Discrete Algebraic Structures

WiSe 2025/2026

Prof. Dr. Antoine Wiehe
Research Group for Theoretical Computer Science

Meetings:

- Lectures: Monday 15:00-16:30
- Lecture hall exercises:
  - Wednesday 08:30-09:15
  - Recap of the lecture, exercises to practice the methods
- Tutorials: Starting this week.
  - Choose a group on Stud.IP (in Participants → Groups) and stick to that group.
  - Deeper look into the content of the lecture.
- All learning material available on StudIP: slides, lecture notes, exercise sheets, recordings of the lectures

Exam and Bonus points:

- Tests every week at the beginning of the tutorial
- Get up to 3 points per week ($3 \times 12 = 36$ points)
- Need ≥ 16 points to validate the *Studienleistung*
- Extra points are given as bonus for the exam (up to 10% of the grade)

Test points	Exam points	End result
< 16		module not validated
$16 + y$	x	$x + y$
36	80/200 (fail)	100/200 (pass)

Things you will do in your career:

- Develop algorithms
- Solve problems
- Think and communicate with others about complex/abstract concepts

Things you will do in your career:

- Develop algorithms
- Solve problems
- Think and communicate with others about complex/abstract concepts

From the CS *Modulhandbuch*:

"*Ferner werden methodische Grundlagen erworben, um sich stets an neue berufliche Entwicklungen und Innovationen anzupassen. Daher sollten die Absolventen und Absolventinnen in nahezu allen Branchen eine verantwortungsvolle Tätigkeit finden können.*"

From the DS *Modulhandbuch*:

"*Der Data Scientist arbeitet forschungsnah und ist somit immer auf dem neuesten Stand über neue Entwicklungen in dem sich rasant entwickelnden Feld.*"

Things you will do in your career:

- Develop algorithms
- Solve problems
- Think and communicate with others about complex/abstract concepts

From the CS *Modulhandbuch*:
"*Ferner werden methodische Grundlagen erworben, um sich stets an neue berufliche Entwicklungen und Innovationen anzupassen. Daher sollten die Absolventen und Absolventinnen in nahezu allen Branchen eine verantwortungsvolle Tätigkeit finden können.*"

From the DS *Modulhandbuch*:
"*Der Data Scientist arbeitet forschungsnah und ist somit immer auf dem neuesten Stand über neue Entwicklungen in dem sich rasant entwickelnden Feld.*"

- Maths is necessary for keeping up with the state of the art
- English is as well (research papers are published in English, no matter where the research is done)

Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Shafi Goldwasser
Turing award 2012

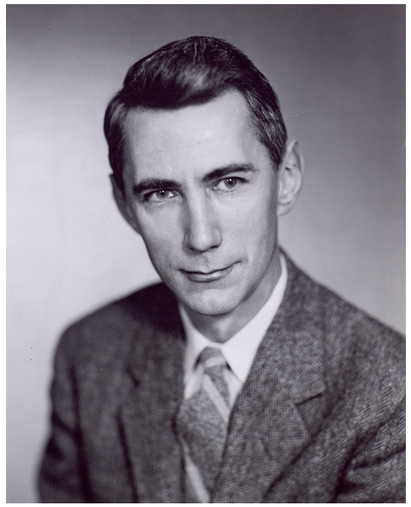
Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Shafi Goldwasser
Turing award 2012



Claude Shannon
Big guy

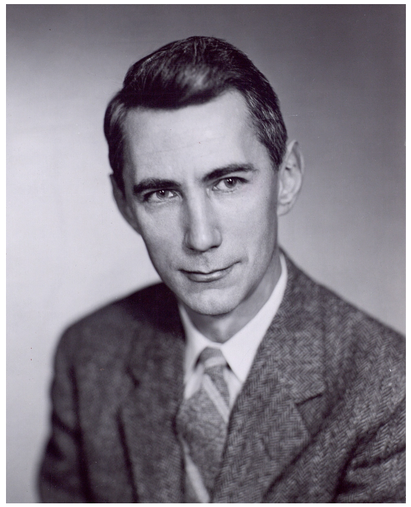
Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Shafi Goldwasser
Turing award 2012



Claude Shannon
Big guy

Frances Allen
Turing award 2006

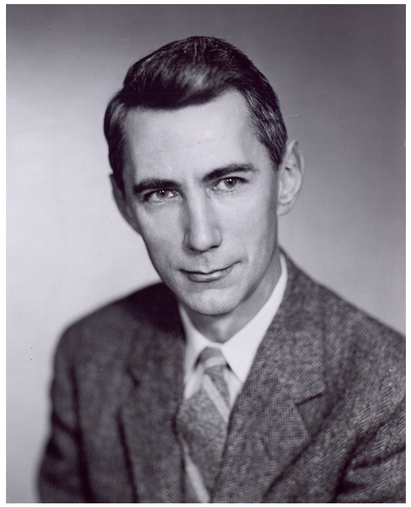
Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

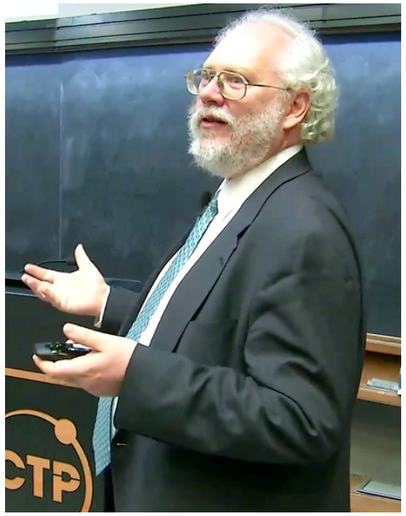
- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Shafi Goldwasser
Turing award 2012



Claude Shannon
Big guy

Frances Allen
Turing award 2006



Peter Shor
Shannon award 2025

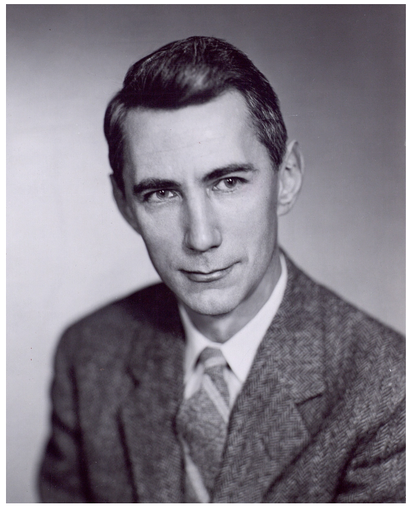
Two types of mathematics:

- Study of “continuous” things: real numbers, functions, quantities that can change smoothly
- Study of “discrete” things: natural numbers, things that can only take values changing in steps

Discrete maths is the mathematics of computers (and compute science):

