University of Colorado **Boulder**

Department of Computer Science
CSCI 2824: Discrete Structures
Chris Ketelsen

Sets and Functions

Functions

Functions

The idea of a function is ubiquitous in Computer Science

In [ ]:
```
int square(int n){
    int s;
    s = n * n;
    return s;
}
```

A function is a procedure or routine that takes some kind of input, produces some kind of output (sometimes), and possibly causes some kind of side-effect.

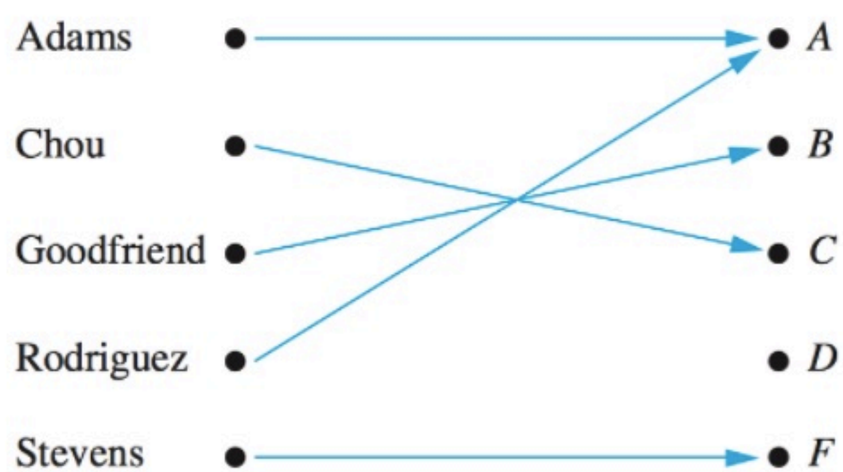
Sometimes we make a distinction between a **function** that takes **inputs** and produces **outputs** and a **procedure** which may take some inputs but may cause a **side-effect** instead of returning a value.

In this course we will use the narrower definition of a function

Functions

In this class we'll abstract this to the mathematical idea of a function

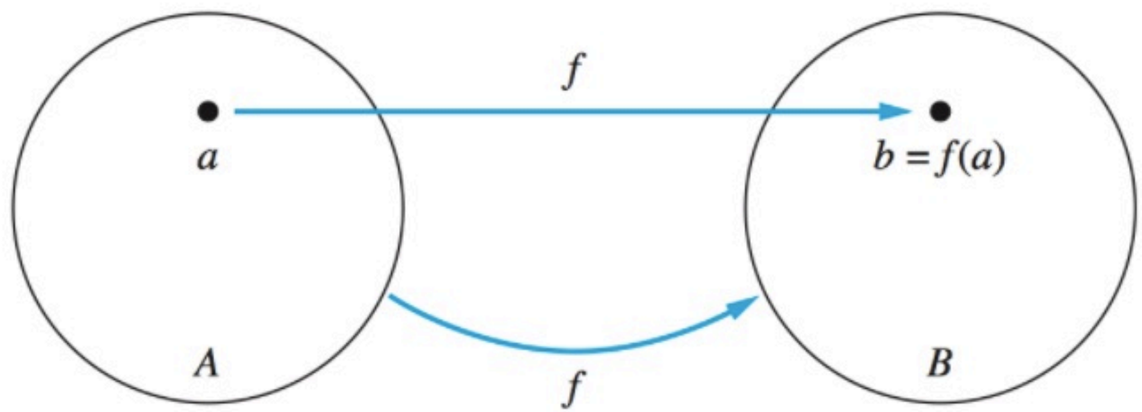
Example: Suppose you have a function that assigns grades to Discrete Structures students



The grade assignment function takes students {Adams, Chou, Stevens, ...} and maps them to letter grades {A, B, C, D, F}

Functions

The key idea behind functions is that for each member of the input set A they produce exactly one member of the output set B



