

# Variables and Expressions

Department of Computer Sciences  
University of Wisconsin-Madison

Readings:

Chapter 2 of Think Python,  
Chapter 3 of Python for Everybody

Due: PI

# Learning Objectives

Evaluate expressions by identifying:

- operators and operands
- literal values and variables
- correct order of operations

Write correct Boolean expressions

- containing Boolean operators “or” and “and”

Write assignment statements

- with variables following proper naming rules

Define, give examples of, and identify 3 kinds of errors

- Syntax, runtime, and semantic

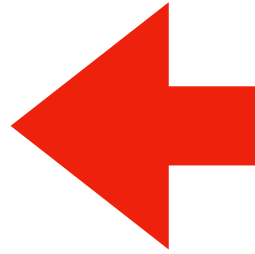
Write code to perform computations with

- int, float, string, and bool types

# Today's Outline

## Review

- Operator Precedence



Expressions, Variables, and Assignments

*Demos*

Bugs



*Demos*

Naming variables

*Demos*

# Unordered

What is it?	Python Operator
comparison	==, !=, <, <=, >, >=
signs	+x, -x
AND	and
add/subtract	+, -
exponents	**
NOT	not
OR	or
multiply/divide	*, /, //, %

# Ordered by Precedence

What is it?	Python Operator

simplify first

simplify last

# Unordered

What is it?	Python Operator
comparison	==, !=, <, <=, >, >=
signs	+x, -x
AND	and
add/subtract	+, -
NOT	not
OR	or
multiply/divide	*, /, //, %

# Ordered by Precedence

What is it?	Python Operator
exponents	**

simplify first

simplify last

# Unordered

What is it?	Python Operator
comparison	==, !=, <, <=, >, >=
AND	and
add/subtract	+, -
NOT	not
OR	or
multiply/divide	*, /, //, %

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x

simplify first

simplify last

# Unordered

What is it?	Python Operator
comparison	==, !=, <, <=, >, >=
AND	and
add/subtract	+, -
NOT	not
OR	or

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %

simplify first

simplify last

# Unordered

What is it?	Python Operator
comparison	==, !=, <, <=, >, >=
AND	and
NOT	not
OR	or

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -

simplify first

simplify last



# Unordered

What is it?	Python Operator
AND	and
NOT	not
OR	or

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=

simplify first

simplify last

# Unordered

What is it?	Python Operator
AND	and
OR	or

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=
NOT	not

simplify first

simplify last

**Unordered**

What is it?	Python Operator
OR	or

**Ordered by Precedence**

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=
NOT	not
AND	and

**simplify first**

**simplify last**

**Unordered**

What is it?	Python Operator

**Ordered by Precedence**

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=
NOT	not
AND	and
OR	or

**simplify first**

**simplify last**

Unordered

What is it?	Python Operator

Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=
NOT	not
AND	and
OR	or

simplify first

10 - -2 // 3

simplify last

# Unordered

What is it?	Python Operator

# Ordered by Precedence

What is it?	Python Operator
exponents	**
signs	+x, -x
multiply/divide	*, /, //, %
add/subtract	+, -
comparison	==, !=, <, <=, >, >=
NOT	not
AND	and
OR	or

simplify first

simplify last

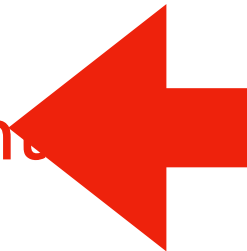
1+1==2 or 3 \*\* 10000000 > 2 \*\* 20000000

logical operators  
can "short circuit"

# Today's Outline

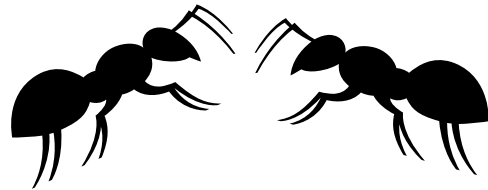
Review

Expressions, Variables, and Assignment



*Demos*

Bugs



*Demos*

Naming variables

*Demos*

# Expressions

**Expressions** are a mix of **operators** and **operands**. For example:

$5 + 5$

$(8/2) ** 2 * 3.14$

$3 * 3 > 4 + 4$

$3 \% 2 == 0$  or  $3 \% 2 == 1$



# Expressions

**Expressions** are a mix of **operators** and **operands**. For example:

$5 + 5$

$(8/2) ** 2 * 3.14$

$3 * 3 > 4 + 4$

$3 \% 2 == 0$  or  $3 \% 2 == 1$

Each of these operands is an example of a *literal*: a fixed value

# Expressions

Expressions are a mix of operators and operands. For example:

$x + y$

$(\text{diameter}/2) ** 2 * \text{pi}$

$\text{value1} * \text{value1} > \text{value2} + \text{value2}$

$\text{num} \% 2 == 0$  or  $\text{num} \% 2 == 1$

An operand may also be a *variable*: not fixed

# Expressions

Expressions are a mix of operators and operands. For example

$x + y$

(diameter

value l

num %

Quick Test! Circle the literals (others are variables)

1. 0
2. zero
3. num1
4. True
5. hello
6. "goodbye"

An operand may also be a variable: not fixed

# Expressions

**Expressions** are a mix of **operators** and **operands**. For example

**x + y**

**(diameter**

**value l**

**num %**

Quick Test! Circle the **literals** (others are **variables**)

1. 0

2. zero

3. num1

4. True

5. hello

6. "goodbye"

An operand may also be a **variable**: not fixed

# Expressions

**Expressions** are a mix of **operators** and **operands**. For example

**x + y**

**(diameter**

**value l**

**num %**

Quick Test! Circle the **literals** (others are **variables**)

1. 0

2. zero

3. num1

4. True

5. hello

6. "goodbye"

An operand may also be a **variable**: not fixed

# Expressions

**Expressions** are a mix of **operators** and **operands**. For example

$x + y$

(diameter)

value 1

num %

Quick Test! Circle the **literals** (others are **variables**)

1. 0

2. zero

3. num1

4. True

5. hello

6. "goodbye"

An operand may also be a **variable**: not fixed

# Expressions

**Expressions** are a mix of **operators** and **operands**. For example

$x + y$

(diameter

value l

$\text{num} \%$

Quick Test! Circle the **literals** (others are **variables**)

1. 0

2. zero

3. num1

4. True

5. hello

6. "goodbye"

An operand may also be a **variable**: not fixed

How do we put a value in a variable?

# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

**x + y**

**(diameter/2) \*\* 2 \* pi**

**value1 \* value1 > value2 + value2**

**num % 2 == 0 or num % 2 == 1**



# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

$x + y$

$(\text{diameter}/2) ** 2 * \text{pi}$

$\text{value1} * \text{value1} > \text{value2} + \text{value2}$

$\text{num} \% 2 == 0$  or  $\text{num} \% 2 == 1$

# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

= **x** + **y**

= (**diameter**/2) \*\* 2 \* **pi**

= **value1** \* **value1** > **value2** + **value2**

= **num** % 2 == 0 or **num** % 2 == 1

# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

```
total = x + y
```

```
area = (diameter/2) ** 2 * pi
```

```
is_bigger = value1 * value1 > value2 + value2
```

```
is_even_or_odd = num % 2 == 0 or num % 2 == 1
```

# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

**total** = **x** + **y**

**area** = (**diameter**/2) \*\* 2 \* **pi**

**is\_bigger** = **value1** \* **value1** > **value2** + **value2**

**is\_even\_or\_odd** = **num** % 2 == 0 or **num** % 2 == 1

**Expression**

# Assignment

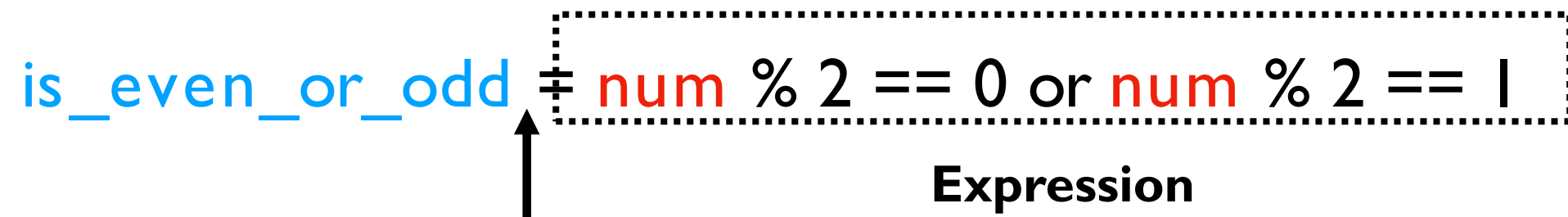
An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