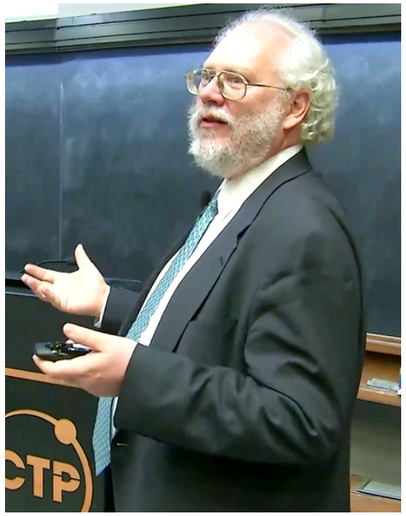
- In a computer, only two things exist: 0 and 1 and finite combinations of 0s and 1
  So π is not a computer thing
- Innovations in encryption, networks, error mitigation, machine learning... can only be obtained by people who have the appropriate mathematical knowledge

Shafi Goldwasser
Turing award 2012



Claude Shannon
Big guy

Frances Allen
Turing award 2006



Peter Shor
Shannon award 2025

- No mathematical knowledge is expected
- Critical thinking
- That's it

You:

- dedicate at least 8–10 hours per week to the course
  - Lectures: 1.5 hours
  - Lecture hall exercises: 0.75 hour
  - Tutorials: 1.5 hours
  - All the rest: read the material, do exercises, talk with your study partner
- find somebody to study with
- are active and use the material we give you: lecture notes, reference books, supplemental exercises
- will reach out to your tutors and use the consultation hours if you are falling behind

We (me + tutors):

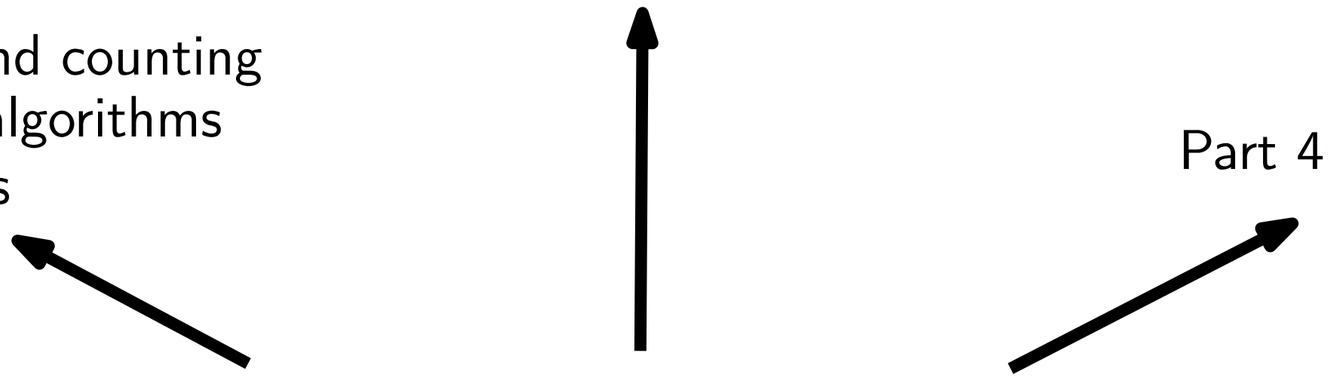
- prepare exercises and solutions for you
- are available for taking care of you if you have difficulties
- present on the forum to answer questions
- will do what we can so that you succeed

Common main objective: your success

Part 5: Algebra
- Cryptography
- Error correction
- Quantum Computing

Part 3: Combinatorics and counting
- Runtime analysis of algorithms
- Basics of probabilities

Part 4: basic number theory



Part 2: Graphs and relations
- Representing data and relationships between things
- Important basis for the rest of your studies (in particular "Algorithms and Data Structures")

Part 1: introduction to mathematics
- Sets
- Basics in logic and proofs

Many everyday things are built on mathematics:

- Google Maps
- Online banking
- QR codes
- Recommendations in Netflix/...
- Large Language Models
- Video games graphics / CGI in movies
- ...

Many everyday things are built on mathematics:

- Google Maps
- Online banking
- QR codes
- Recommendations in Netflix/...
- Large Language Models
- Video games graphics / CGI in movies
- ...

Like all other things, you need to start with the small things before you can get to the top

Many everyday things are built on mathematics:

- Google Maps
- Online banking
- QR codes
- Recommendations in Netflix/...
- Large Language Models
- Video games graphics / CGI in movies
- ...

Like all other things, you need to start with the small things before you can get to the top

Probably not the right attitude!

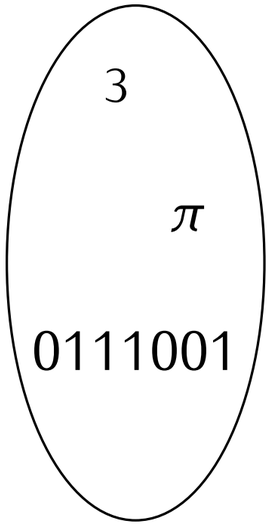
Part 1
Introduction to mathematics

Definition. A set is an unordered bag containing distinct objects.

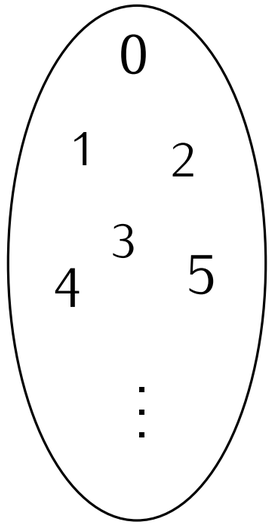
- unordered: no "first" or "last" element in the bag
- distinct: something is either in or out; cannot be there multiple times

Definition. A set is an unordered bag containing distinct objects.

- unordered: no "first" or "last" element in the bag
- distinct: something is either in or out; cannot be there multiple times



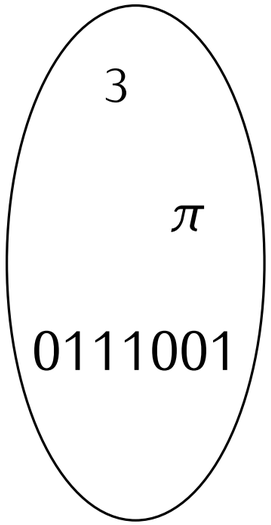
A set with 3 elements



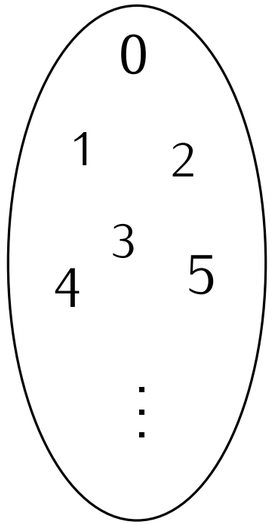
A set with infinitely many elements

Definition. A set is an unordered bag containing distinct objects.