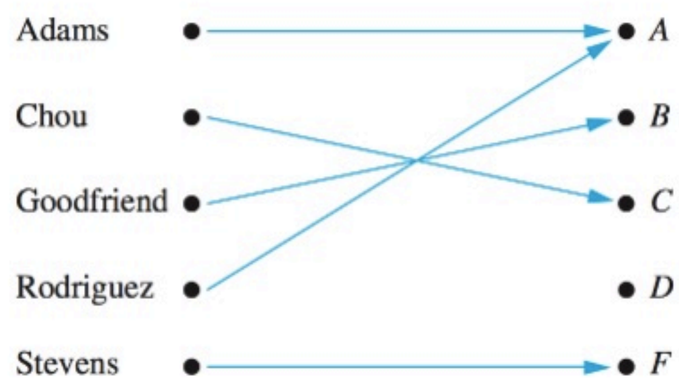
Def: Let A and B be nonempty sets. A **function** f from A to B is an assignment of exactly one element of B to each element of A. We write $f(a) = b$ to indicate that b is the unique element of B assigned to A by f. If f is function from A to B we write $f : A \to B$

Functions

Def: The **domain** of f is the set A that f maps *from*

Def: The **codomain** of f is set of elements that f *could* map *to*

Def: The **range** of f is the subset of B that f maps *does* map to



$$domain = \{\text{Adams, Chou, Rodriguez, Goodfriend, Stevens}\}$$

$$codomain = \{\text{A, B, C, D, F}\} \quad range = \{\text{A,B,C,F}\}$$

Functions

Example: Consider again this C++ function

In [ ]:
```
int square(int n){
    int s;
    s = n * n;
    return s;
}
```

Q: What are the `square` function's domain, codomain, and range?

Functions

Example: Consider again this C++ function

In [ ]:
```
int square(int n){
    int s;
    s = n * n;
    return s;
}
```

Q: What are the `square` function's domain, codomain, and range?

A: From type declarations, can think of as $f : \mathbb{Z} \to \mathbb{Z}$

As defined:

- Domain is the set of all integers
- Codomain is also the set of all integers
- Range is the set of perfect squares $\{0, 1, 4, 9, \ldots\}$

Functions

Note that the choice of domain affects the nature of f

Example: Define $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = \frac{1}{x}$

Q: Is this a valid function?

Functions

Note that the choice of domain affects the nature of f

Example: Define $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = \frac{1}{x}$

Q: Is this a valid function?

A: **NO!** Because f doesn't map every element of domain into range because can't plug in zero

Q: What if we change it to $f : (\mathbb{R} - \{0\}) \to \mathbb{R}$?

Functions

Note that the choice of domain affects the nature of f

Example: Define $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = \frac{1}{x}$

Q: Is this a valid function?

A: **NO!** Because f doesn't map every element of domain into range because can't plug in zero

Q: What if we change it to $f : (\mathbb{R} - \{0\}) \to \mathbb{R}$?

A: Now f maps every element of its domain to something in its range, so it's a function

Functions

Example: Consider this C++ function

In [ ]:
```
int add(int m, int n){
    int sum;
    sum = m + n;
    return sum;
}
```

Q: What are the add function's domain, codomain, and range?

Functions

Example: Consider this C++ function

In [ ]:
```
int add(int m, int n){
    int sum;
    sum = m + n;
    return sum;
}
```

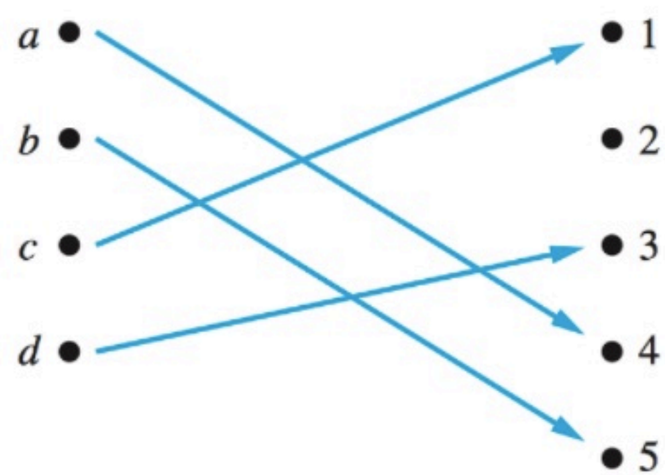
Q: What are the add function's domain, codomain, and range?

