never change) often in ALL_CAPITALS
- Variables i, j, and k often used for counting or indexing
- Variables typically start with lower-case letters

Variable Values

When we see a variable name in a program, that is a placeholder for the value contained in that variable's "box" of memory.

Example: Say the variable "age" holds value 19. What would be the value of age+3?

```
age+3 = 19+3 = 22
```



Assignment

- The process of sticking a value into the box of memory is called assignment.
- Can assign to any variable – new or old
- When a new value is assigned to a variable, the old value disappears – it is replaced by the new one (Often happens *many* times for the same variable.)

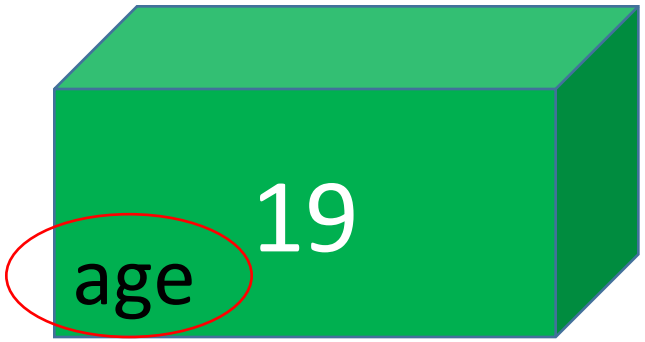
Assignment statements

```
<Variable name> = <Value to assign>
```

Assignment statements

```
<Variable name> = <Value to assign>
```

The left hand side is the variable name for the “box” in memory that will hold the value.

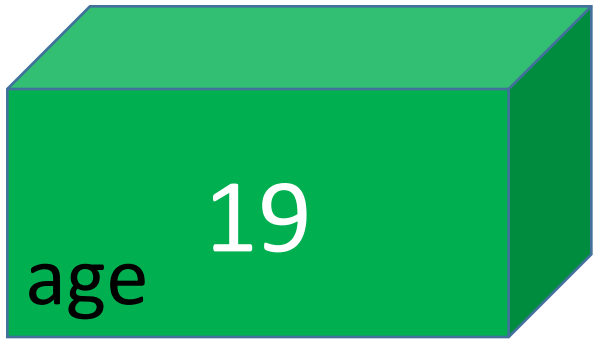


Assignment statements

```
<Variable name> = <Value to assign>
```

The = is the **assignment operator**.

It is NOT the "equals" sign, though we still often read it aloud as "equals". We can read it aloud as "gets" or "is assigned."



Assignment statements

```
<Variable name> = <Value to assign>
```

The right hand side of the assignment operator is the value that will be assigned to the box

Assignment statements

```
<Variable name> = <Value to assign>
```

When this statement is executed:

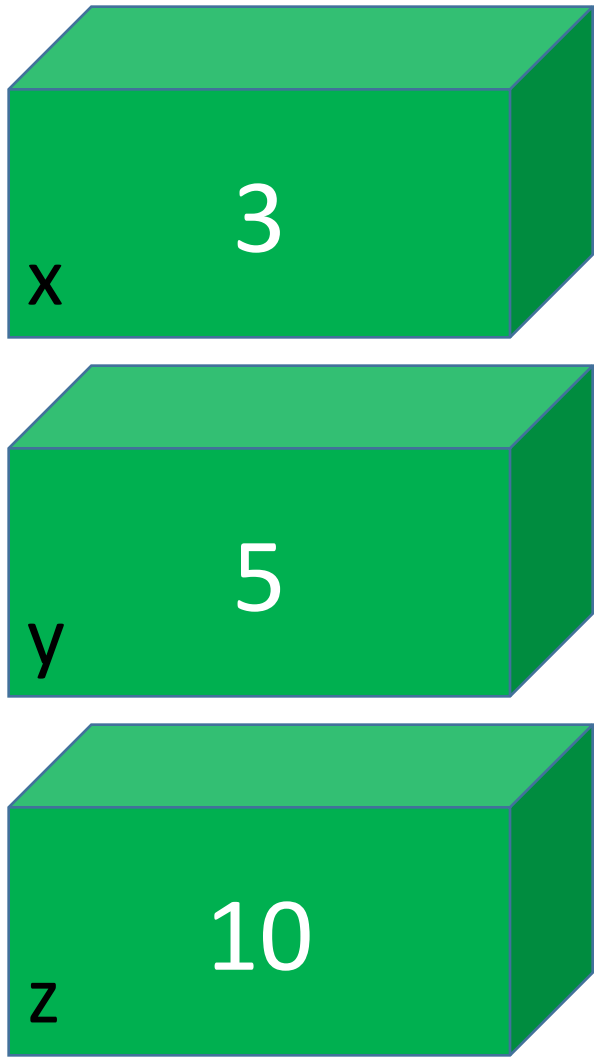
- We **FIRST** evaluate the right hand side
- Then, we assign that value to the left hand side

Example

ENGINEERING
TEXAS A&M UNIVERSITY

What is the result of the executing the statement:

```
z=x+1
```



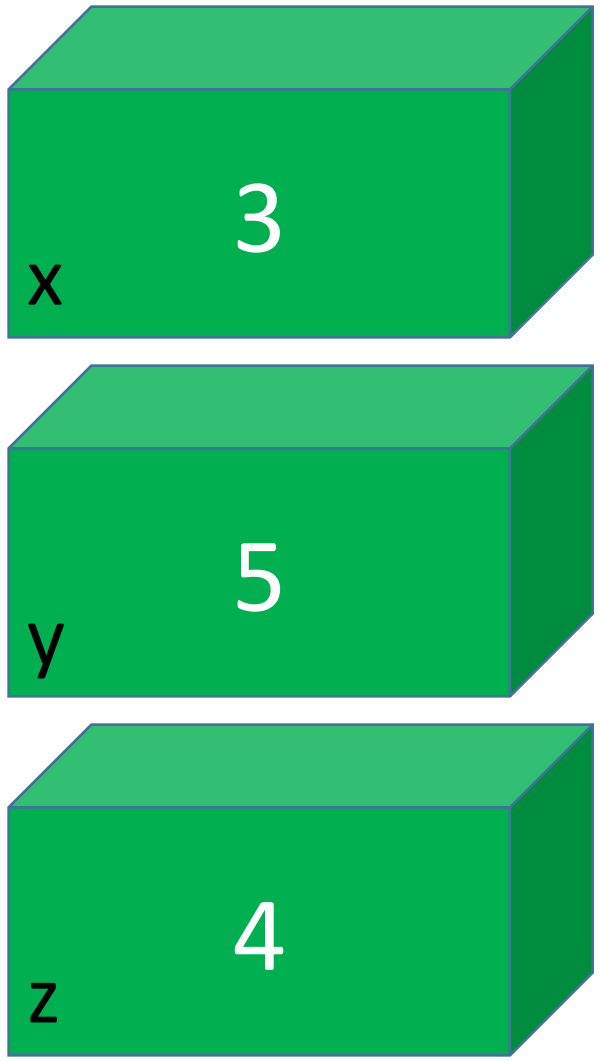
Example

ENGINEERING
TEXAS A&M UNIVERSITY

What is the result of the executing the statement:

```
z=x+1
```

1. We first find what is the value in variable x, which is 3.
2. Then, we add 3+1 to get 4.
3. Then, we assign the value of 4 to variable z, replacing the value previously stored there.