- unordered: no "first" or "last" element in the bag
- distinct: something is either in or out; cannot be there multiple times



A set with 3 elements



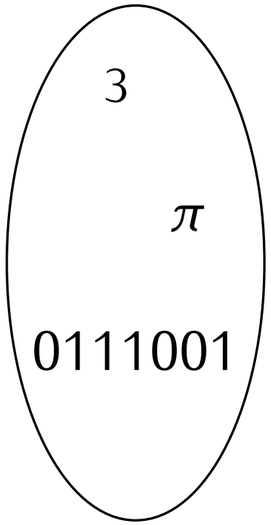
A set with infinitely many elements

Instead of drawing potatoes, we use the following notation:

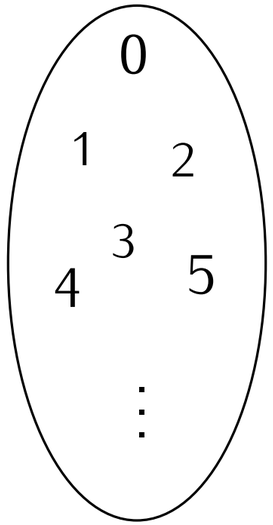
Notation. $\{3, \pi, 0111001\}, \{0, 1, 2, 3, 4, 5, \dots\}$

Definition. A set is an unordered bag containing distinct objects.

- unordered: no "first" or "last" element in the bag
- distinct: something is either in or out; cannot be there multiple times



A set with 3 elements



A set with infinitely many elements

Instead of drawing potatoes, we use the following notation:

Notation. $\{3, \pi, 0111001\}, \{0, 1, 2, 3, 4, 5, \dots\}$

This notation does not care about multiple elements and order:

$$\{3, 3, 1, 2\} = \{1, 2, 3\} = \{1, 1, 2, 2, 3, 3\}$$

What is your relationship to programming?

- don't know anything about it
- I know (at least a little) C
- I know (at least a little) Python
- I know (at least a little) another programming language

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"

```
S = {3,4,5}
print(5 in S) # True
print(2 in S) # False
```

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"

```
S = {3,4,5}
print(5 in S) # True
print(2 in S) # False
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(S.issubset(S)) # True
print(T.issubset(S)) # False
print(U.issubset(S)) # True
print(x.issubset(S)) # ??
```

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"

```
S = {3,4,5}
print(5 in S) # True
print(2 in S) # False
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"

- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(S.issubset(S)) # True
print(T.issubset(S)) # False
print(U.issubset(S)) # True
print(x.issubset(S)) # ??
```

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"

```
S = {3,4,5}
print(5 in S) # True
print(2 in S) # False
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"
- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(S.issubset(S)) # True
print(T.issubset(S)) # False
print(U.issubset(S)) # True
print(x.issubset(S)) # ??
```

- $\emptyset$: the only set that contains absolutely nothing

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"
- $y \notin S$: "y is not in S"

```
S = {3,4,5}
print(5 not in S) # False
print(2 not in S) # True
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"

- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(S.issubset(S)) # True
print(T.issubset(S)) # False
print(U.issubset(S)) # True
print(x.issubset(S)) # ??
```

- $\emptyset$: the only set that contains absolutely nothing

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"
- $y \notin S$: "y is not in S"

```
S = {3,4,5}
print(5 not in S) # False
print(2 not in S) # True
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"
- $T \nsubseteq S$: some element of T is not in S

- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(not S.issubset(S)) # False
print(not T.issubset(U)) # True
print(not U.issubset(S)) # False
```

- $\emptyset$: the only set that contains absolutely nothing

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"
- $y \notin S$: "y is not in S"

```
S = {3,4,5}
print(5 not in S) # False
print(2 not in S) # True
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"
- $T \nsubseteq S$: some element of T is not in S

- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!
- $S \neq T$: something is in exactly one of S and T

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(not S.issubset(S)) # False
print(not T.issubset(U)) # True
print(not U.issubset(S)) # False
```

- $\emptyset$: the only set that contains absolutely nothing

Definition. A set is an unordered bag containing distinct objects.

- $y \in S$: "y is in S", "y is an element of S", "S contains y"
- $y \notin S$: "y is not in S"

```
S = {3,4,5}
print(5 not in S) # False
print(2 not in S) # True
```

- $T \subseteq S$: every element of T is also an element of S, "T is a subset of S"
- $T \not\subseteq S$: some element of T is not in S

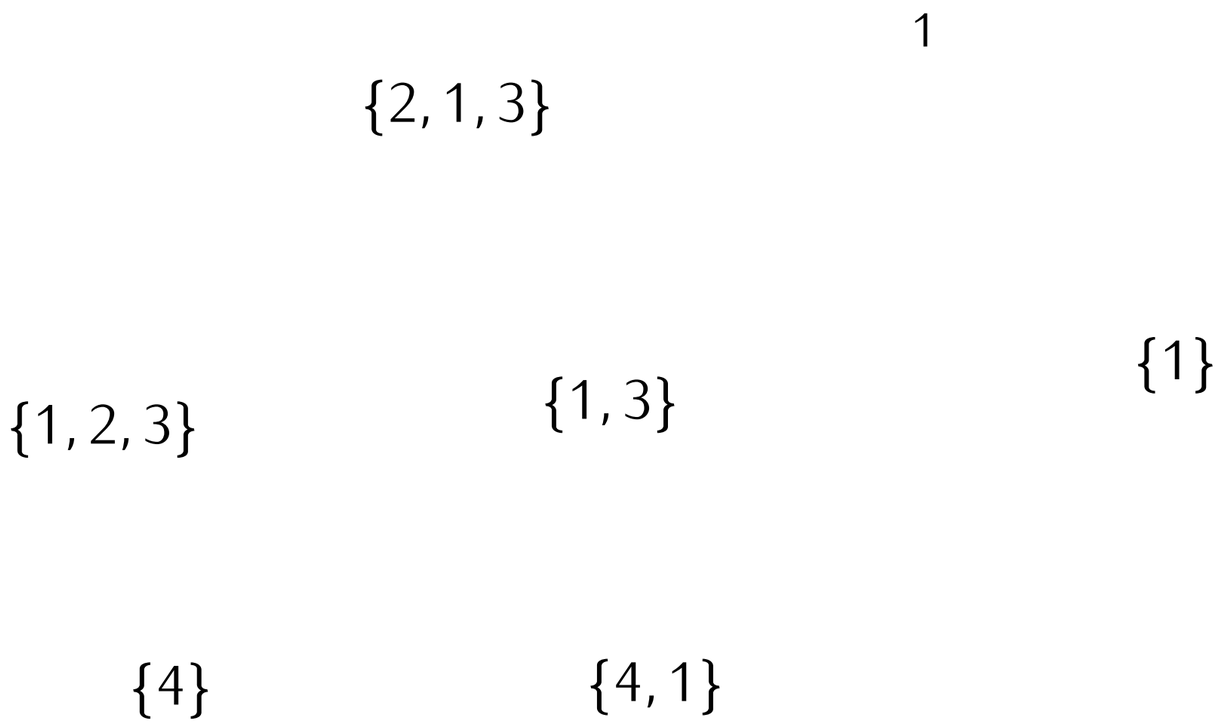
- $S = T$: S and T contain exactly the same elements
  Same as $S \subseteq T$ and $T \subseteq S$ being true at the same time!
- $S \neq T$: something is in exactly one of S and T

```
S = {3,4,5}
T = {2,4}
U = {3}
x = 3
print(not S.issubset(S)) # False
print(not T.issubset(U)) # True
print(not U.issubset(S)) # False
```