A: From type declarations, can think of as $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$

- Domain is the set of all ordered pairs of ints (m, n)
- Codomain is also the set of all integers
- Range is also set of all integers

One-to-One Functions

Some functions never assign the same value to two different domain elements. These functions are called **one-to-one** functions



Def: A function f is said to be **one-to-one** if and only if $f(a) = f(b)$ implies that $a = b$ for all a and b in the domain of f

One-to-One Functions

Example: $f(n) = n^3$ is one-to-one

Strategy: To prove that a function is one-to-one, assume $f(m) = f(n)$ and show it must be the case that $m = n$

Proof: Assume $n^3 = m^3$.

One-to-One Functions

Example: $f(n) = n^3$ is one-to-one

Strategy: To prove that a function is one-to-one, assume $f(m) = f(n)$ and show it must be the case that $m = n$

Proof: Assume $n^3 = m^3$.

Taking the cube-root of both sides yields $n = m$, so f is one-to-one

Example: $f(n) = n^2$ is **not** one-to-one

Strategy: Show that there is at least one pair m and n such that $m \neq n$ but $f(m) = f(n)$

Proof:

One-to-One Functions

Example: $f(n) = n^3$ is one-to-one

Strategy: To prove that a function is one-to-one, assume $f(m) = f(n)$ and show it must be the case that $m = n$

Proof: Assume $n^3 = m^3$.

Taking the cube-root of both sides yields $n = m$, so f is one-to-one

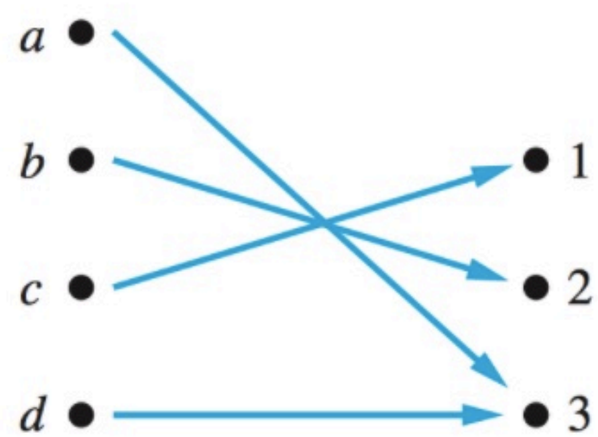
Example: $f(n) = n^2$ is **not** one-to-one

Strategy: Show that there is at least one pair m and n such that $m \neq n$ but $f(m) = f(n)$

Proof: Let $m = 2$, $n = -2$, then $f(2) = 2^2 = 4 = (-2)^2 = f(n)$

Onto Functions

Some functions have the property that they actually do map to every element in their codomain (i.e. their codomain and range are the same). These functions are called **onto** functions.



Def: A function f from A to B is called **onto** if and only if for every element of $b \in B$ there is an element $a \in A$ s.t. $f(a) = b$

Onto Functions

Example: $f(m, n) = m + n$ where $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is onto

Strategy: Given an element of the codomain, give a procedure for finding the associated element of the domain

Proof: Suppose $f(m, n) = m + n = b$,

Let n be any integer and set $m = b - n$

Q: Is $f(m, n) = m + n$ one-to-one?

Onto Functions

Example: $f(m, n) = m + n$ where $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is onto

Strategy: Given an element of the codomain, give a procedure for finding the associated element of the domain

Proof: Suppose $f(m, n) = m + n = b$,

Let n be any integer and set $m = b - n$

Q: Is $f(m, n) = m + n$ one-to-one?

A: Not by a long-shot. Suppose $f(m, n) = 3$, then

$$f(-1, 4) = f(0, 3) = f(1, 2) = f(128, -125) = 3$$

Onto Functions

Example: $f(n) = n^2$ where $f : \mathbb{Z} \to \mathbb{Z}$ is not onto

Strategy: Show that there is at least one element of the codomain that can't be reached by the function

Proof: There is no $n \in \mathbb{Z}$ such that $f(n) = -1$

Q: What if we redefined f to be $f : \mathbb{Z} \to \mathbb{N}$?