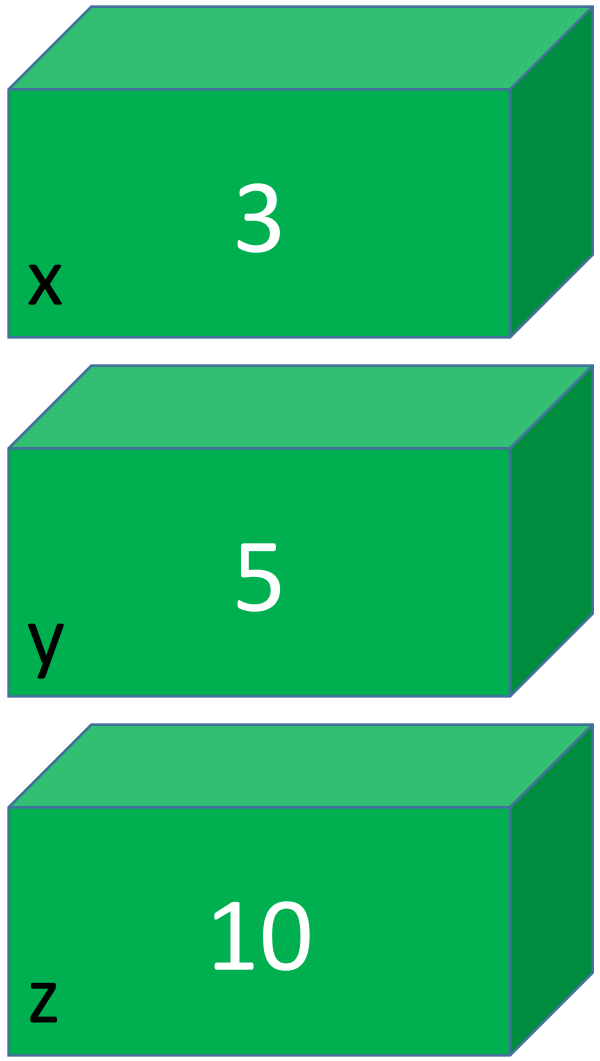


Example

ENGINEERING
TEXAS A&M UNIVERSITY

What is the result of the executing the statement:

```
z=x+2*y
```



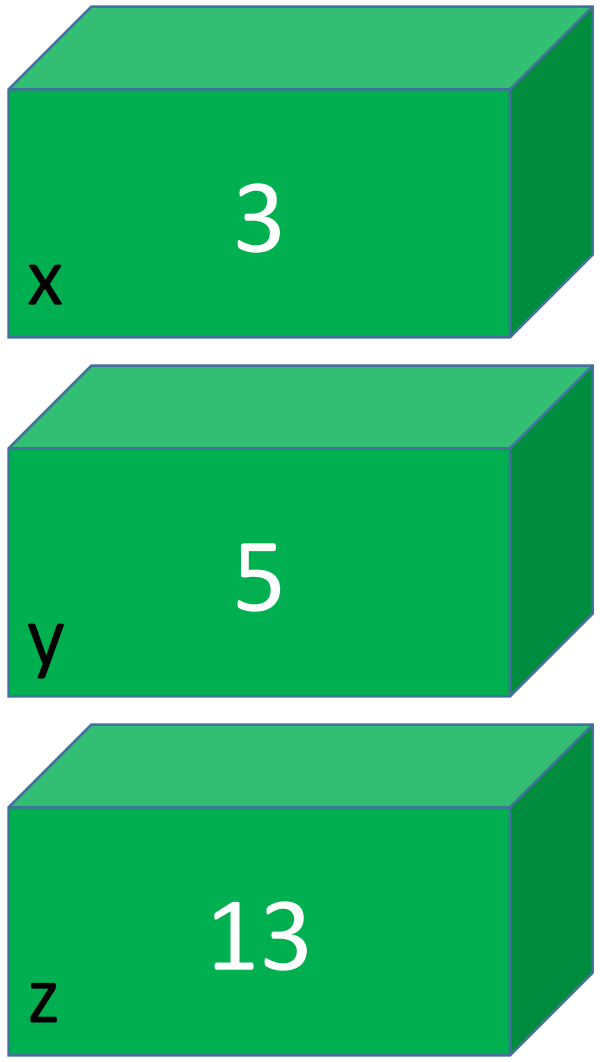
Example

ENGINEERING
TEXAS A&M UNIVERSITY

What is the result of the executing the statement:

```
z=x+2*y
```

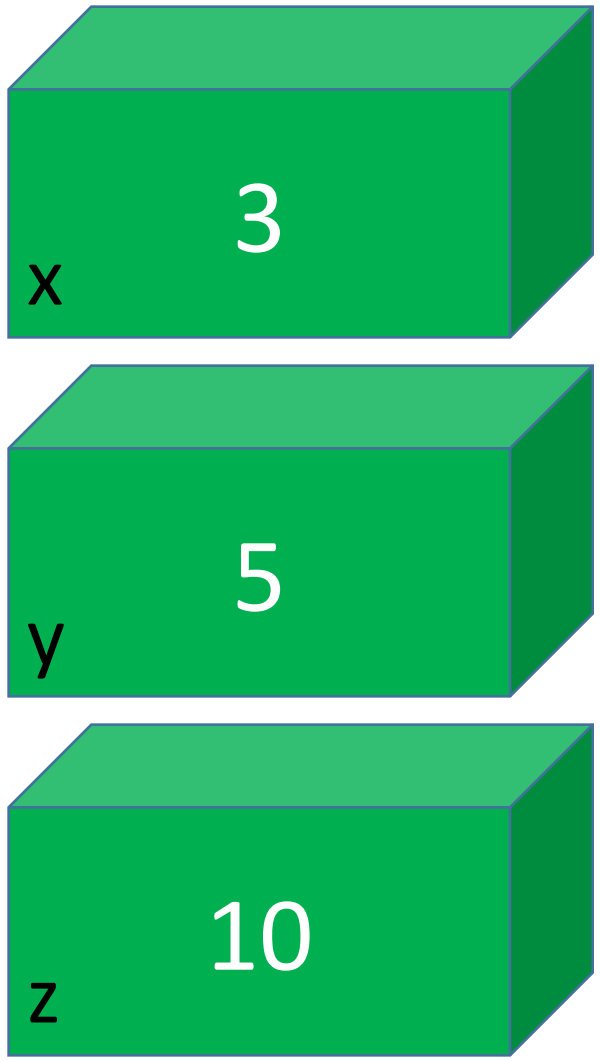
1. We first find what is the value in variables x and y, which are 3 and 5.
2. Then, we multiply 2 times 5 (order of operations!), to get 10.
3. Then, we add 3 to 10, to get 13
4. Then, we assign that value, 13, to variable z, replacing the value previously stored there.



Example

What is the result of the executing the statement:

```
z=z+1
```



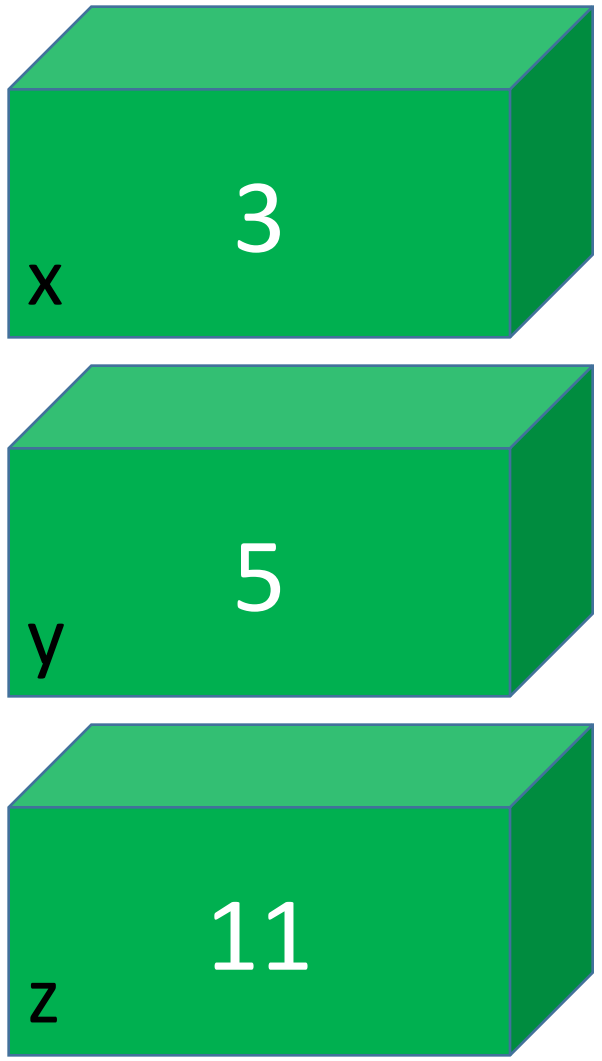
Example

ENGINEERING
TEXAS A&M UNIVERSITY

What is the result of the executing the statement:

```
z=z+1
```

1. We first find what is the value in variable z, which is 10.
2. Then, we add 10+1 to get 11.
3. Then, we assign the value of 11 to variable z, replacing the value previously stored there.



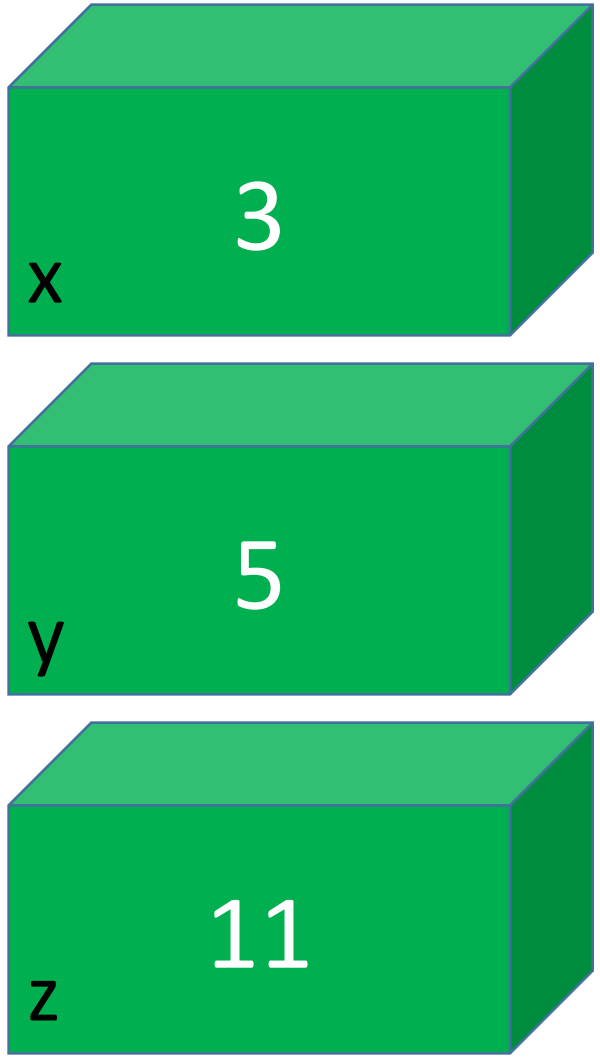
Example

What is the result of the executing the statement:

```
z=z+1
```

Notice the use of the same variable on the right and left side is not a problem!

While this statement would make no sense if the = really meant “equals”, it makes perfect sense since = is actually an assignment operation.



Giving Instructions

If you were asked to give a stranger directions from one place to another, how would you do it?

- Drive straight to the next light
- Turn right
- Drive 1 mile
- Take the entrance ramp to the highway
- Drive until exit 391
- …

This is a **sequence** of steps that the person should follow

Sequential Steps

We are used to giving and receiving instructions as a sequential series of steps.

- This is a very important process – learning how to break up a complex task in various ways is critical to programming, and to dealing with any large project
- One of the main ways this is done is to create a sequence of steps
- You'll get some practice with this in lab

Sequential Steps

When we give instructions to a computer, we'll do the same thing

- Our instructions will be a sequence of steps that we want the computer to do
- We can think of the computer as mindlessly following those instructions, in the order they're specified
- Later, we will see ways of giving instructions that aren't sequential!

Example 1: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

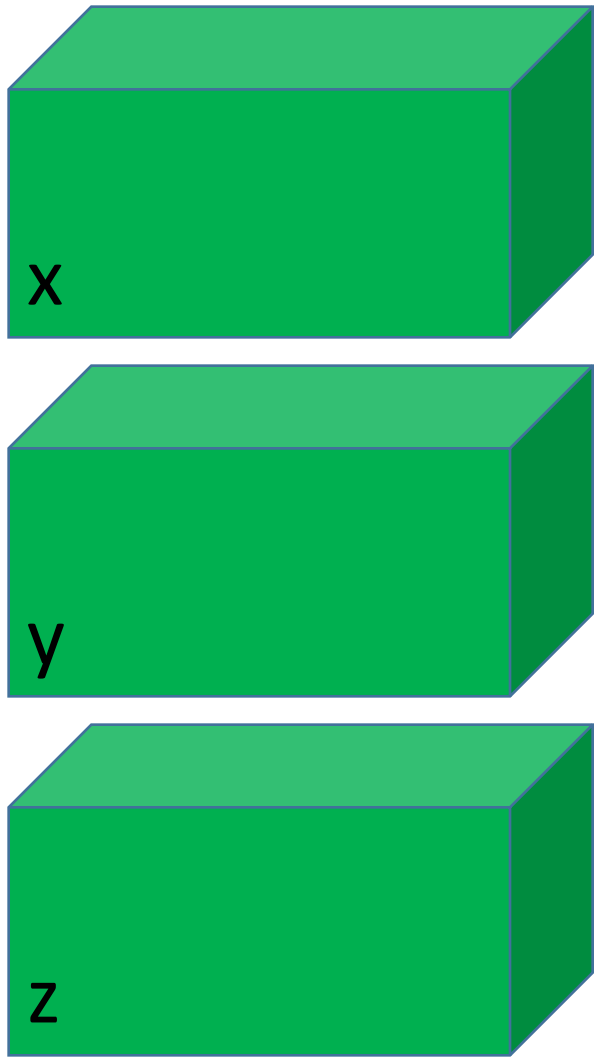
Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
z=3
```