- $\emptyset$: the only set that contains absolutely nothing

Remark. Note the following:
- For each y and S: either $y \in S$ or $y \notin S$ (any thing is either in or out of S)
- A set can have other sets as elements: $\{0, 1, \{0, 1\}, 2\}$ is a set with elements

Draw an arrow from x to y if $x \subseteq y$:



Notation. We use the following notation:

- $\mathbb{N}_0 \overset{\text{def}}{=} \{0, 1, 2, 3, \dots\}$ natural numbers
- $\mathbb{N} \overset{\text{def}}{=} \{1, 2, 3, \dots\}$
- $\mathbb{Z} \overset{\text{def}}{=} \{0, 1, -1, 2, -2, \dots\}$ integers
- $\mathbb{Q} \overset{\text{def}}{=} \{0, 1, \frac{1}{2}, -\frac{1}{5}, \frac{7}{12}, \dots\}$ rational numbers
- $\mathbb{R} \overset{\text{def}}{=} \{0, 1, \pi, \sqrt{2}, e, \dots\}$ real numbers

Notation. We use the following notation:

- $\mathbb{N}_0 \overset{\text{def}}{=} \{0, 1, 2, 3, \dots\}$ natural numbers
- $\mathbb{N} \overset{\text{def}}{=} \{1, 2, 3, \dots\}$
- $\mathbb{Z} \overset{\text{def}}{=} \{0, 1, -1, 2, -2, \dots\}$ integers
- $\mathbb{Q} \overset{\text{def}}{=} \{0, 1, \frac{1}{2}, -\frac{1}{5}, \frac{7}{12}, \dots\}$ rational numbers
- $\mathbb{R} \overset{\text{def}}{=} \{0, 1, \pi, \sqrt{2}, e, \dots\}$ real numbers

When a set S is already known, we can create a subset of it by selecting some elements:

$$\{x \in S \mid \text{selection criterion } P(x)\}$$

- $\{x \in \mathbb{N}_0 \mid x \text{ is even}\}$
- $\{x \in \mathbb{R} \mid x^2 = 1\}$
- $\{x \in \mathbb{N}_0 \mid x \text{ is even and is a square of a number}\}$

Notation. We use the following notation:

- $\mathbb{N}_0 \stackrel{\text{def}}{=} \{0, 1, 2, 3, \dots\}$ natural numbers
- $\mathbb{N} \stackrel{\text{def}}{=} \{1, 2, 3, \dots\}$
- $\mathbb{Z} \stackrel{\text{def}}{=} \{0, 1, -1, 2, -2, \dots\}$ integers
- $\mathbb{Q} \stackrel{\text{def}}{=} \{0, 1, \frac{1}{2}, -\frac{1}{5}, \frac{7}{12}, \dots\}$ rational numbers
- $\mathbb{R} \stackrel{\text{def}}{=} \{0, 1, \pi, \sqrt{2}, e, \dots\}$ real numbers

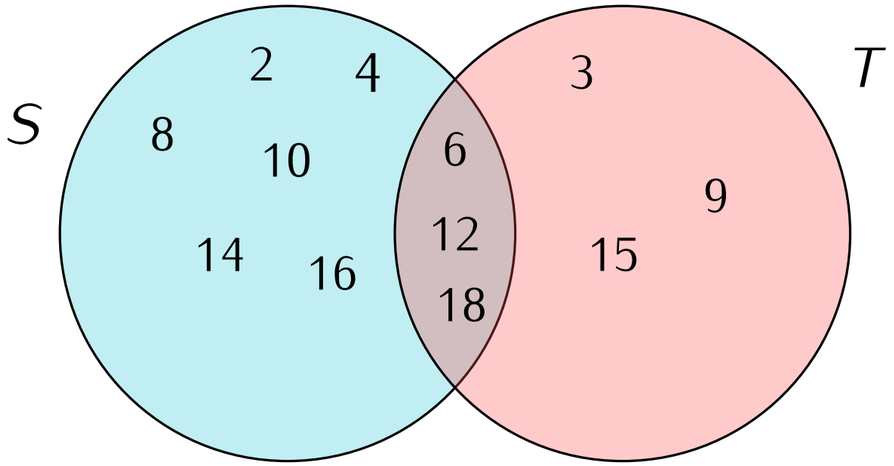
When a set S is already known, we can create a subset of it by selecting some elements:

$$\{x \in S \mid \text{selection criterion } P(x)\}$$

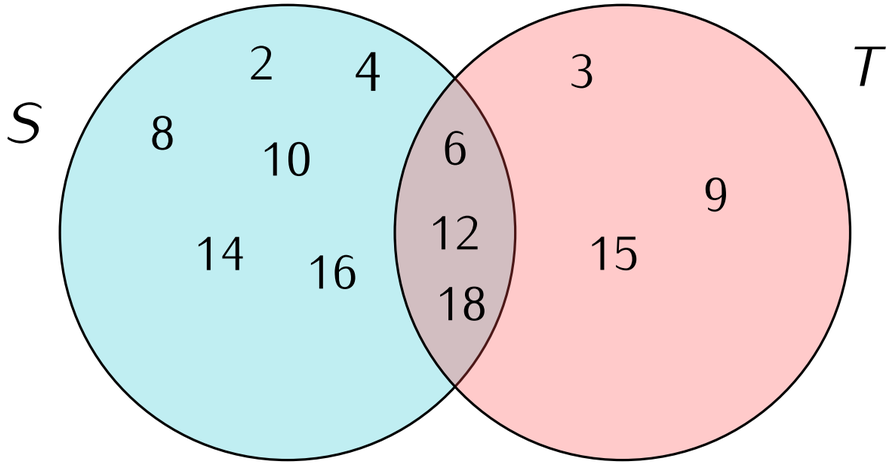
- $\{x \in \mathbb{N}_0 \mid x \text{ is even}\}$
- $\{x \in \mathbb{R} \mid x^2 = 1\}$
- $\{x \in \mathbb{N}_0 \mid x \text{ is even and is a square of a number}\}$

```
def setbuilder(S:set, P) -> set:
    T = set()
    # loop through all elements of S
    for x in S:
        # If x satisfies the criterion
        if P(x) == True:
            T.add(x) # Add it
    return T

{ x for x in S if P(x) } # Same thing
```