Onto Functions

Example: $f(n) = n^2$ where $f : \mathbb{Z} \to \mathbb{Z}$ is not onto

Strategy: Show that there is at least one element of the codomain that can't be reached by the function

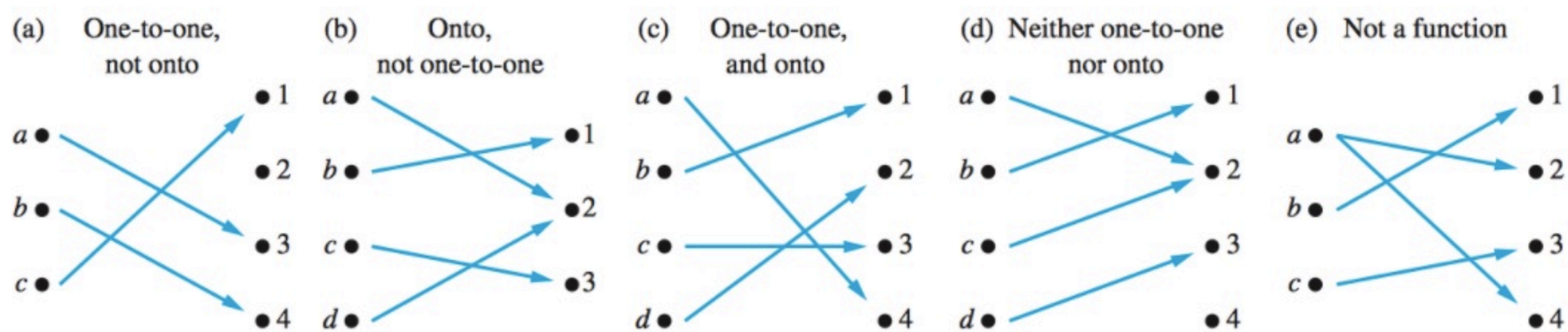
Proof: There is no $n \in \mathbb{Z}$ such that $f(n) = -1$

Q: What if we redefined f to be $f : \mathbb{Z} \to \mathbb{N}$?