```
z=5
```

```
z=1
```



Example 1: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

Next

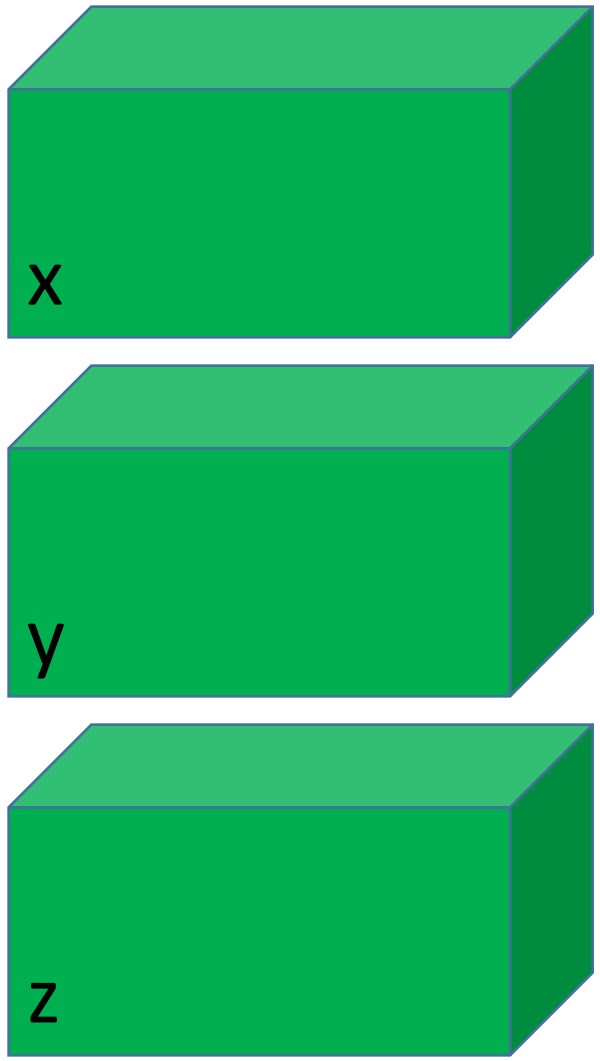
```
z=3
```

```
z=5
```

```
z=1
```

It can be helpful to keep track of the "instruction pointer" – that is, what the next instruction to be executed is. We can think of the program as being "at" these points between statements.

This starts out in front of the first statement.



Example 1: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

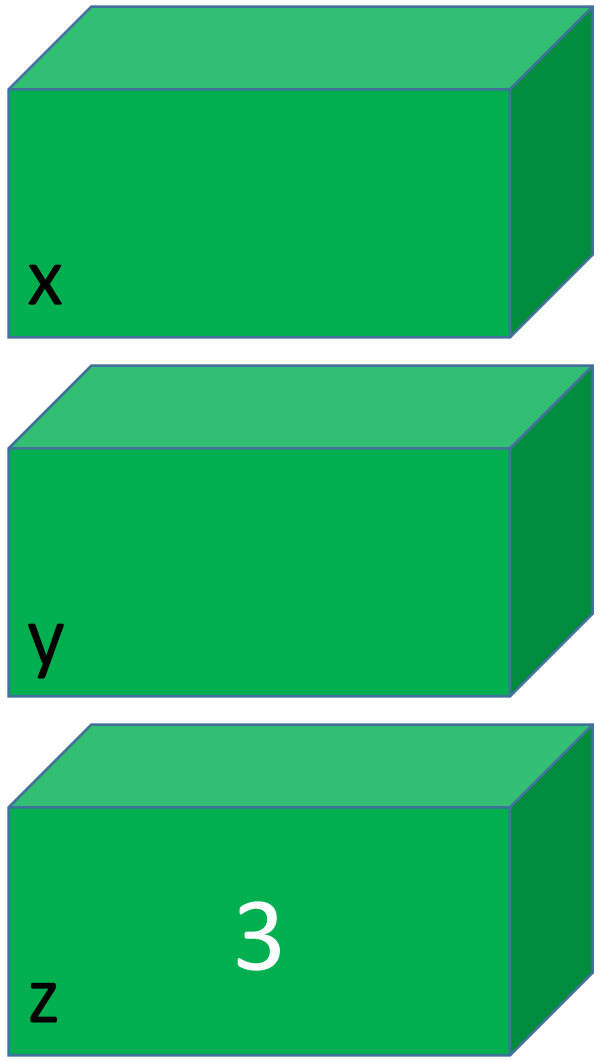
Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
z=3
z=5
z=1
```

Next

After the first statement is executed, the variable z has the value 3 stored inside.



Example 1: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

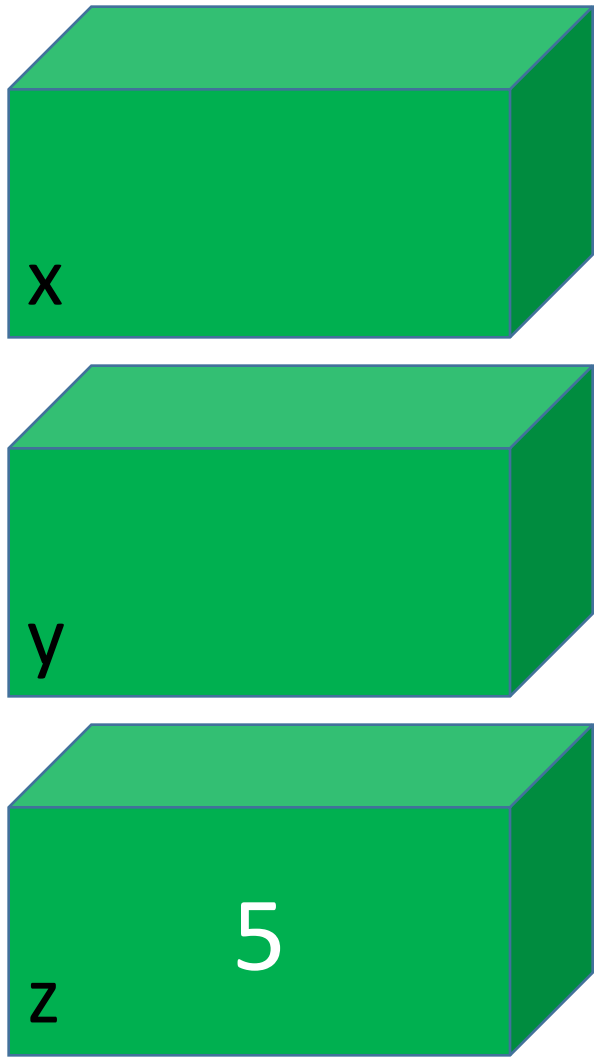
Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
z=3
z=5
z=1
```

Next

After the second statement, z has the value 5.



Example 1: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

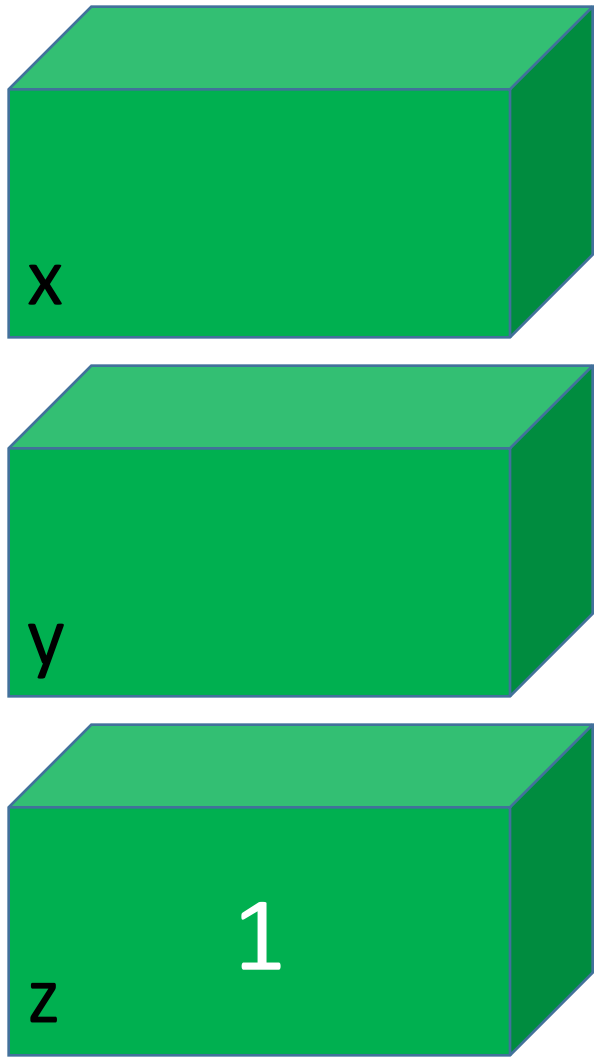
Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
z=3
z=5
z=1
```

Next

After the third statement, z is 1.



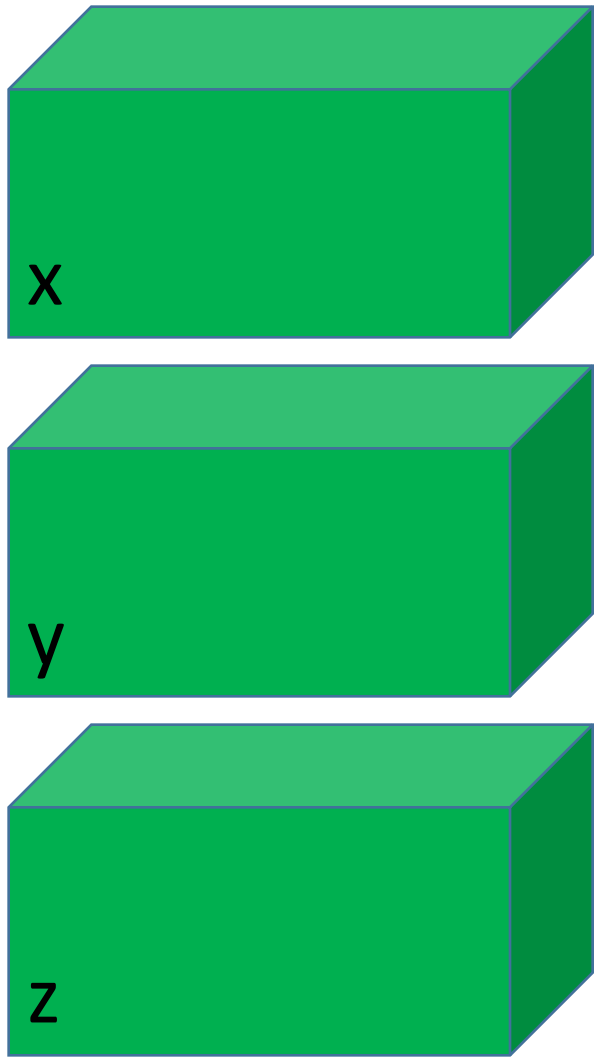
Example 2: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
x=3
y=5
z=x
x=1
```



Example 2: Assignments

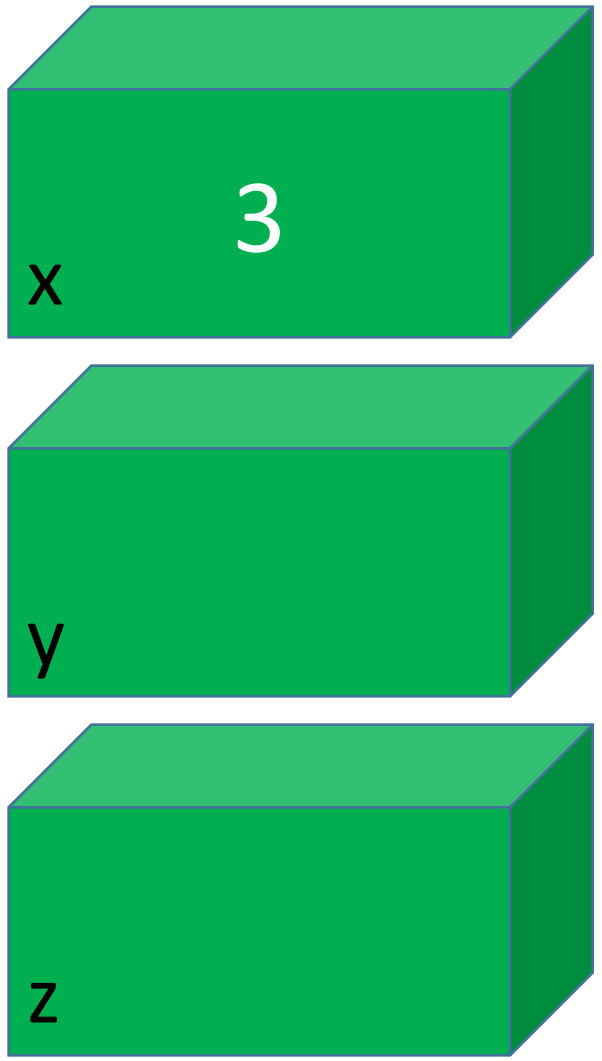
ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

Next