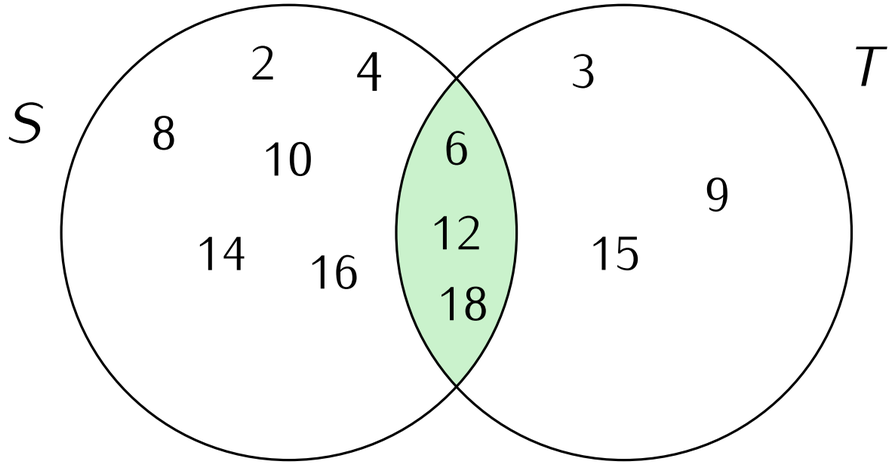
- $S \cup T$: things that are in at least one of S or T



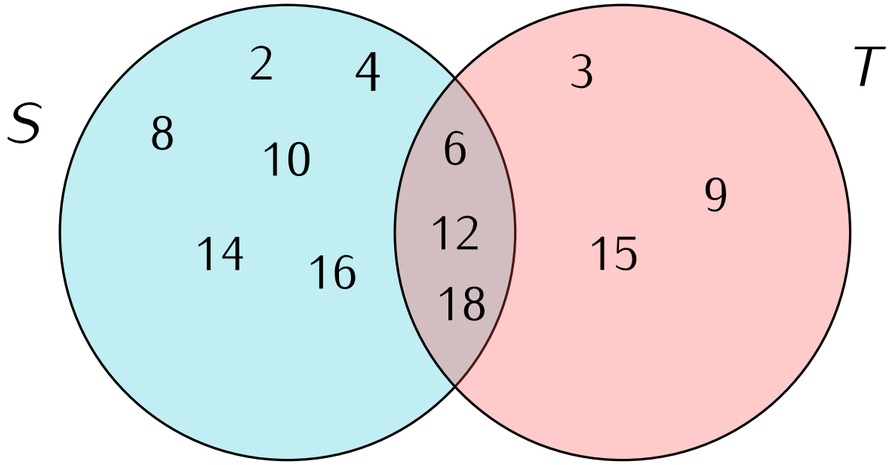
- $S \cup T$: things that are in at least one of S or T



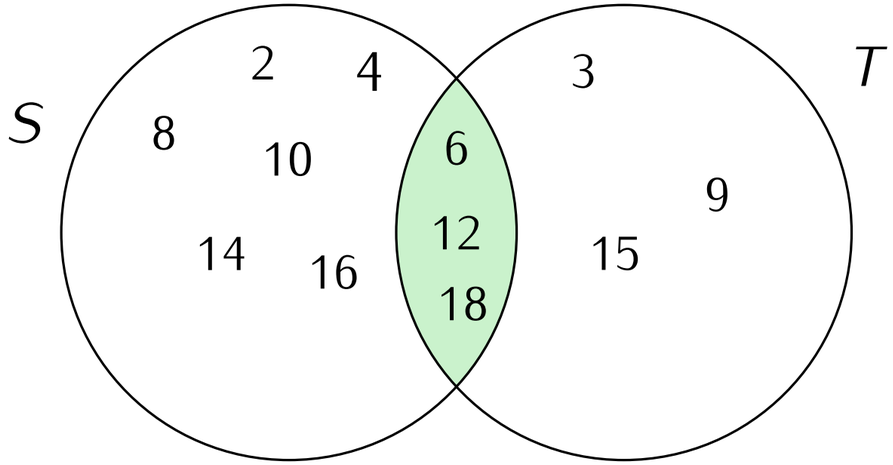
- $S \cap T$: things that are in both S and T



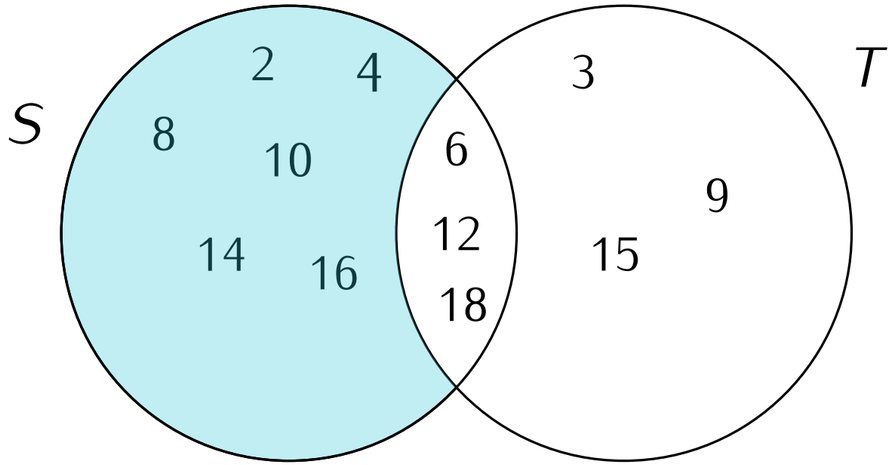
- $S \cup T$: things that are in at least one of S or T



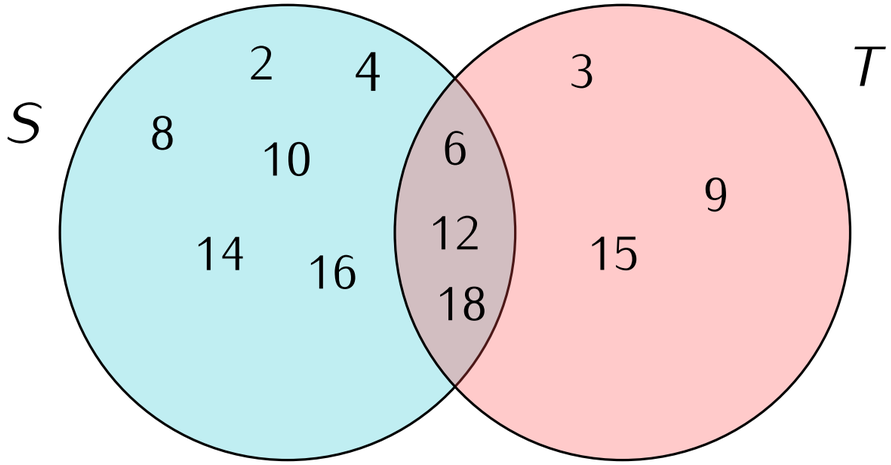
- $S \cap T$: things that are in both S and T



