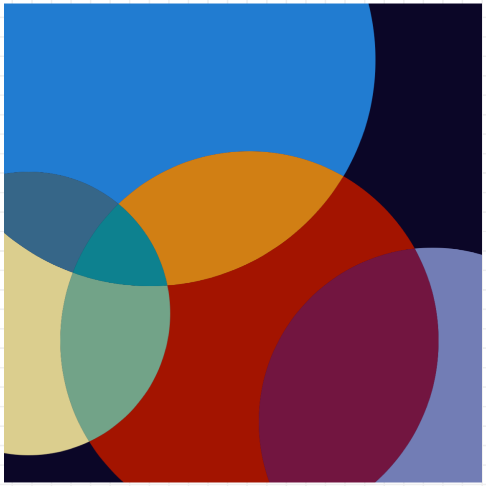


CSCI 2824: Discrete Structures

Lecture 15: Functions. Cardinality.

Reminders

Submissions:

- Homework 5: **Mon 10/7 at noon** – 1 try on Moodle
- Homework 6: **Fri 10/11 at noon** – Gradescope
- **Quizlet 5: due Friday 10/4 at 8pm**

Readings:

- This week Ch. 2 – SETS (2.3-2.4)

What did we do last time?

- doing stuff with sets (proofs and manipulations)

Today:

Functions!

- Things we can do to sets of stuff
- Special kinds of functions
- Special properties functions can have
- Should we fear them?

Functions

Functions

Functions are everywhere in computer science and engineering.

```
In [7]: def Square(x_in):
   ...:
   ...:     x2_out = x_in * x_in
   ...:
   ...:     return (x2_out)
   ...:
```

A function is a routine that takes some kind of input, *does stuff*, and yields some kind of output, as well as possibly some side effects.

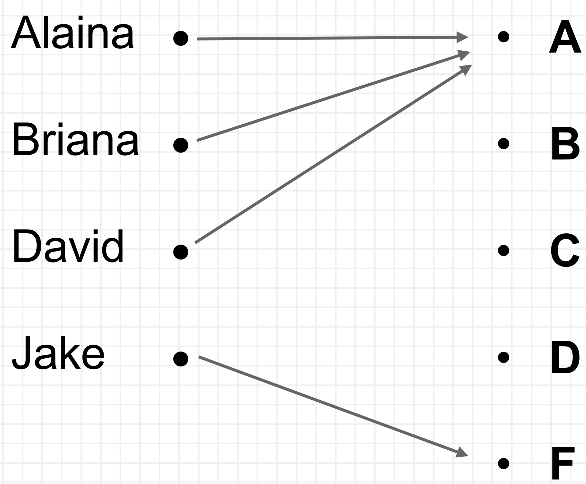
Computer science distinction: **function:** takes inputs, produces outputs
vs. **procedure:** takes inputs, produces side effects (but no outputs)

We will use this narrower definition of a function (inputs ⇒ outputs)

Functions

We will abstract this to the mathematical idea of a function.

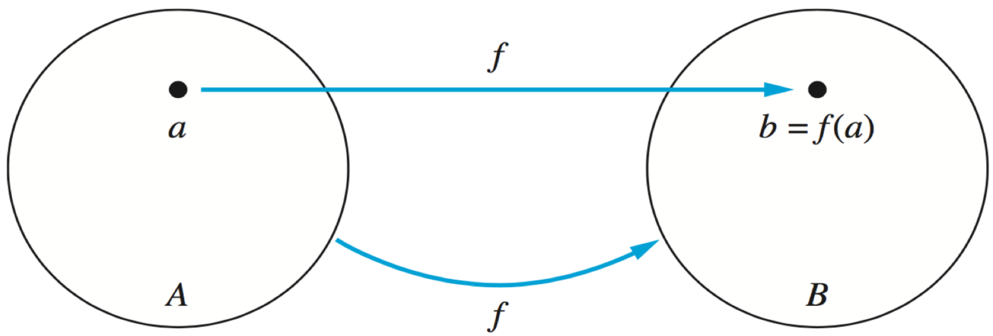
Example: Here is a function that assigns grades to Discrete Structures students.



The grade assignment function takes students {Alaina, Briana, David, Jake, … } 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***.



Definition: Let *A* and *B* be nonempty sets. A **<u>function</u>** *f* from *A* to *B* is an assignment of exactly one element of *B* to each element of *A*.