A: Still not onto because no $n \in \mathbb{Z}$ s.t. $f(n) = 7$

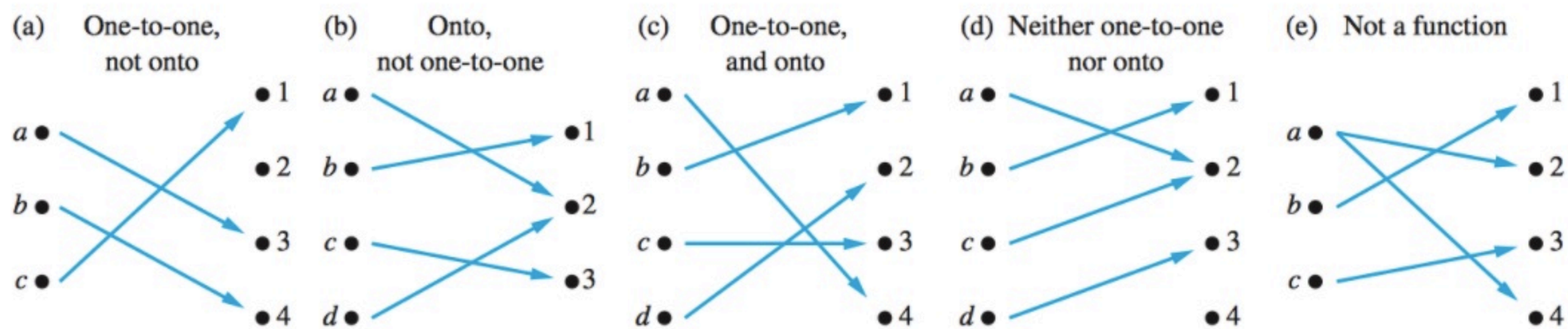
One-to-One and Onto Functions

Example: Classify each of these functions



One-to-One and Onto Functions

Example: Classify each of these functions



Inverse Functions

Some nice things happen when f is **both** one-to-one and onto

If f is both one-to-one and onto then

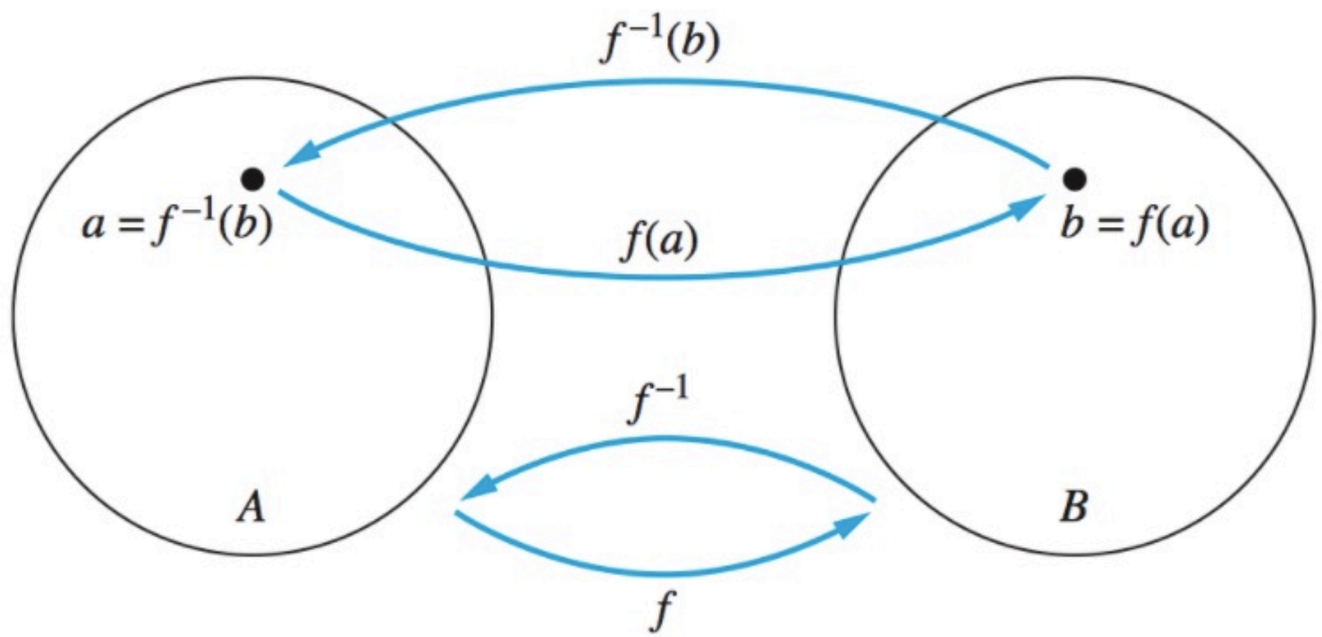
- f maps everything from A to unique element of B
- Every element of B has an element of A that maps onto it

This implies that there is a unique one-to-one **correspondence** between elements of A and elements of B

When this happens it means we can go **back and forth** between A and B via an **inverse** function, f^{-1}

Inverse Functions

Def: Let f be a one-to-one and onto function from A to B. Then there exists an inverse function, f^{-1}, such that $f^{-1}(b) = a$ when $f(a) = b$



Inverse Functions

Example: Inverse of $f(x) = x^3$ where $f : \mathbb{R} \to \mathbb{R}$ is $f^{-1}(y) = \sqrt[3]{y}$

Example: $f(x) = x^2$ where $f : \mathbb{R} \to \mathbb{R}$ does not have an inverse because f is not one-to-one (because $f(2) = 4$ and $f(-2) = 4$)

Q: Could we redefine f so that it does have an inverse?

Inverse Functions

Example: Inverse of $f(x) = x^3$ where $f : \mathbb{R} \to \mathbb{R}$ is $f^{-1}(y) = \sqrt[3]{y}$

Example: $f(x) = x^2$ where $f : \mathbb{R} \to \mathbb{R}$ does not have an inverse because f is not one-to-one (because $f(2) = 4$ and $f(-2) = 4$)

Q: Could we redefine f so that it does have an inverse?