```
x=3
y=5
z=x
x=1
```



Example 2: Assignments

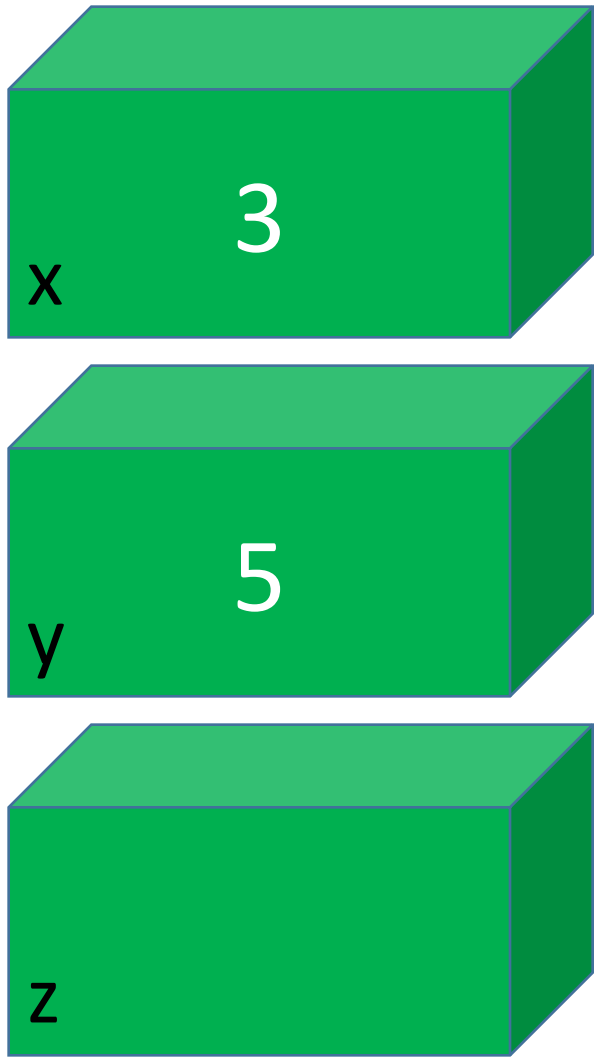
ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
x=3
y=5
z=x
x=1
```

Next



Example 2: Assignments

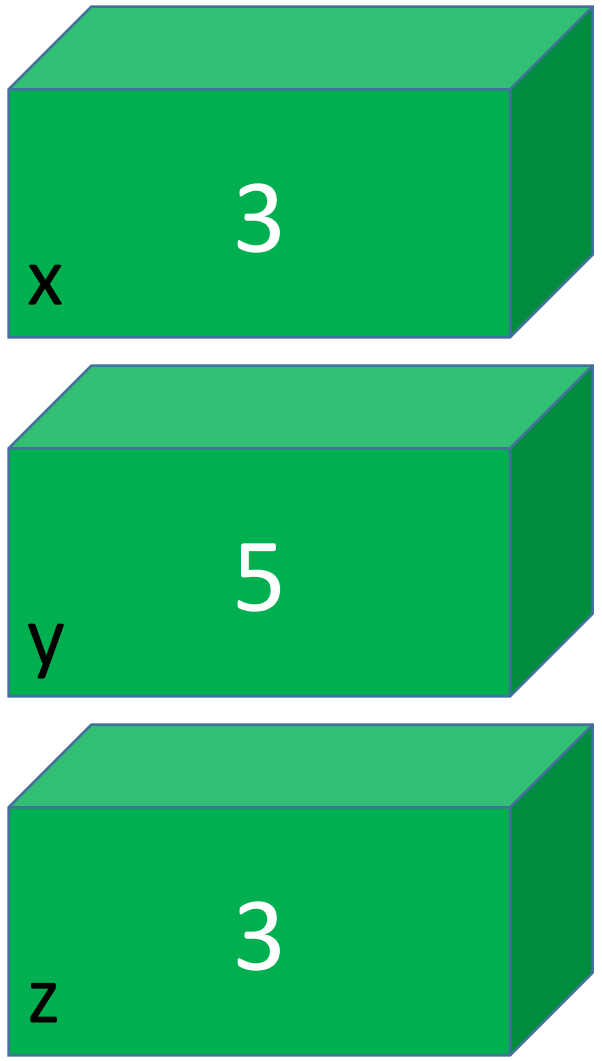
ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

(i.e. what is the value stored in the variable named z after the final statement?)

```
x=3
y=5
z=x
x=1
```

Next



Example 2: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of z at the end?

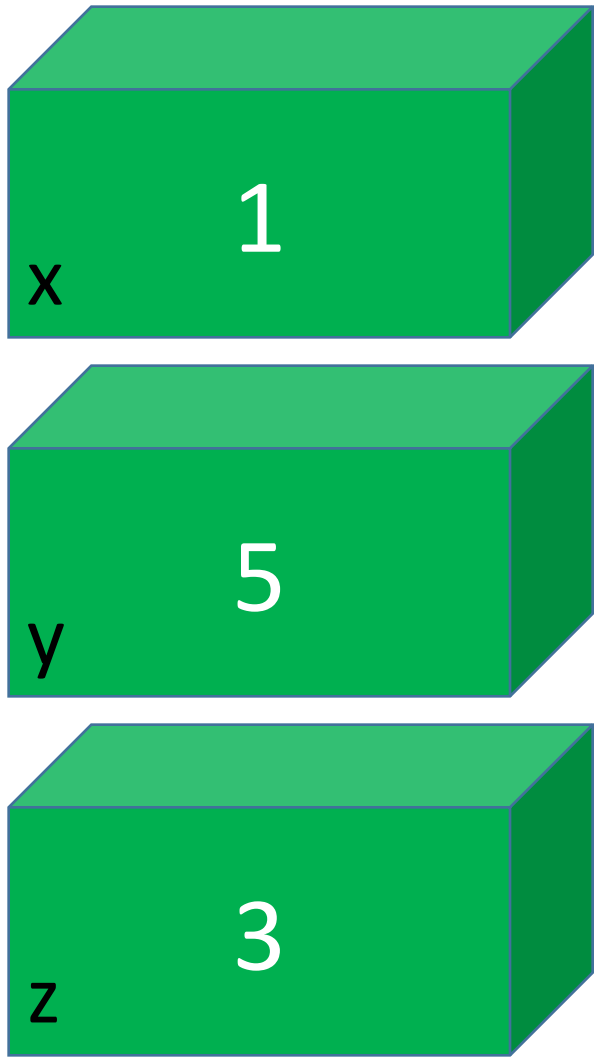
(i.e. what is the value stored in the variable named z after the final statement?)

```
x=3
y=5
z=x
x=1
```

Next

Notice that the value of x changed, but NOT the value of z.

The third statement, assigning z=x was just assigning the current value of x to z, not making z forever equivalent to x.