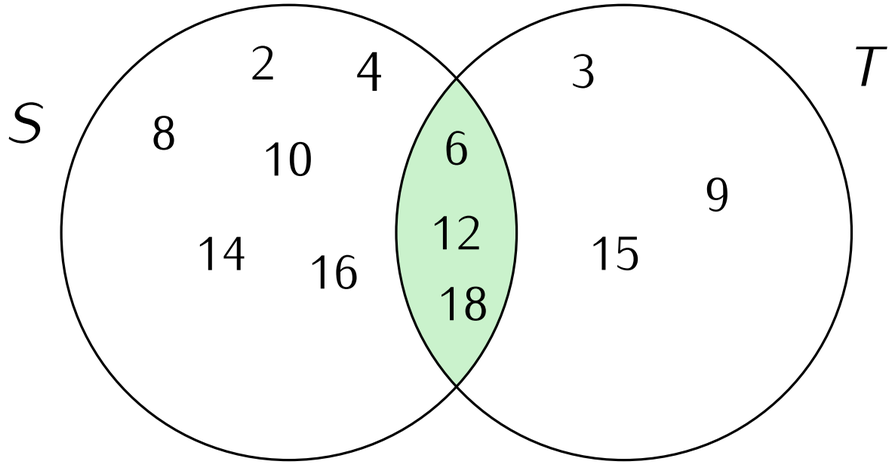
- $S \setminus T$: things in S but not in T



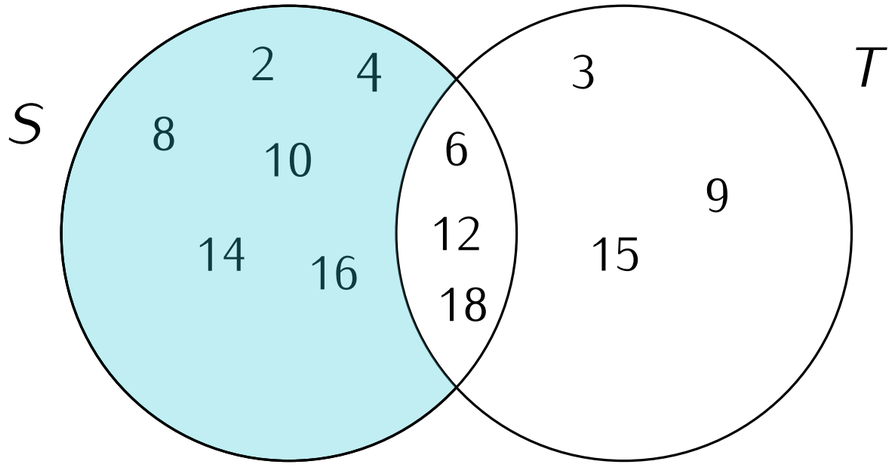
- $S \cup T$: things that are in at least one of S or T



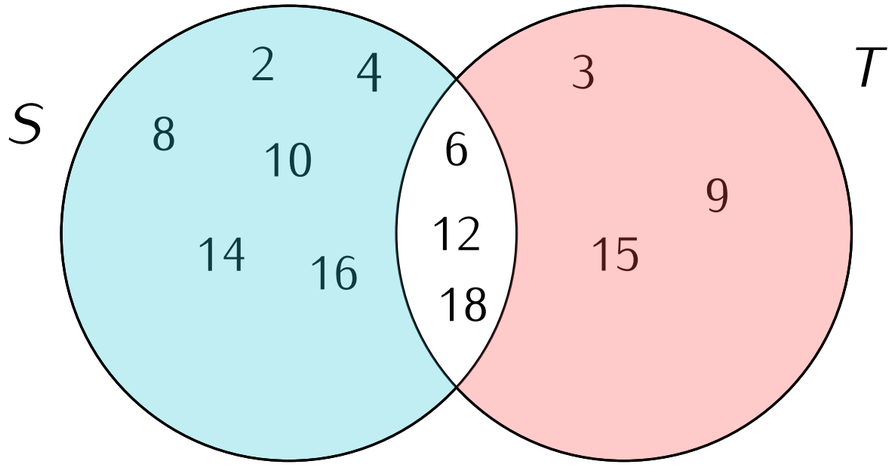
- $S \cap T$: things that are in both S and T



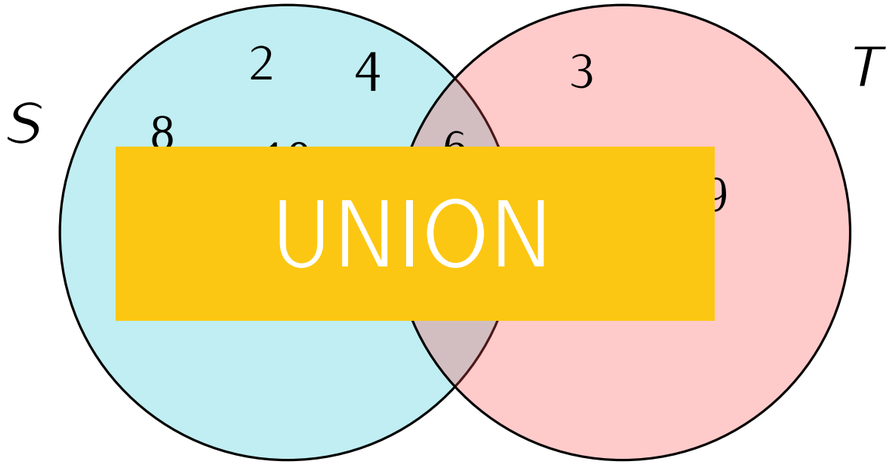
- $S \setminus T$: things in S but not in T



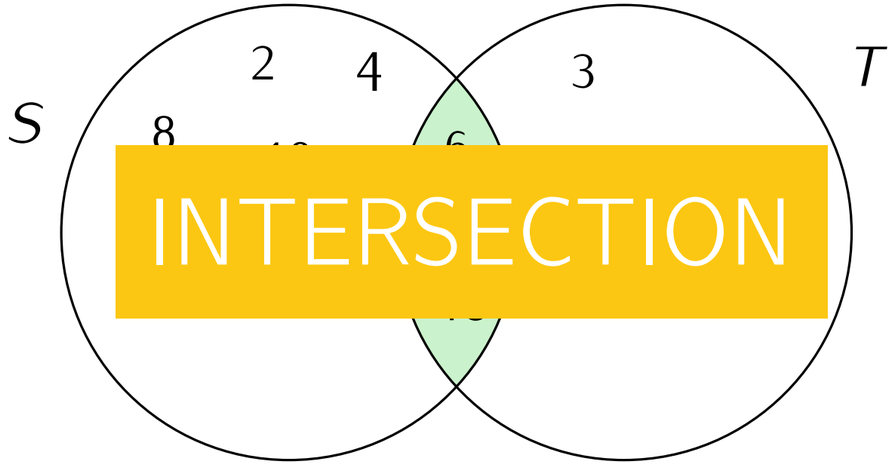
- $S \Delta T$: things in S or T, but not in both