**total** = **x** + **y**

**area** = (**diameter**/2) \*\* 2 \* **pi**

**is\_bigger** = **value1** \* **value1** > **value2** + **value2**

**is\_even\_or\_odd** = **num** % 2 == 0 or **num** % 2 == 1



The diagram shows the assignment statement `is_even_or_odd = num % 2 == 0 or num % 2 == 1`. An upward-pointing arrow is positioned below the equals sign, and the text "Assignment Operator" is centered below the arrow. A dashed rectangular box encloses the expression `num % 2 == 0 or num % 2 == 1`, with the text "Expression" centered below the box.

**Assignment Operator**

**Expression**

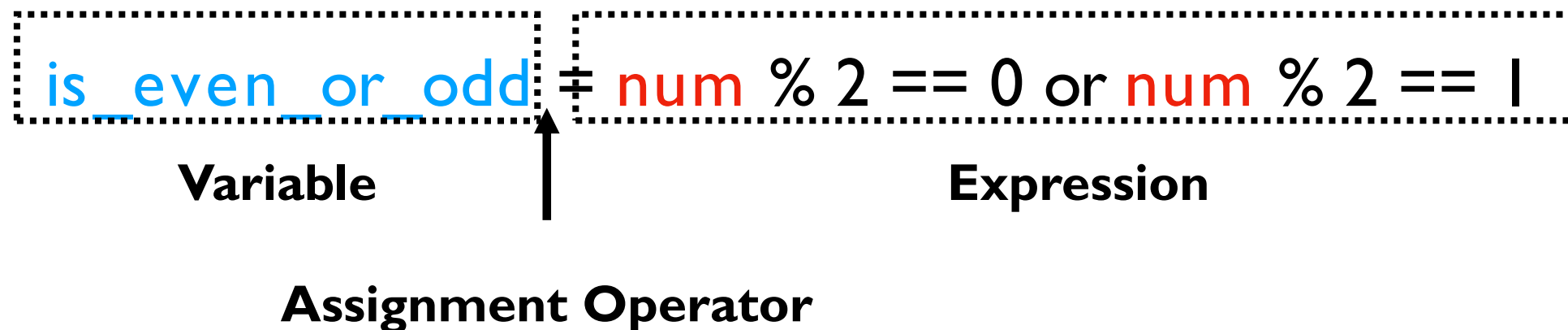
# Assignment

An **assignment** computes an expression (maybe a simple one) and puts the result in a variable:

**total** = **x** + **y**

**area** = (**diameter**/2) \*\* 2 \* **pi**

**is\_bigger** = **value1** \* **value1** > **value2** + **value2**

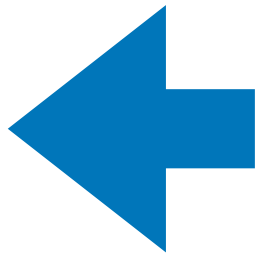


# Today's Outline

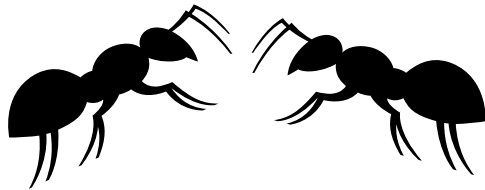
Review

Expressions, Variables, and Assignments

*Demos*



Bugs



*Demos*

Naming variables

*Demos*

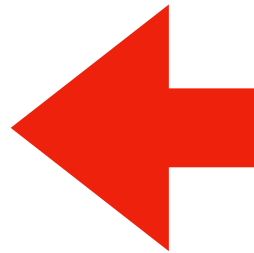
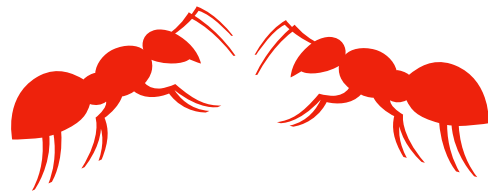
# Today's Outline