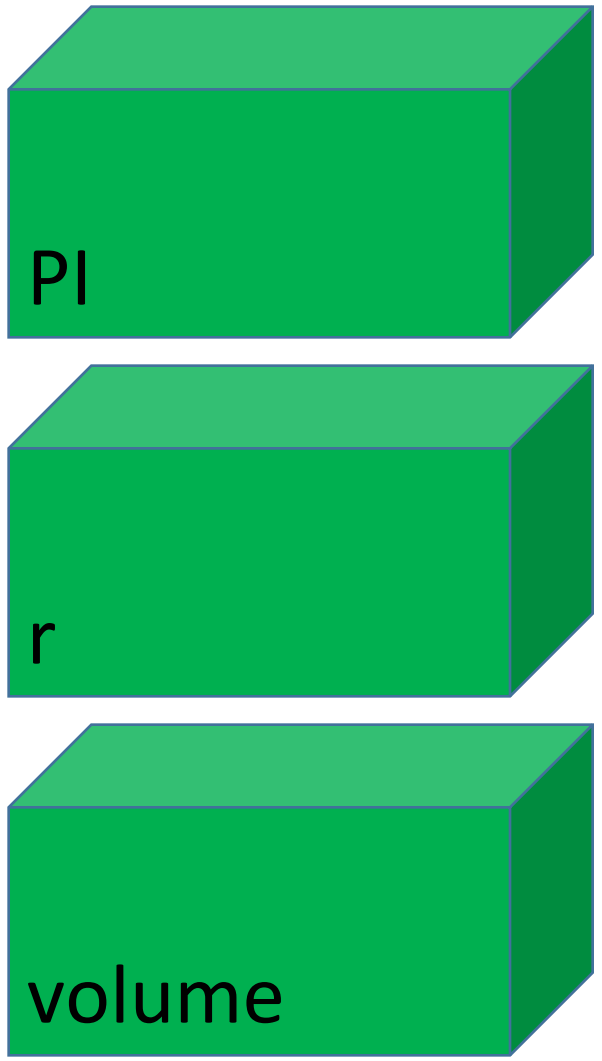


Example 3: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
PI=3.14159
r=5.3
volume=PI*r*r*r*(4/3)
```



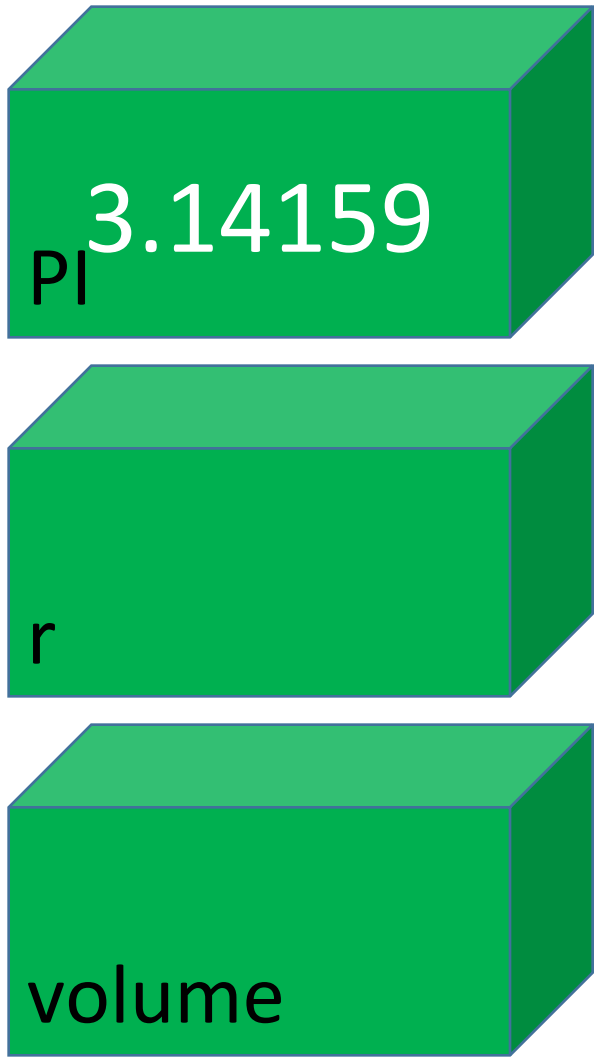
Example 3: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

Next

```
PI=3.14159
r=5.3
volume=PI*r*r*r*(4/3)
```



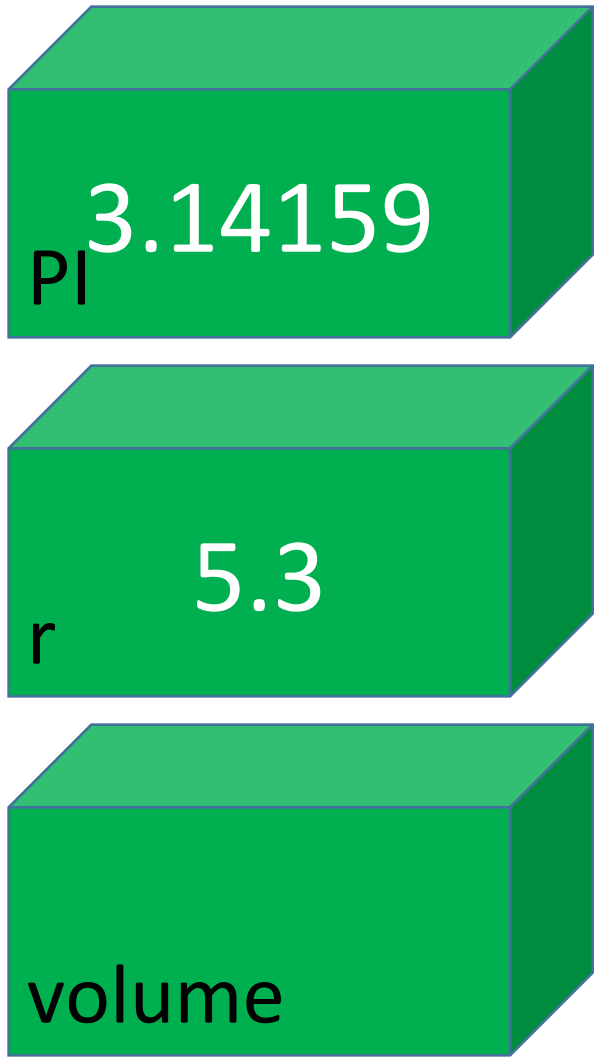
Example 3: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
PI=3.14159
r=5.3
volume=PI*r*r*r*(4/3)
```