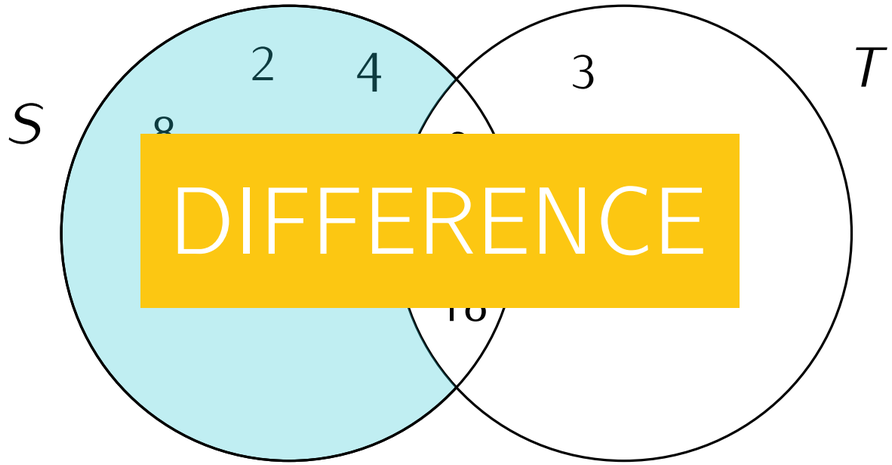
- $S \cup T$: things that are in at least one of S or T



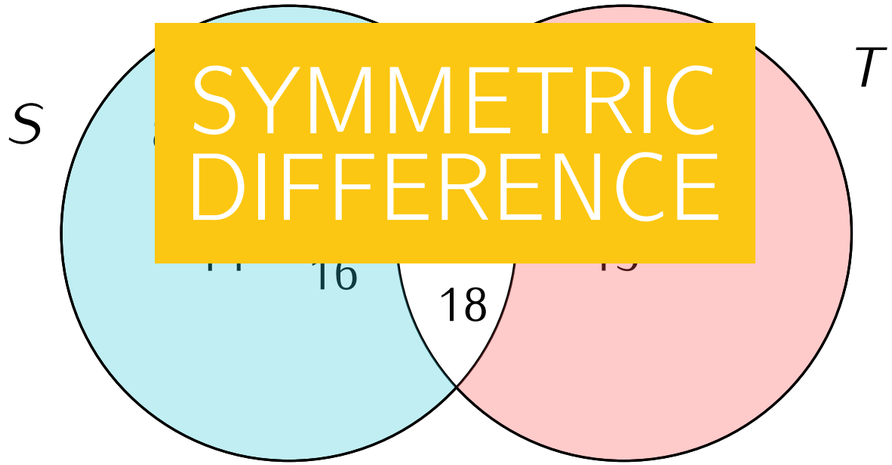
- $S \cap T$: things that are in both S and T



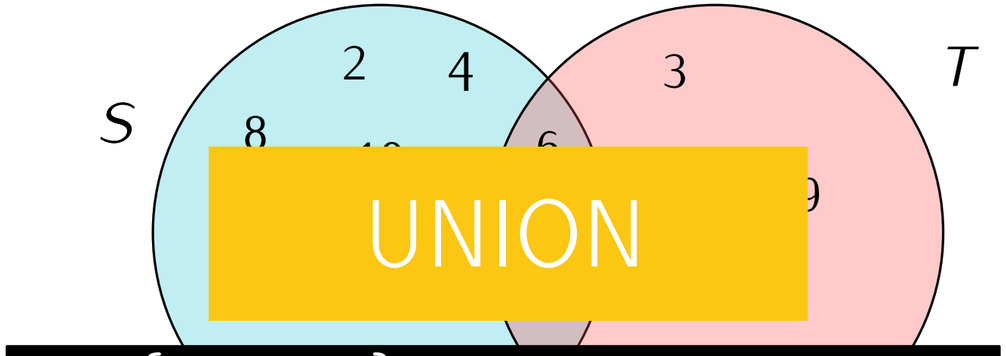
- $S \setminus T$: things in S but not in T



- $S \Delta T$: things in S or T, but not in both

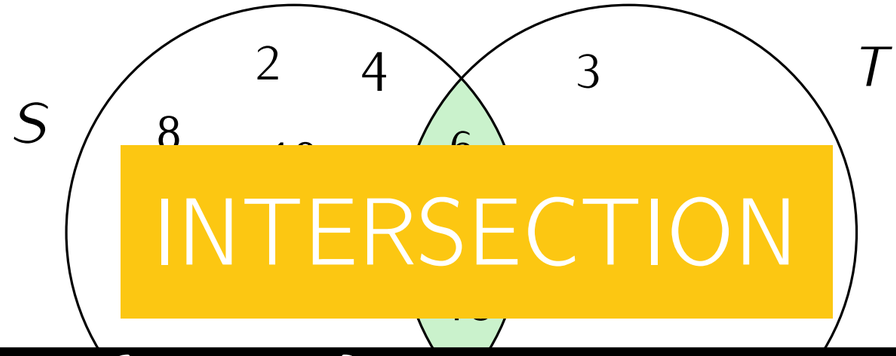


- $S \cup T$: things that are in at least one of S or T



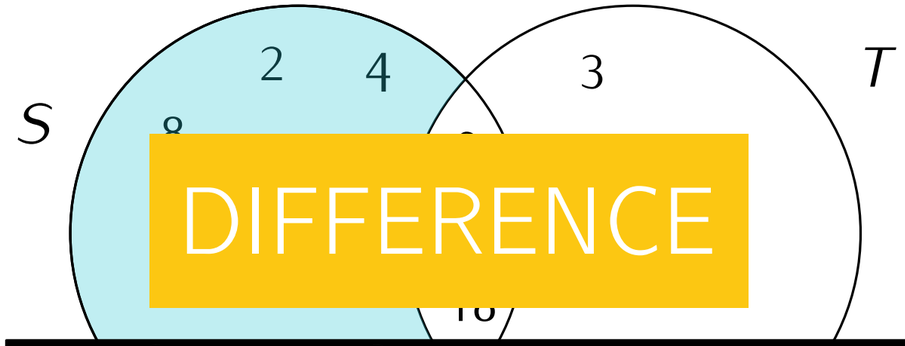

```
S = {3,4,5,6}
T = {2,3,7}
print(S.union(T)) # {3,4,5,6,2,7}
```