Review

Expressions, Variables, and Assignments

*Demos*

Bugs



*Demos*

Naming variables

*Demos*



# Categories of Errors

1

dog cat the of chase any

[word soup, not grammatically sensible]

2

3

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
  - `5 = x`

2

3

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
- `5 = x`

2

this sentence is false

[grammatical, but my head explodes if I think about it]

3

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
- `5 = x`

2

## Runtime Error

- Need to run to find out whether it will crash
- Appears with different names (TypeError, ZeroDivisionError, etc)
- `x = 5 / 0`

3

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
- `5 = x`

2

## Runtime Error

- Need to run to find out whether it will crash
- Appears with different names (TypeError, ZeroDivisionError, etc)
- `x = 5 / 0`

3

one week is 10 days long  
[grammatical, coherent, but incorrect]

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
- `5 = x`

2

## Runtime Error

- Need to run to find out whether it will crash
- Appears with different names (TypeError, ZeroDivisionError, etc)
- `x = 5 / 0`

3

## Semantic Error

- It runs with no error, but you get the wrong answer
- `square_area = square_side * 2`

# Categories of Errors

1

## Syntax Error

- It never makes sense in any context; Python doesn't even run
- `5 = x`

2

## Runtime Error

- **what kind of error is the worst?**
- `x = 5 / 0`

3

## Semantic Error

- It runs with no error, but you get the wrong answer
- `square_area = square_side * 2`

# Today's Outline

Review

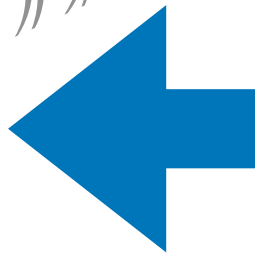
Expressions, Variables, and Assignments

*Demos*

Bugs



*Demos*



Naming variables

*Demos*



# Example: int expressions

```
seconds = 12345
```

**Print out hours, minutes, and seconds**



# Example: float expressions

## Compound growth:

- you start with \$1000
- every year it grows by 7%
- you wait 30 years
- how much do you have at the end?

year 0: \$1000

year 1: \$1070

year 2: ...



# Example: string expressions

## Visually compare two scores:

- Alice has 10 points
- Bob has 8 points

## Desired output:

```
alice: |||||  
bob:  |||||
```

even better

```
alice: |||||  
bob:  |||||
```

# Example: bool expressions

**Bounds check:** is the value between 0 and 100?

**YES**

**output is**

you may continue: True

**NO**

**output is**

you may continue: False

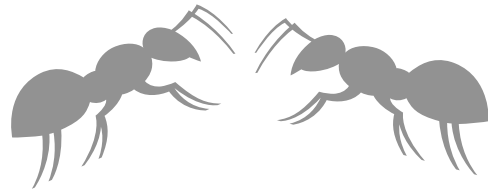
# Today's Outline

Review

Expressions, Variables, and Assignments

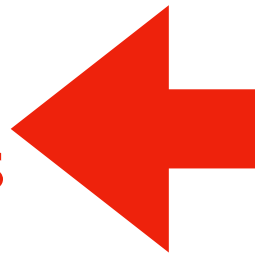
*Demos*

Bugs



*Demos*

Naming variables



*Demos*

# What Variable Names are Allowed?

`1st_score = 100` [bad variable]

`score_1 = 100` [good variable]

`firstScore = 100` [not a recommended variable]

`first_score = 100` [recommended variable]

current rules are quite complex:

<https://www.python.org/dev/peps/pep-3131>

please don't use camel case:

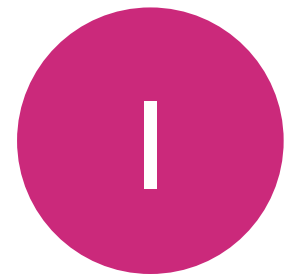
<https://www.python.org/dev/peps/pep-0008/>

Python 3 has become friendlier to non-English programmers

`quero_café = True`

← this is allowed, and  
different than "e"

# Rules for naming variables



Only use letters a-z (upper and lower), numbers, and underscores



Don't start with a number



Don't use Python keywords (e.g., and, False, etc)

For 220, you may use only variables containing English alphabets

# Rules for naming variables

1

Only use letters a-z (upper and lower), numbers, and underscores

2

Don't start with a number

3

Don't use Python keywords (e.g., and, False, etc)

## **GOOD:**

```
cs220  
CS220  
cs_220  
_cs220
```

## **BAD:**

```
220class  
and  
pi3.14  
x!
```

*what rules are violated?*



# Rules for naming variables

1

Only use letters a-z (upper and lower), numbers, and underscores

2

Don't start with a number

3

Don't use Python keywords (e.g., and, False, etc)

## **GOOD:**

```
cs220  
CS220  
cs_220  
_cs220
```

## **BAD:**

```
220class  
and  
pi3.14  
x!
```

2

# Rules for naming variables

1

Only use letters a-z (upper and lower), numbers, and underscores

2

Don't start with a number

3

Don't use Python keywords (e.g., and, False, etc)

## GOOD:

```
cs220  
CS220  
cs_220  
_cs220
```

## BAD:

```
220class 2  
and 3  
pi3.14  
x!
```

# Rules for naming variables

1

Only use letters a-z (upper and lower), numbers, and underscores

2

Don't start with a number

3

Don't use Python keywords (e.g., and, False, etc)

## GOOD:

```
cs220  
CS220  
cs_220  
_cs220
```

## BAD:

```
220class 2  
and 3  
pi3.14 1  
x!
```

# Rules for naming variables

1

Only use letters a-z (upper and lower), numbers, and underscores

2

Don't start with a number

3

Don't use Python keywords (e.g., and, False, etc)

## GOOD:

```
cs220  
CS220  
cs_220  
_cs220
```

## BAD:

```
220class 2  
and 3  
pi3.14 1  
x! 1
```

# Identifying keywords

3

Don't use Python keywords (e.g., and, False, etc)

*How to figure out if something is a Python keyword?*

- Python keywords turn green in color in jupyter notebook
- If used as a variable, that keyword will no longer work as intended!