Next



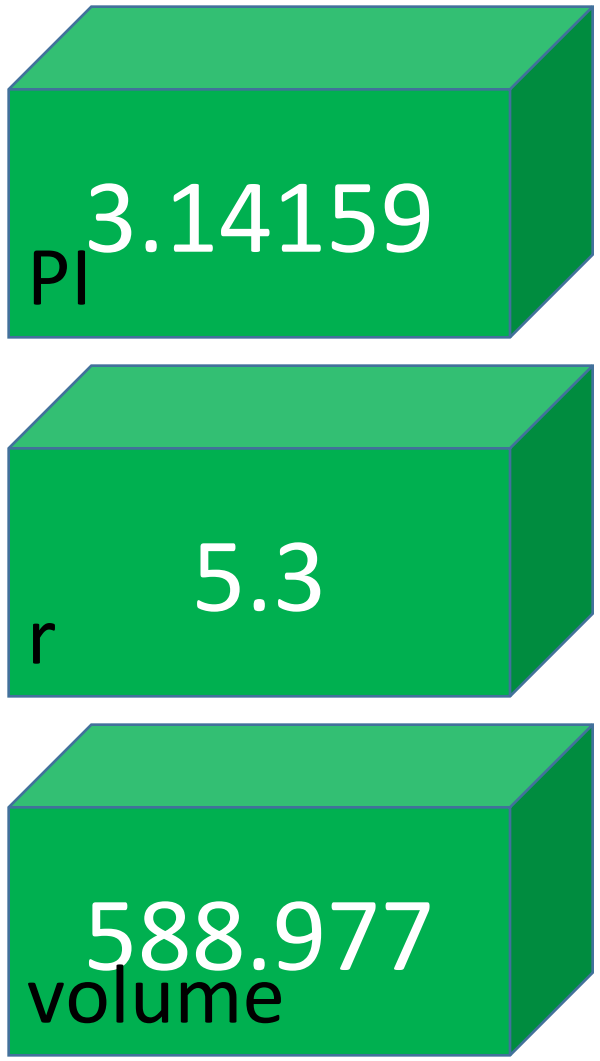
Example 3: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
PI=3.14159
r=5.3
volume=PI*r*r*r*(4/3)
```

Next

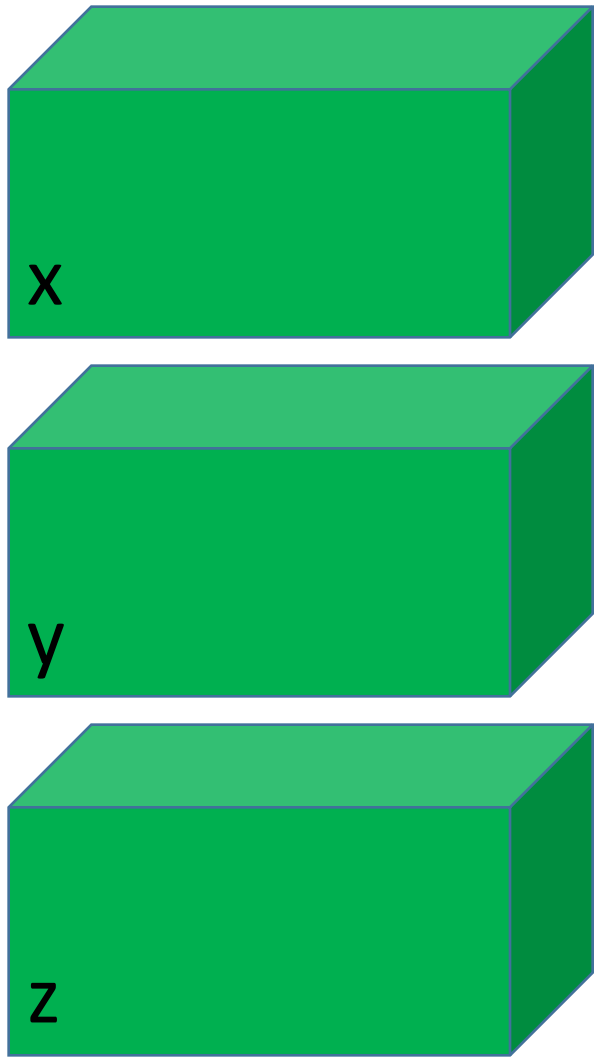


Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```



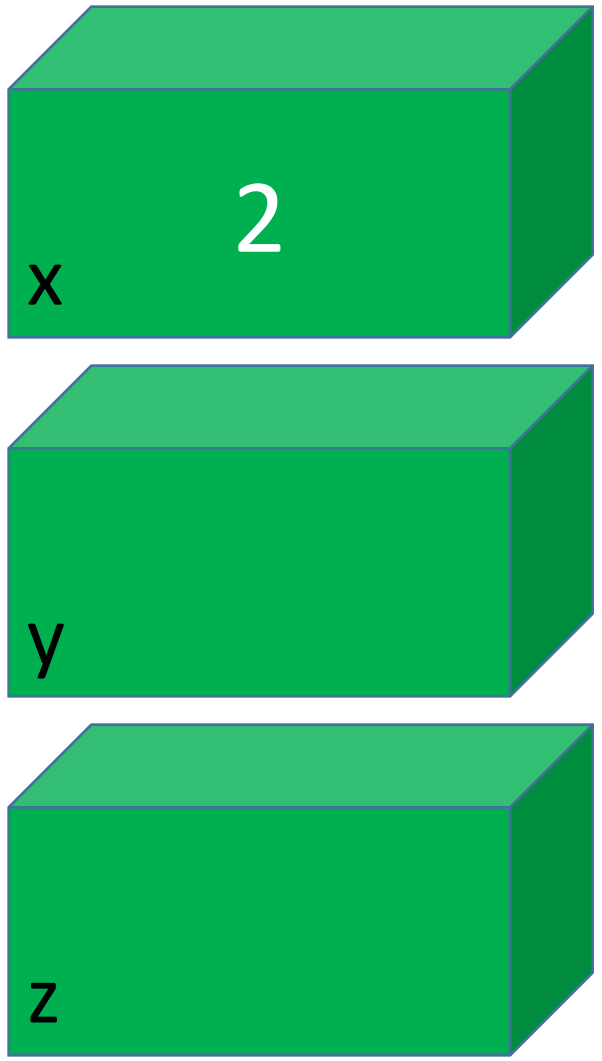
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



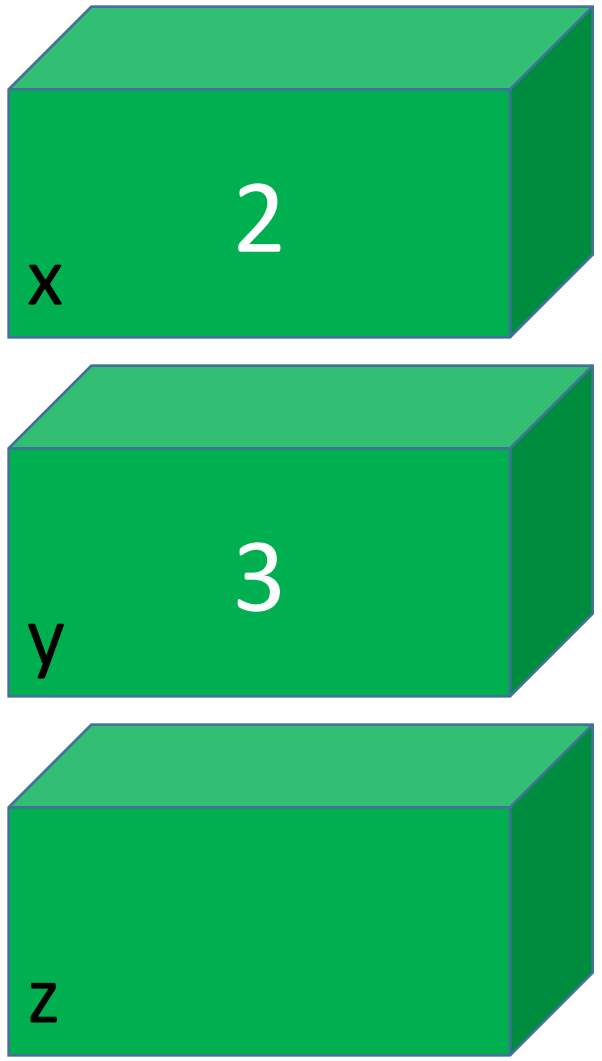
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



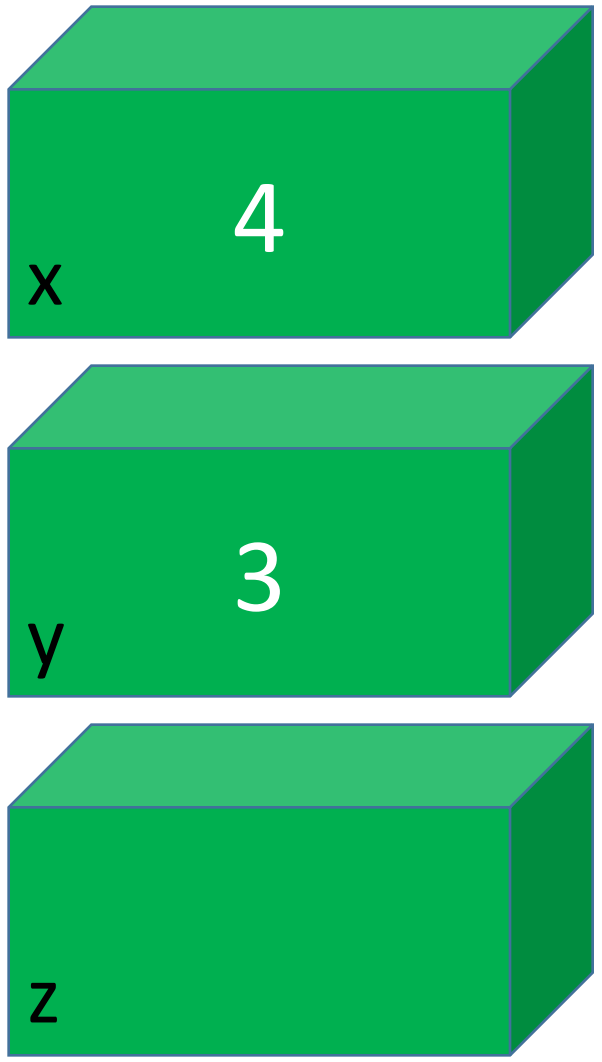
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



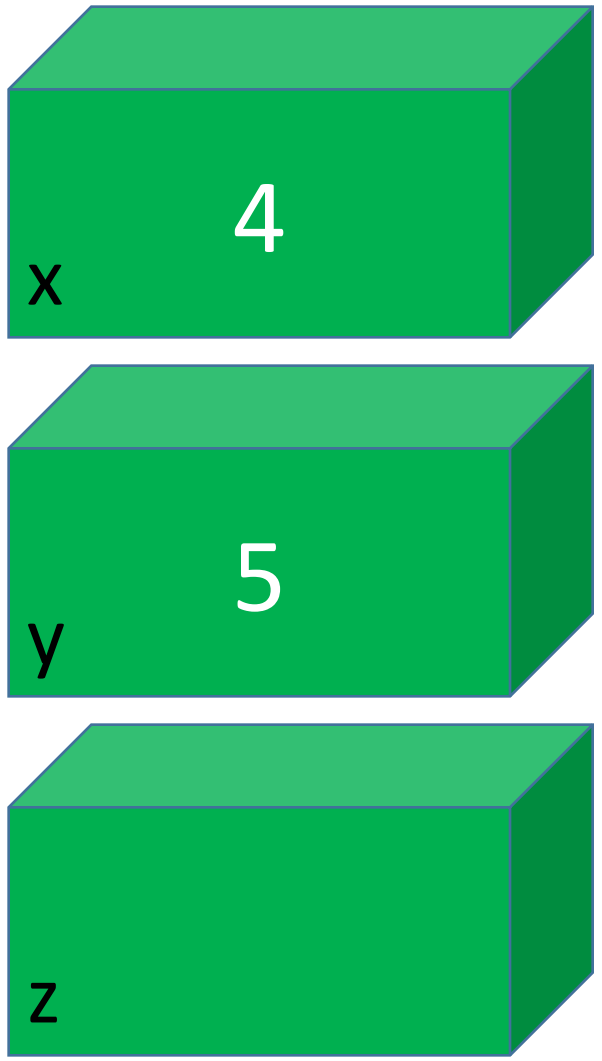
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



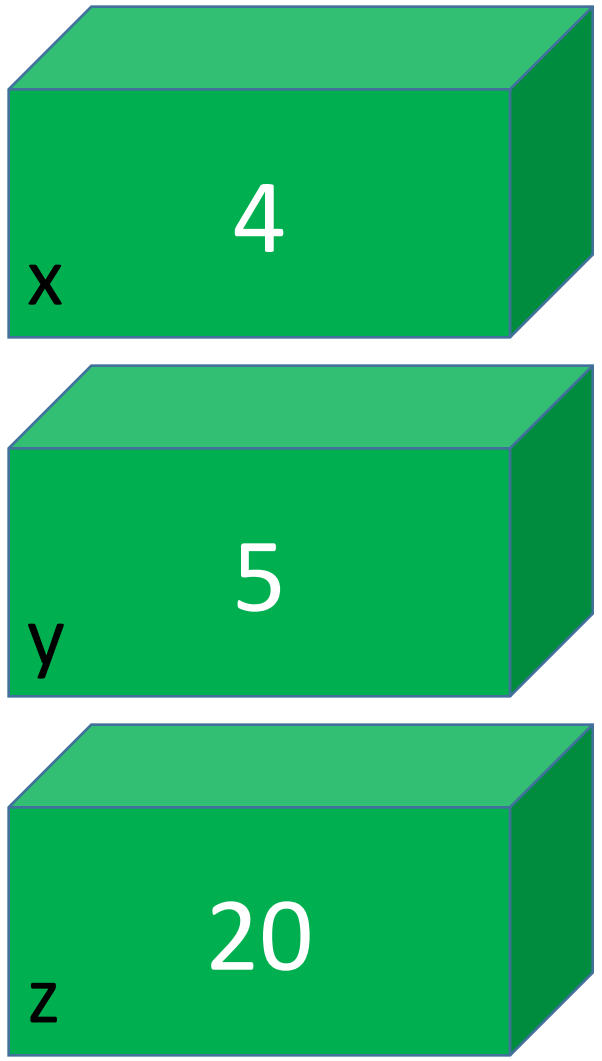
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



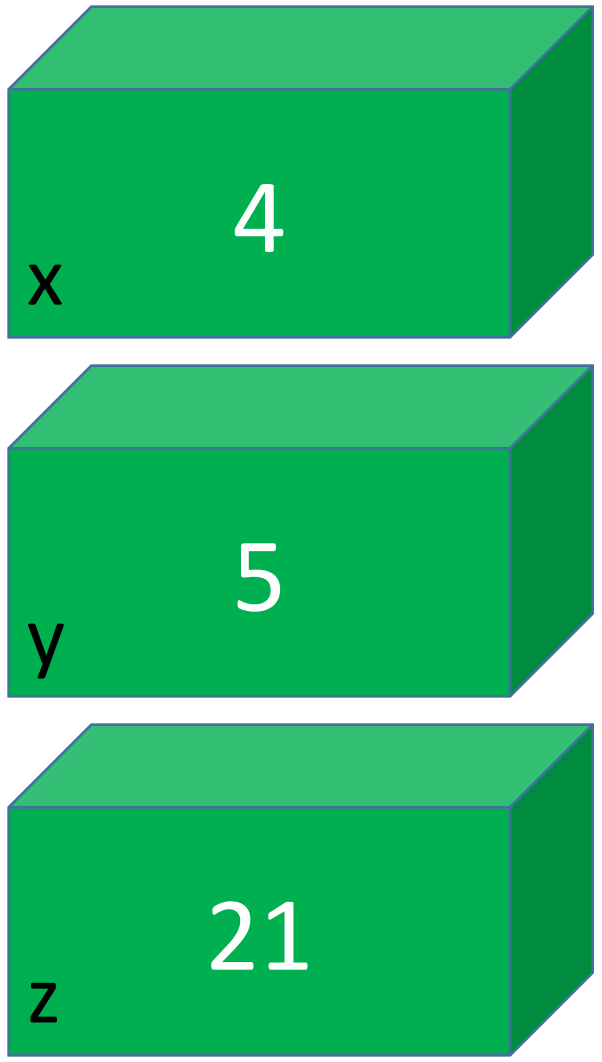
Example 4: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x=x*x
y=y+2
z=x*y
z=z+1
```

Next



Special Assignment Operators

Certain types of operations are very common, and special assignment operators have been defined.