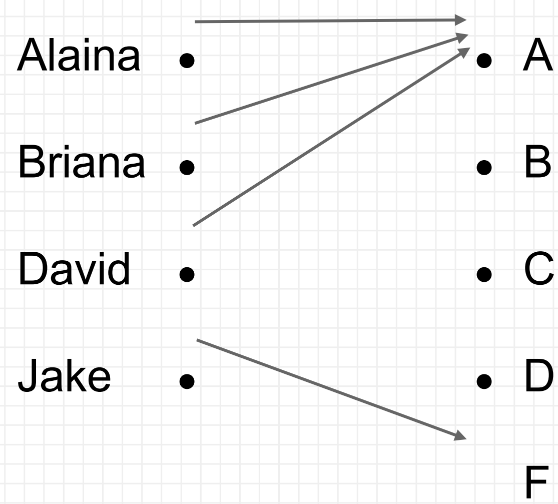
We write $f(a) = b$ to indicate that *b* is the unique element of *B* corresponding to $a \in A$.

If *f* is a function from *A* to *B*, we write $f : A \to B$.

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

Definition: The **codomain** of f is the set of elements that f ***could*** **map to**.

Definition: The **range** of f is the set of elements that f ***actually does*** **map to**.



Here: *domain* =

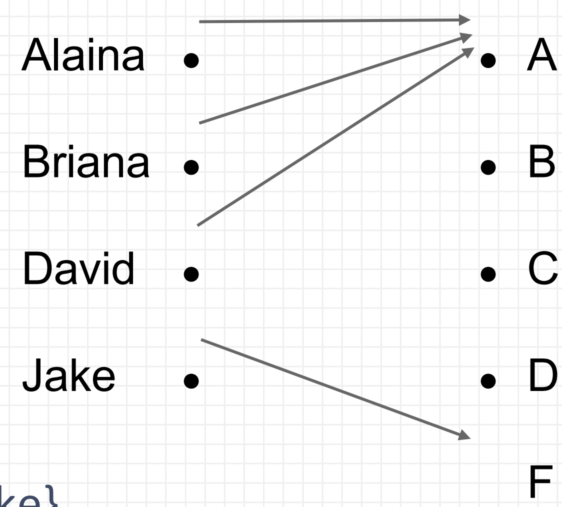
codomain =

range =

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

Definition: The **codomain** of f is the set of elements that f ***could* map to**.

Definition: The **range** of f is the set of elements that f ***actually does* map to**.



Here: *domain* = {Alaina, Briana, David, Jake}
codomain = {A, B, C, D, F}
range = {A, F}

Functions

Consider this Square function again.

```
In [7]: def Square(x_in):
   ...:
   ...:     x2_out = x_in * x_in
   ...:
   ...:     return (x2_out)
   ...:
```

Question: what are its domain, codomain and range? (assuming you are using it for its intended purpose…)

Here: *domain* =

codomain =

range =

Functions

Consider this Square function again.

```
In [7]: def Square(x_in):
   ...:
   ...:     x2_out = x_in * x_in
   ...:
   ...:     return (x2_out)
   ...:
```

Answer: *domain* = set of all real numbers (don't worry about complex)
codomain = set of all non-negative real numbers, plus 0
range = set of all non-negative real numbers, plus 0

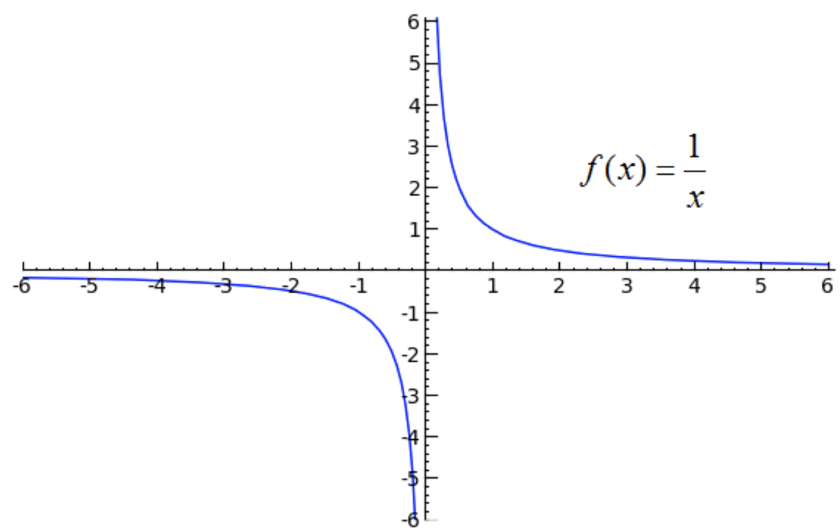
Zero is the **integer** denoted 0 that, when used as a counting number, means that no objects are present. It is the only **integer** (and, in fact, the only real number) that is neither negative nor positive