**GOOD:**

```
In [ ]: 1 player
        2 start
        3 hurricane_speed
        4 final_score
```

**BAD:**

```
In [ ]: 1 print
        2 list
        3 type
        4 bool
```

*Pay attention to green colorization within jupyter notebook*

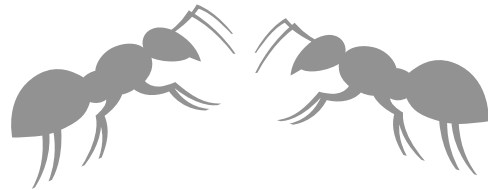
# Today's Outline

Review

Expressions, Variables, and Assignments

*Demos*

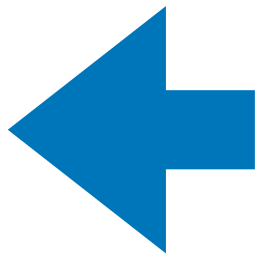
Bugs



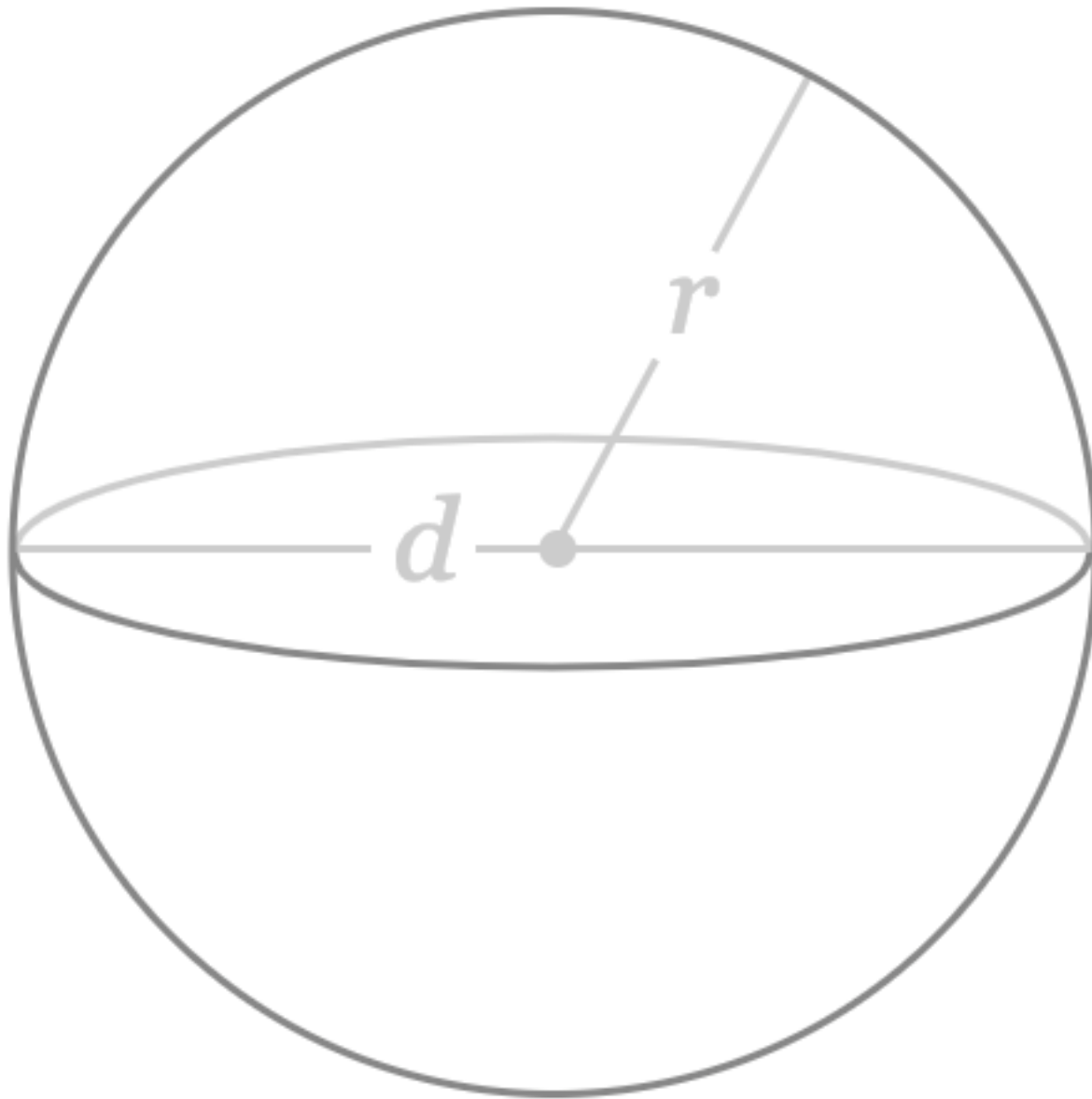
*Demos*

Naming variables

*Demos*



# Practice: Sphere Volume

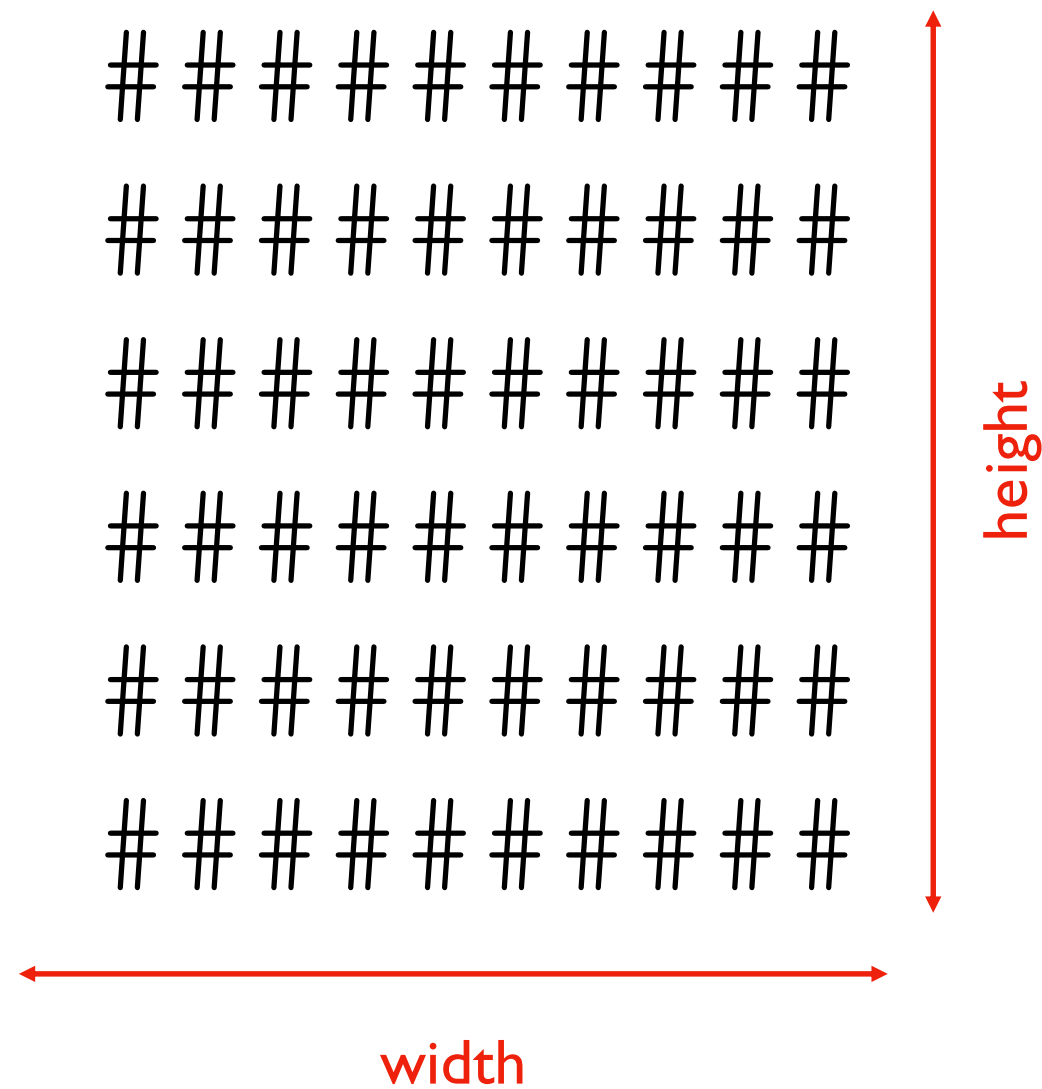


$$V = \frac{4}{3} \pi r^3$$

**extension:** find radius given a volume

# Practice: Character Art - Block

write some code to draw the following:





# Practice: Quadratic Formula

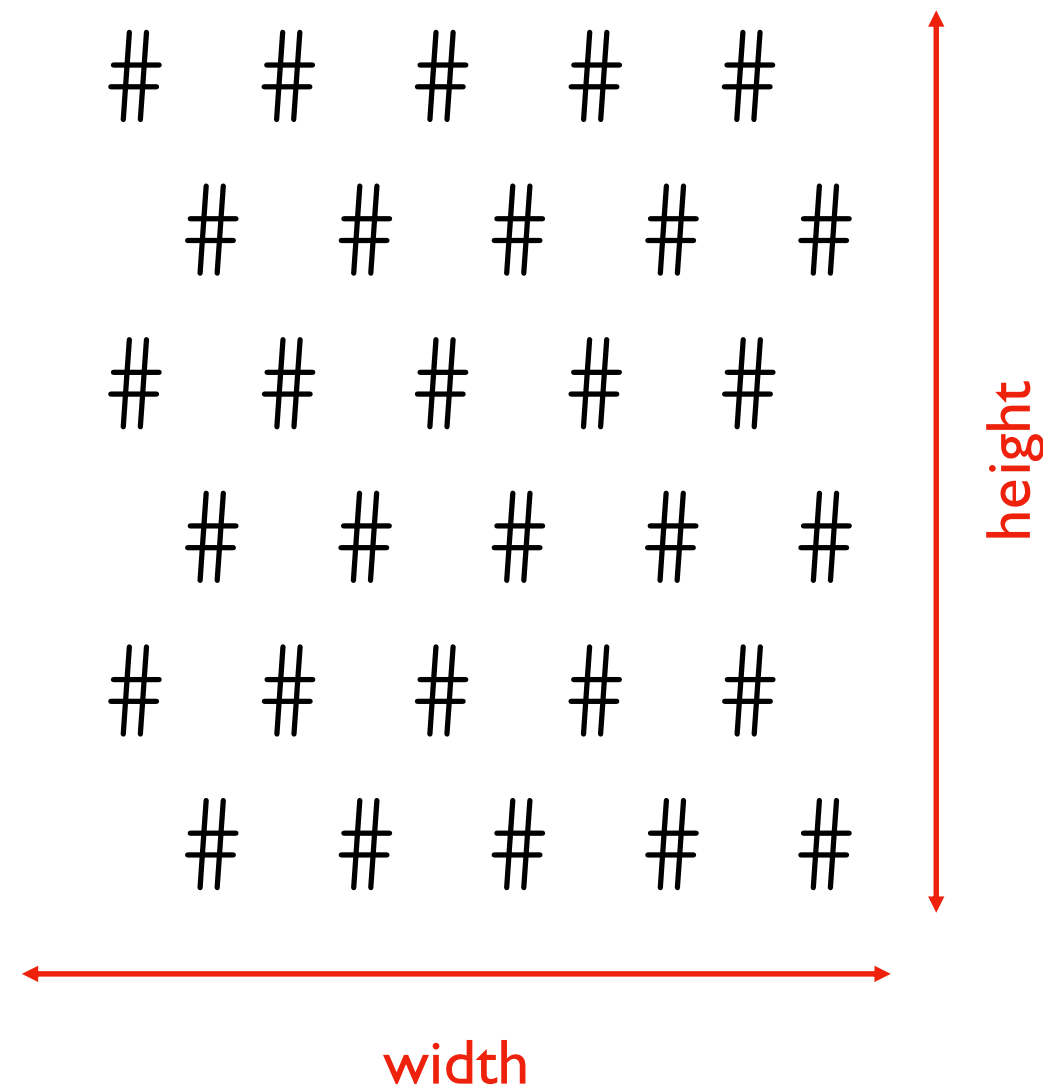
$$ax^2 + bx + c = 0$$

*what values of x satisfy the above?*

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# Challenge\*: Checkers

write some code to draw the following:



\* Challenge = beyond what you would be asked to do on an exam

# Challenge: Border

**write some code to draw the following:**

A diagram illustrating a 2D grid of hash symbols (#). The grid is 6 rows high and 10 columns wide. The symbols are arranged in a rectangular border, with the top and bottom rows being solid, and the middle four rows having symbols only at the first and last columns. A red double-headed arrow at the bottom indicates the 'width' of the grid, and a red double-headed arrow on the right indicates the 'height' of the grid.

# Challenge: Snake

**write some code to draw the following:**

A diagram illustrating a 2D grid of hash symbols (#). The grid is 6 rows high and 10 columns wide. The symbols are arranged in a pattern that suggests a specific layout, possibly for a document or a form. A red double-headed arrow at the bottom indicates the width, and a red double-headed arrow on the right indicates the height.

#	#	#	#	#	#	#	#	#	#
#									
#	#	#	#	#	#	#	#	#	#
								#	
#	#	#	#	#	#	#	#	#	#
#									
#	#	#	#	#	#	#	#	#	#
								#	

width

height