For example:

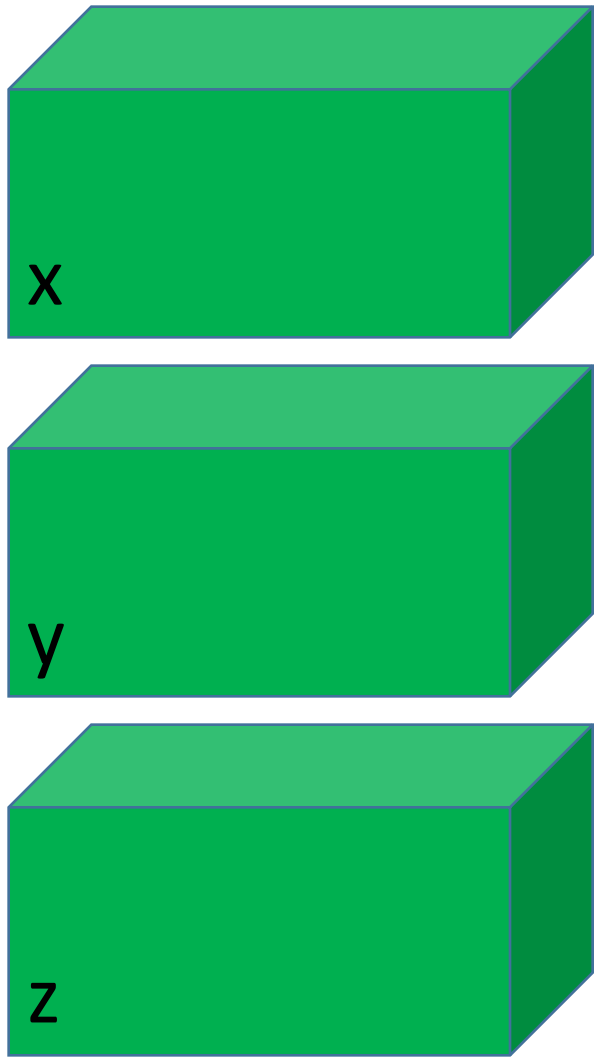
`x = x + c`	can be written:	`x += c`
`x = x - c`	can be written:	`x -= c`
`x = x * c`	can be written:	`x *= c`
`x = x / c`	can be written:	`x /= c`

Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```



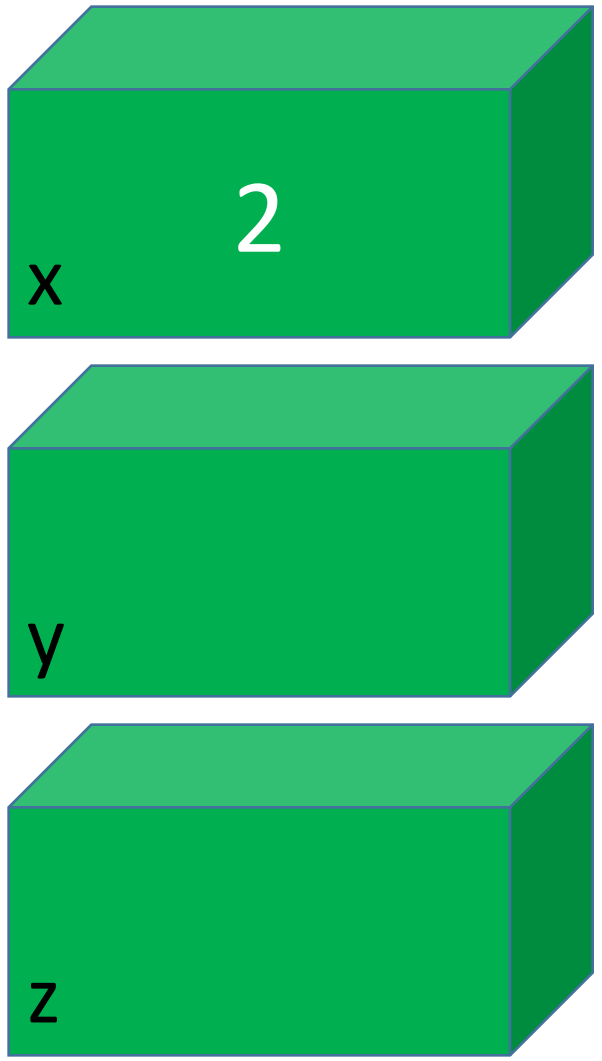
Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

Next

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```



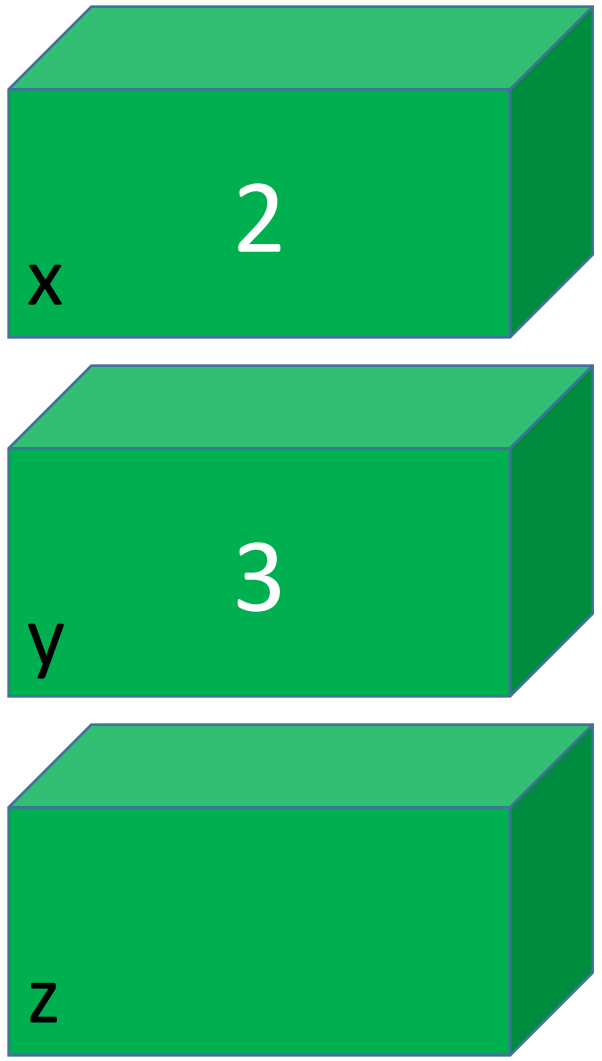
Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```

Next



Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

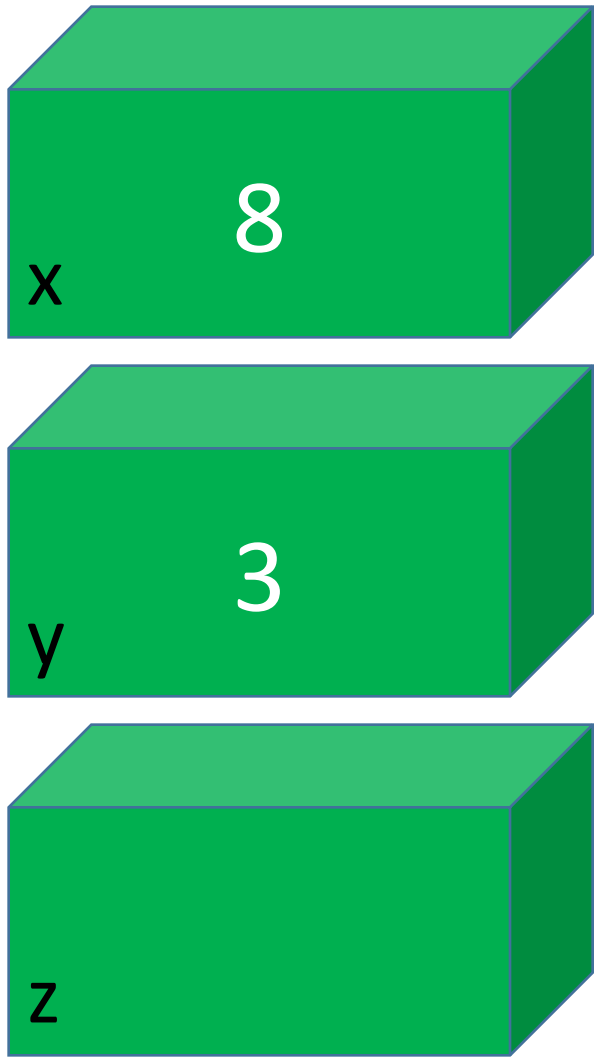
Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```

Next

The variable x gets its prior value, x, multiplied by 4.

This was equivalent to:
x=x*4



Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

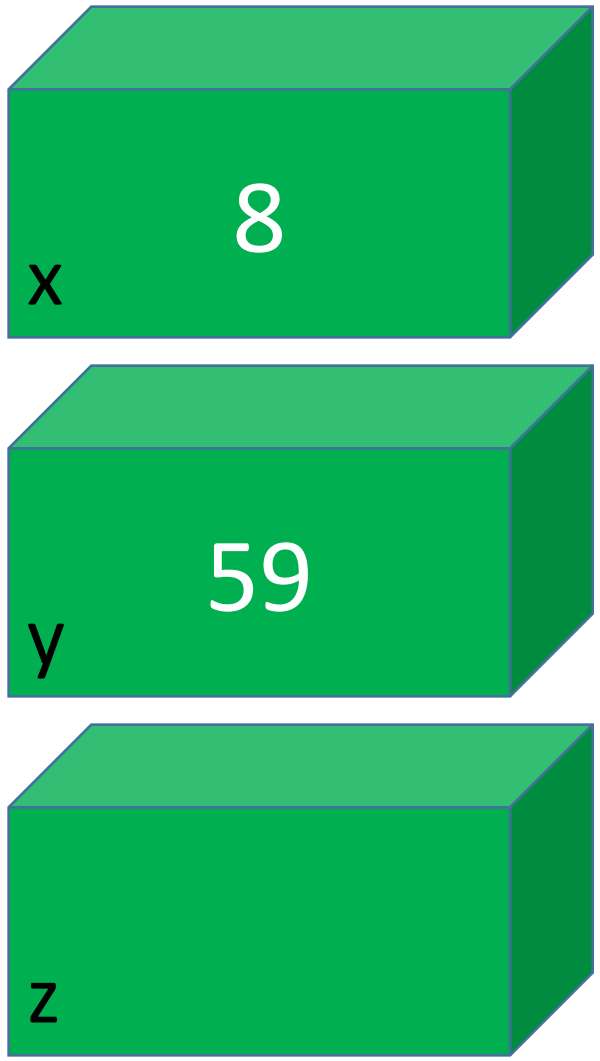
Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```

Next

The variable y has its value incremented by 7 times the value of x, or 56.