Functions

The choice of domain affects the nature of the function f

Example: Define $f : \mathbf{R} \rightarrow \mathbf{R}$ by $f(x) = 1/x$

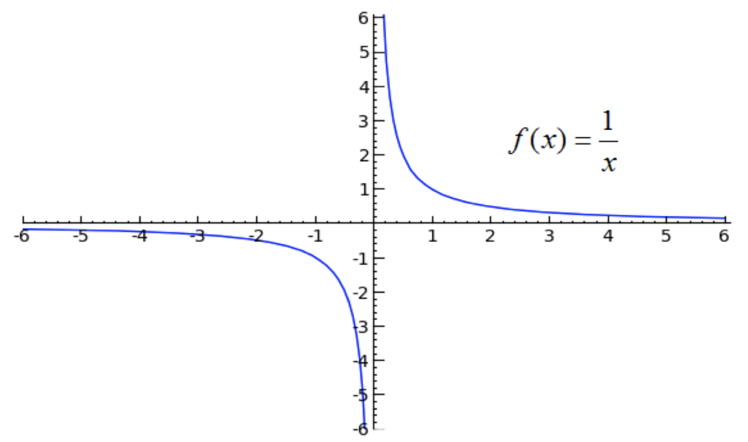
Question: Is this a valid function?



Functions

The choice of domain affects the nature of the function f

Example: Define f : **R** → **R** by $f(x) = 1/x$



Question: Is this a valid function?

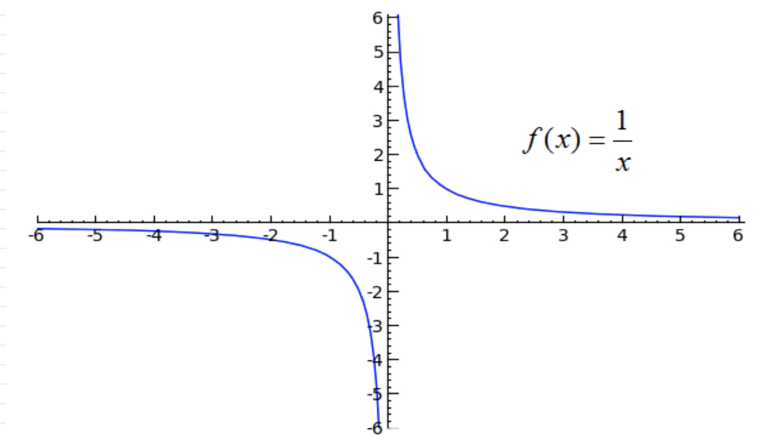
➢ **Answer:** No! Because f doesn't map every element of the domain into the codomain

Question: What if we change it to f : (**R** - {0}) → **R** ?

Functions

The choice of domain affects the nature of the function f

Example: Define $f : \mathbf{R} \rightarrow \mathbf{R}$ by $f(x) = 1/x$



Question: Is this a valid function?

Answer: No! Because f doesn't map every element of the domain into the codomain

Question: What if we change it to $f : (\mathbf{R} - \{0\}) \rightarrow \mathbf{R}$?

Answer: Yes! Now f maps every element of the domain to something in the codomain, so it's a function.

Functions

Consider this super unnecessary Python function to add two numbers

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Question: What are the Add function's domain, codomain and range?

Functions

Consider this super unnecessary Python function to add two numbers

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Question: What are the Add function's domain, codomain and range?

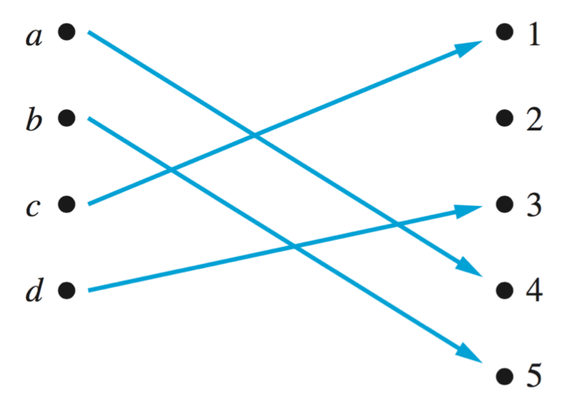
Answer: Assuming we are not dealing with complex numbers, can think of Add as