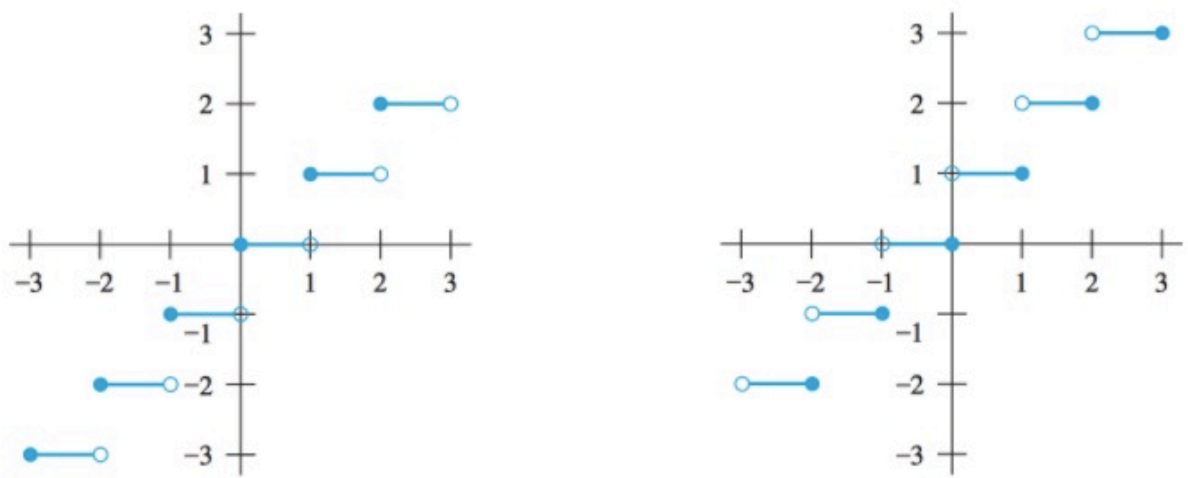
A: **Yes!** If we restrict the domain to the set of nonnegative real numbers then f becomes one-to-one

Its inverse is then $f^{-1}(y) = \sqrt{y}$

Some Important Functions

Def: The **floor** function, denoted $\lfloor x \rfloor$, assigns to the real number x the largest integer that is less than or equal to x. The **ceiling** function, denoted $\lceil x \rceil$, assigns to the real number x the smallest integer that is less than or equal to x.

Remark: Floor and Ceiling crop up constantly in CS



(a) $y = \lfloor x \rfloor$ (b) $y = \lceil x \rceil$

Some Important Functions

Def: The **floor** function, denoted $\lfloor x \rfloor$, assigns to the real number x the largest integer that is less than or equal to x. The **ceiling** function, denoted $\lceil x \rceil$, assigns to the real number x the smallest integer that is less than or equal to x.

Examples:

$$\lfloor 3.5 \rfloor = 3 \qquad \lfloor 5 \rfloor = 5 \qquad \lfloor -2.5 \rfloor = -3$$

$$\lceil 3.5 \rceil = 4 \qquad \lceil 5 \rceil = 5 \qquad \lceil -2.5 \rceil = -2$$

Question: What happens in Python (and most languages) if you cast a `float` as an `int`?

Some Important Functions

Question: What happens in Python (and most languages) if you cast a `float` as an `int`?

In [4]:
```
bar = 2.5
foo1 = int(bar)
foo2 = int(-bar)
```

What are the values of `foo1` and `foo2`?

Some Important Functions

Question: What happens in Python (and most languages) if you cast a `float` as an `int`?

In [4]:
```
bar = 2.5
foo1 = int(bar)
foo2 = int(-bar)
```

What are the values of `foo1` and `foo2`?

- `foo1` $= \quad 2.0$
- `foo2` $= -2.0$

Is this `floor` or `ceil`?

Some Important Functions

Question: What happens in Python (and most languages) if you cast a `float` as an `int`?

In [4]:
```
bar = 2.5
foo1 = int(bar)
foo2 = int(-bar)
```

What are the values of `foo1` and `foo2`?

- `foo1` $= \quad 2.0$
- `foo2` $= -2.0$

Is this `floor` or `ceil`? It's **both**! $\texttt{int}(x) = \begin{cases} \lfloor x \rfloor & \text{if } x \geq 0 \\ \lceil x \rceil & \text{if } x < 0 \end{cases}$

Functions Wrap-Up

Big Picture:

- Soon we'll talk about algorithms and their complexity
- Model complexity as functions of size of program inputs
- Ask how fast or slow they grow and decide whether they're efficient or not