This was equivalent to:
y=y+7*x



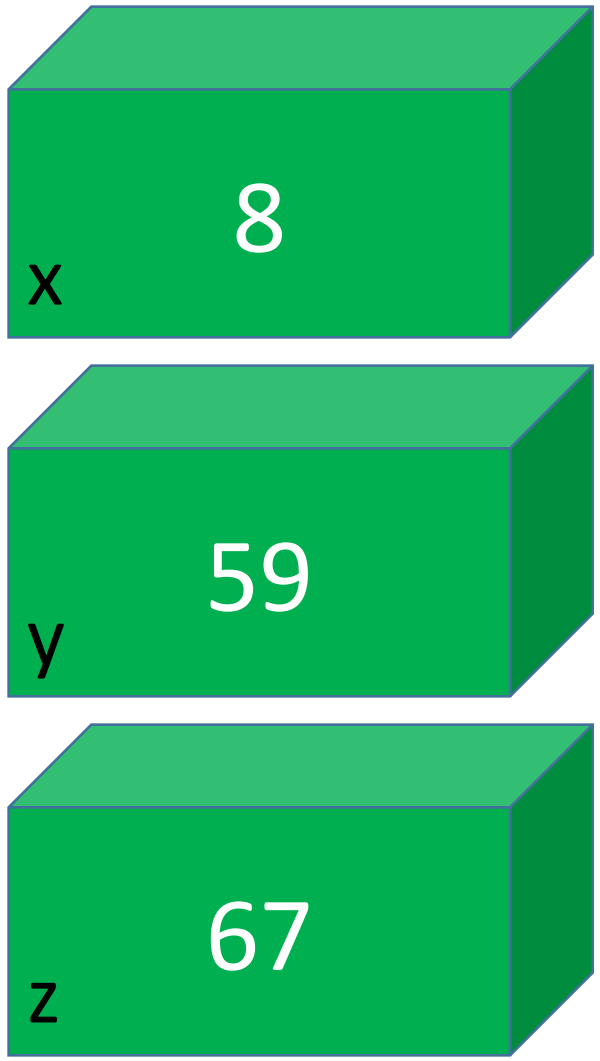
Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```

Next



Example 5: Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

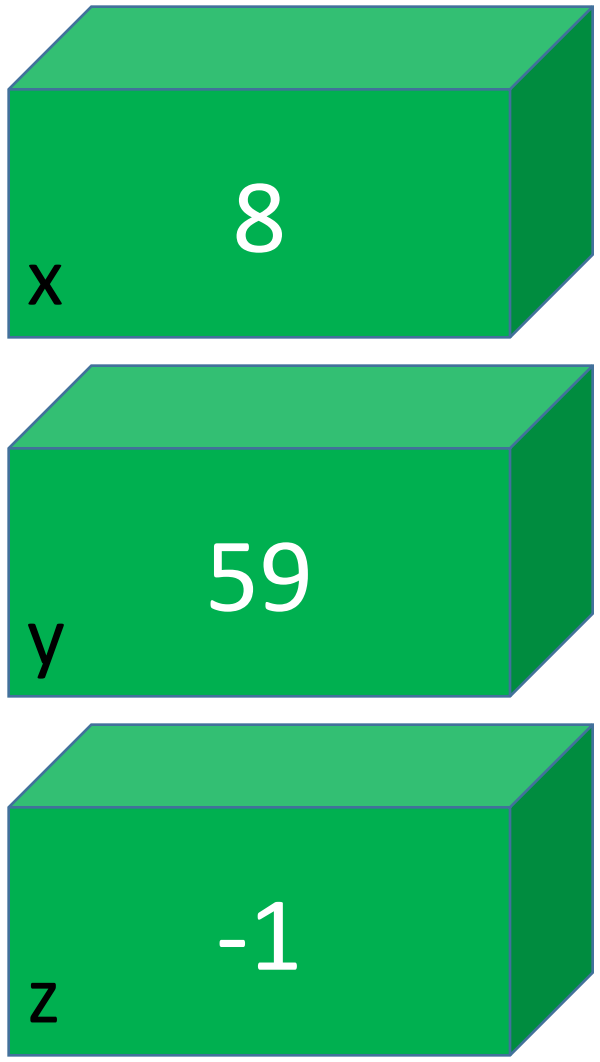
Consider the following sequences of assignments.
What is the value of each variable at the end?

```
x=2
y=3
x*=4
y+=7*x
z=x+y
z-=z+1
```

Next

The variable z has its value decremented by the value of z plus one.

This was equivalent to:
z=z-(z+1)



A reminder of output

Basic output is to the “console” – the screen that shows the program output.

To show the value of a variable, we “print” it:

```
print(x)
```

The x can be:

- a constant fixed value (a “literal”)
- a variable
- or an expression
  - a combination of literals, constants, variables, functions, etc. that compute to a single value

Example 6: `Print(x)`

ENGINEERING
TEXAS A&M UNIVERSITY

What is the output of the following program?

```
a = 10
b = 2**4
b -= a
c = b//2
print(c)
```

Example 6: `Print(x)`

ENGINEERING
TEXAS A&M UNIVERSITY

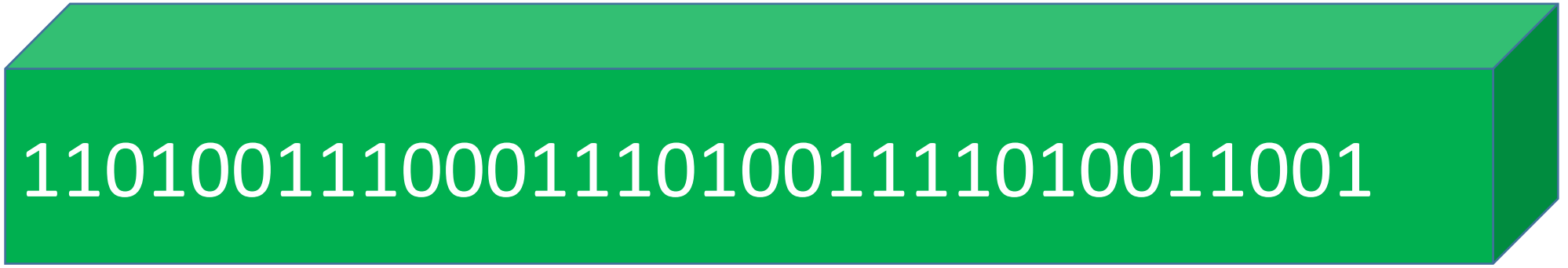
What is the output of the following program?

```
a = 10
b = 2**4
b -= a
c = b//2
print(c)
```

3

Variable Types

- Computer memory consists of a bunch of 1s and 0s
- Remember that a variable describes some area of memory
  - That memory contains only 1s and 0s.
- How do we know how to interpret a set of 1s and 0s?



- The variable **type** says how we should interpret that data

Types of Variables

- There are some standard types that are built in to the language or commonly used. Some of these include:
  - Integers
  - Floating-Point Numbers
  - Booleans
  - Strings
- We'll encounter other types later

Integers

- "Whole numbers", or numbers without a fraction
- Can be positive or negative
- Probably the most commonly used variable type

Examples:

- 1
- 2
- 100
- 0
- -20

Floating-Point Numbers

- Can represent numbers with a decimal point
- May be best to think of these as in "Scientific Notation"
  - A Mantissa (the values, assumed to have 1 digit before the decimal point)
  - An Exponent (the power of 10 that the Mantissa is multiplied by)
  - $1234.567 = 1.234567 * 10^{3}$

Mantissa

Exponent

Examples:

- 1.01
- 2.0
- 100.0
- -20.05
- 0.004

Booleans

- Booleans will have the value True or False
- That's it!
- We'll see a lot more use of Booleans in a couple of weeks…

Examples:

- True
- False

String

- A string is a way of describing text
- Strings are made up of characters “strung” together.
- When we want to define a string, we usually must specify it within quotation marks, otherwise it might look like a variable name(s).
- In Python you can use either single quotes or double quotes

```
"This is a string"
'So is this'
```

Side note: Strings

- If we specify a string with single or double quotes, how do we include a quotation mark in the string itself?
- If we want an apostrophe, use double quotes for the string:

```
"It's"
```

- If we want quotation marks, use single quotes for the string

```
'He said, "What?" to her'
```

- But what if we want both?

What if we want both ‘ and “ in a string?

Two choices:

1. Use triple quotes to begin/end string (single or double):

```
'''He said "I'm tired." '''
"""He said "I'm tired." """
```

2. Use the backslash (\) as an “escape” character before the character you want to specify. (Preferred Method)

```
'He said "I\'m tired." '
"He said \"I'm tired.\" "
```

Knowing a Variable Type

In Python, the type of a variable is implied based on how it was created or last assigned (other languages may require explicit assignments).

`a = 5` → `a` will be an integer.

`a = 5.0` → `a` will be a floating-point number.

`a = True` → `a` will be a Boolean.

`a = "Five"` → `a` will be a string.

Note: it is important to know the type of a variable, or you can get strange effects!

What type is it?

For the following, what is the type of the variable x?

```
x = 2
x = 2.0
x = "True"
x = 2 + 3
x = 2.0 + 3.0
x = 2 / 3
x = 2 / 2
x = 2 // 2
```

What type is it?

For the following, what is the type of the variable x?

```
x = 2
x = 2.0
x = "True"
x = 2 + 3
x = 2.0 + 3.0
x = 2 / 3
x = 2 / 2
x = 2 // 2
```

Integer
Floating-Point
String (x=True would be Boolean)
Integer
Floating-Point
Floating-Point
Floating-Point (dividing integers yields floating-point)
Integer (this is integer division – no remainder)

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = 1 + 2
print(x)
```

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = 1 + 2
print(x)
```

Output:

```
3
```

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = 1.0 + 2.0
print(x)
```

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = 1.0 + 2.0
print(x)
```

Output:

```
3.0
```

Printing a floating-point value will include the decimal point

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = "1" + "2"
print(x)
```

Type and behavior

- Operations on variables/values can vary depending on the type of the operands.
- Let's look at the basic addition operator: +
- What is output by the following?

```
x = "1" + "2"
print(x)
```

Output:

```
12
```

Why did that happen?

- For numerical types, the + operator means addition
- For strings, the + operator means “concatenation”
- Remember, “1” is a **string** (the text representation), not the integer 1
- So, “1” + “2” is the string “1” joined to the string “2”, giving the result, a string “12”.
  - Notice that when the string is printed, the quotation marks don’t appear.

Other operations

- Strings have + defined (concatenation)
  - Needs to be between two strings
  - string + string gives a new string
  - string + int gives an error, though
- How is the operation − defined for strings?
  - It's not! Trying to subtract two strings will give an error

Other operations

- Not every operation is defined for every type (in fact, most won't be)
- For strings, `-` and `/` are not defined at all, `+` is defined between two strings, and `*` is defined for a string and an int

  What do you think `3 * "Howdy"` will be?

Other operations

- Not every operation is defined for every type (in fact, most won't be)
- For strings, `-` and `/` are not defined at all, `+` is defined between two strings, and `*` is defined for a string and an int

  What do you think `3 * "Howdy"` will be?

  `HowdyHowdyHowdy`

Current Assignments

ENGINEERING
TEXAS A&M UNIVERSITY

Complete the BAEN module and quiz *(Engineering Success module is postponed)*

- Due 9/8, by end-of-day

Lab Assignment 2	Due 9/8 by end-of-day
Lab Assignment 2b	Due 9/8 by end-of-day

Preactivity: Complete zyBook chapter 3 and 10.1 - 10.8 prior to class next week