$$\text{Add} : (\mathbf{R} \times \mathbf{R}) \rightarrow \mathbf{R}$$

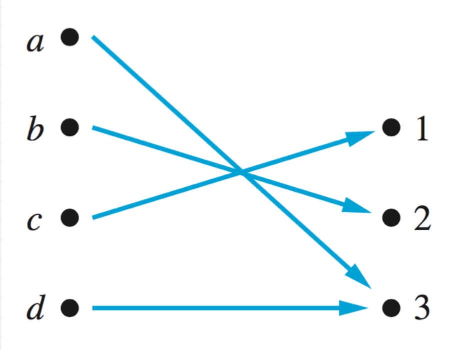
- *Domain* is the set of ordered pairs of real numbers, (x, y)
- *Codomain* is the set of real numbers
- *Range* is also the set of real numbers

One-to-one and *Onto* Functions

Some functions never assign the same value in the range to more than one domain element. These functions are called **one-to-one** functions.



This **is** a one-to-one function



This **is not** a one-to-one function

Definition: A function *f* is **one-to-one**, or *injective*, 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 and *Onto* Functions

Example: Prove that $f(n) = n^3$ is one-to-one. (implicit: $f : \mathbf{R} \to \mathbf{R}$)

- **Strategy:** Assume $f(n) = f(m)$, and show it must be the case that $n = m$.

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^3$ is one-to-one. (implicit: $f : \mathbf{R} \rightarrow \mathbf{R}$)

- **Strategy:** Assume $f(n) = f(m)$, and show it must be the case that $n = m$.

Proof:

Assume f(n) is not one-to-one

(two elements of the domain map to the same element of the codomain)

Then for $n, m \in \mathbf{R}$, $f(n) = f(m)$

$\Rightarrow n^3 = m^3$

$\Rightarrow$ take the cube root of both sides to find $n = m$

$\Rightarrow$ so f must be one-to-one

$\square$

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^2$ is **not** one-to-one. (implicit: $f : \mathbf{R} \rightarrow \mathbf{R}$)

- **Strategy:** Show that there is at least **one pair** of numbers that map to the same place.

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^2$ is **not** one-to-one. (implicit: $f : \mathbf{R} \rightarrow \mathbf{R}$)

- **Strategy:** Show that there is at least **one pair** of numbers that map to the same place.

Proof:

Take $n = 2$ and $m = -2$.

$\Rightarrow$ Then $f(n) = f(m) = 4$

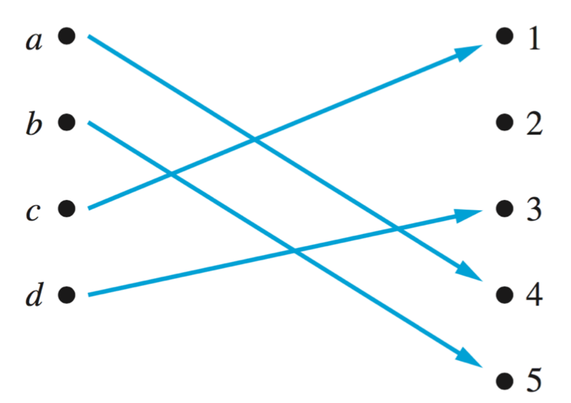
$\Rightarrow$ [at least] two numbers in the domain map to the same element in the range

$\Rightarrow$ so f must not be one-to-one

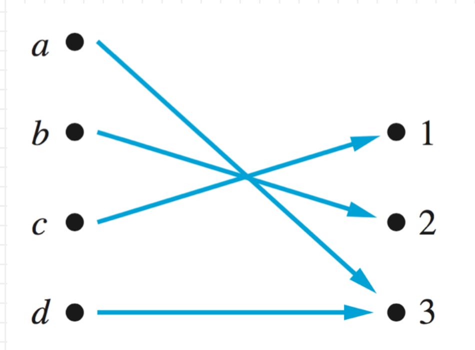
$\square$

One-to-one and *Onto* Functions

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



This **is not** an onto function



This **is** an onto function

Definition: A function *f* is **onto**, or *surjective*, if and only if for every element $b \in B$, there is an element $a \in A$ such that $f(a) = b$.

One-to-one and *Onto* Functions

Example: Prove that our Add function is onto,
where *Add* : (**Z** ✕ **Z**) → **Z** is:

$$Add(m, n) = m + n$$

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Strategy: Show that any arbitrary element of the codomain has at least one element of the domain that maps to it.

One-to-one and *Onto* Functions

Example: Prove that our Add function is onto,
where $Add : (\mathbf{Z} \times \mathbf{Z}) \rightarrow \mathbf{Z}$ is:

$$Add(m, n) = m + n$$

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Strategy: Show that any arbitrary element of the codomain has at least one element of the domain that maps to it.

Proof:

S'pose b is any integer (i.e., an element of the codomain)

$\Rightarrow$ Then we need to find integers m and n such that $m + n = b$.

$\Rightarrow$ Let n be any integer.

$\Rightarrow$ Let $m = b - n$ $\square$

One-to-one and *Onto* Functions

Example: Prove that our Add function is onto, where $Add : (\mathbf{Z} \times \mathbf{Z}) \rightarrow \mathbf{Z}$ is:

$$Add(m, n) = m + n$$

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Question: Is $Add(m, n) = m + n$ one-to-one?

One-to-one and *Onto* Functions

Example: Prove that our Add function is onto, where $Add : (\mathbf{Z} \times \mathbf{Z}) \rightarrow \mathbf{Z}$ is:

$$Add(m, n) = m + n$$

```
In [14]: # a function for adding
    ...: def Add(x_in, y_in):
    ...:
    ...:     sum_out = x_in + y_in
    ...:
    ...:     return (sum_out)
```

Question: Is $Add(m, n) = m + n$ one-to-one?

Answer: Nope.

- S'pose $f(m, n) = m + n = 3$.
- There are LOTS of m and n that add up to 3.

$$f(1,2) = f(3,0) = f(-1000, 1003) = \ldots = 3$$

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^2$ where $f : \mathbf{Z} \rightarrow \mathbf{Z}$ is **not** onto.

Strategy: Show that there is at least one element of the codomain that is not mapped to.

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^2$ where $f : \mathbf{Z} \rightarrow \mathbf{Z}$ is not onto.

Strategy: Show that there is at least one element of the codomain that is not mapped to.

Proof: Any negative number is not mapped to.

- e.g., nothing maps to -1 (there is no $n \in \mathbf{Z}$ s.t. $f(n) = n^2 = -1$)

Let's make it more interesting: What if we redefine it to $f : \mathbf{Z} \rightarrow \mathbf{N}$?

One-to-one and *Onto* Functions

Example: Prove that $f(n) = n^2$ where $f : \mathbf{Z} \rightarrow \mathbf{Z}$ is not onto.

Strategy: Show that there is at least one element of the codomain that is not mapped to.

Proof: Any negative number is not mapped to.

- e.g., nothing maps to -1 (there is no $n \in \mathbf{Z}$ s.t. $f(n) = n^2 = -1$)

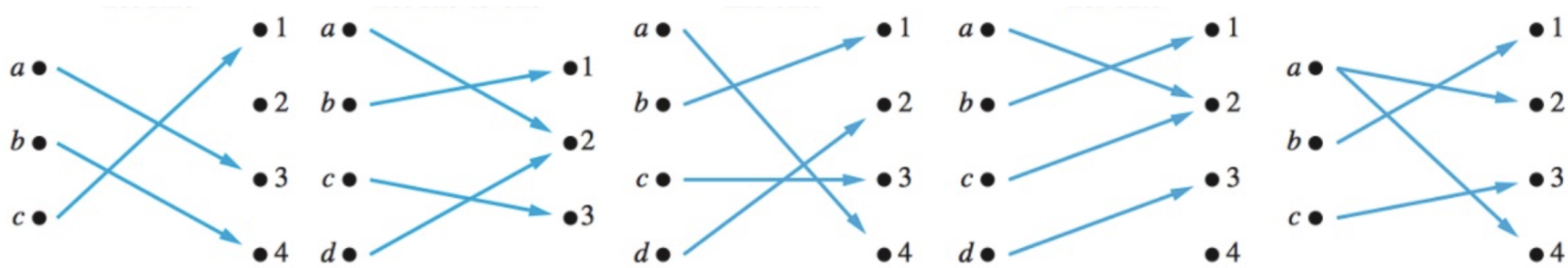
Let's make it more interesting: What if we redefine it to $f : \mathbf{Z} \rightarrow \mathbf{N}$?

Answer: Still no - not every natural number is a perfect square.

- e.g., nothing maps to 5 (there is no $n \in \mathbf{Z}$ s.t. $f(n) = n^2 = 5$)

One-to-one and *Onto* Functions

Example: Classify these functions as one-to-one (1-1), onto, both or neither.



Inverse Functions

Nice things happen when a function is both 1-1 and onto.

- This is called a ***bijective*** function.

  The function can be called a ***bijection*** (special kind of function that is 1-1 and onto).