- $S \cap T$: things that are in both S and T



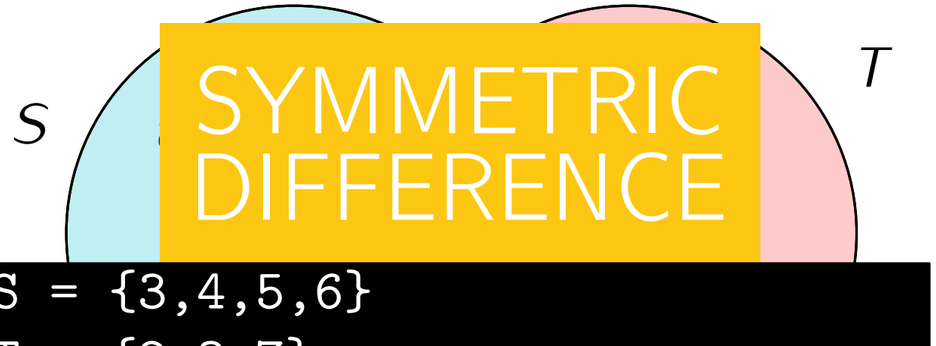

```
S = {3,4,5,6}
T = {2,3,7}
print(S.intersection(T)) # {3}
```

- $S \setminus T$: things in S but not in T




```
S = {3,4,5,6}
T = {2,3,7}
print(S.difference(T)) # {4,5,6}
```

- $S \Delta T$: things in S or T, but not in both




```
S = {3,4,5,6}
T = {2,3,7}
# {4,5,6,2,7}
print(S.symmetric_difference(T))
```

- $S \cup T$: things that are in at least one of S or T
- $S \cap T$: things that are in both S and T
- $S \setminus T$: things in S but not in T
- $S \Delta T$: things in S or T, but not in both

If $S = \{1, 2, \{1, 3\}\}$ and $T = \{1, 3\}$, what is $S \cap T$?

- $\{1, 2, 3\}$
- $\{1, \{1\}\}$
- $\{1, 3\}$
- $\{1\}$

- $S \cup T$: things that are in at least one of S or T
- $S \cap T$: things that are in both S and T
- $S \setminus T$: things in S but not in T
- $S \Delta T$: things in S or T, but not in both

If $S = \{1, 2, \{1, 3\}\}$ and $T = \{1, 3\}$, what is $S \setminus T$?

- $\{1, 2, 3\}$
- $\{1, 2\}$
- $\{2, \{1, 3\}\}$
- $\{2\}$

- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

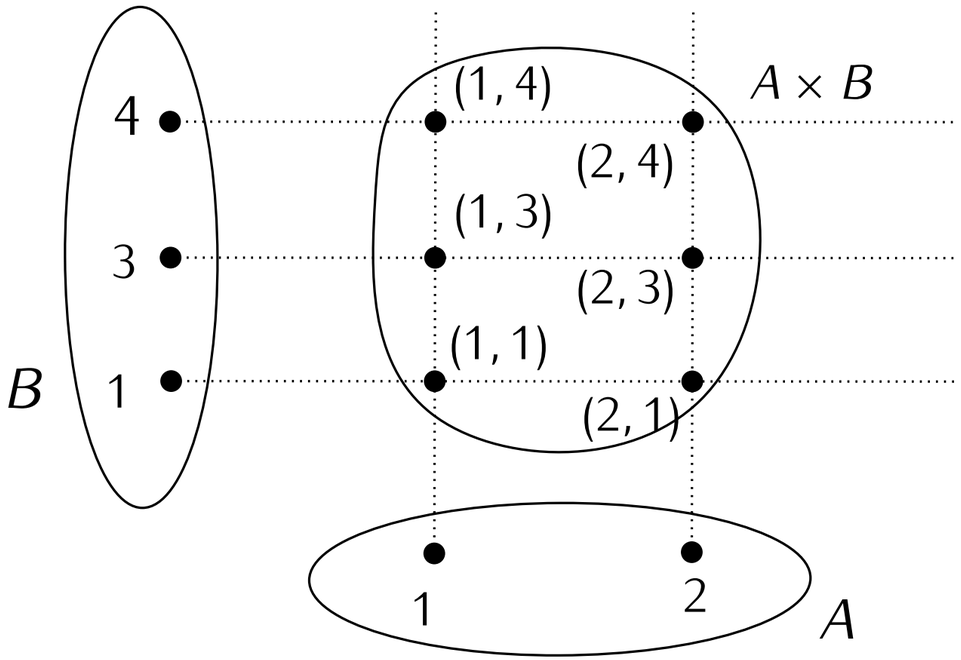
Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

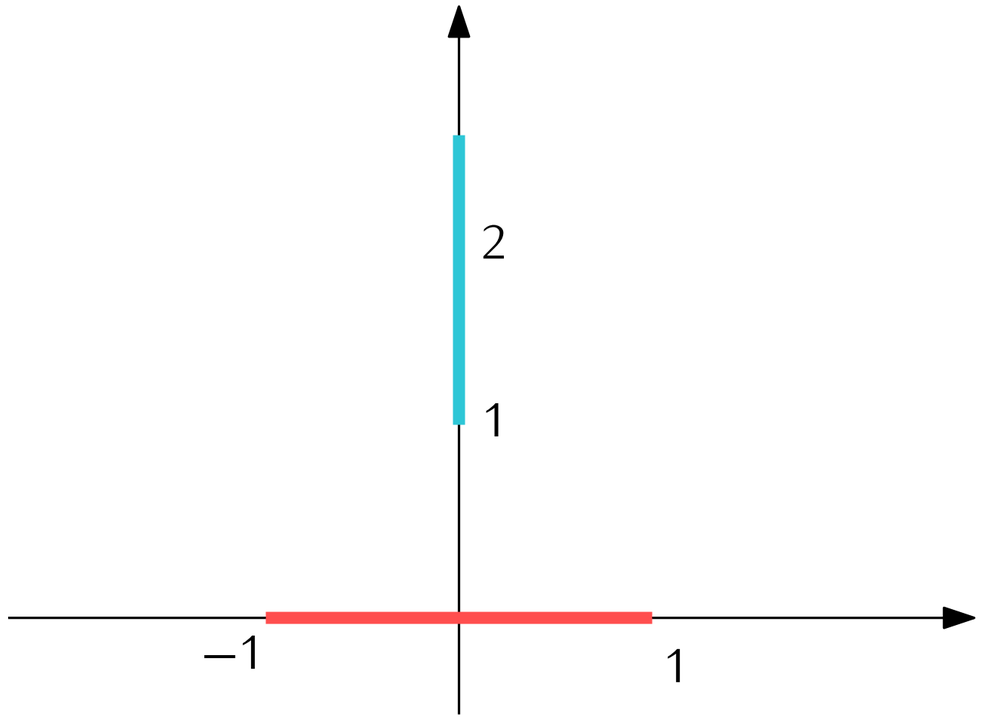
- $A = \{1, 2\}, B = \{1, 3, 4\}$

$A \times B = \{(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4)\}$



- $A = [-1, 1], B = [1, 2.5]$

$A \times B$: all points (x, y) with $x \in [-1, 1], y \in [1, 2.5]$

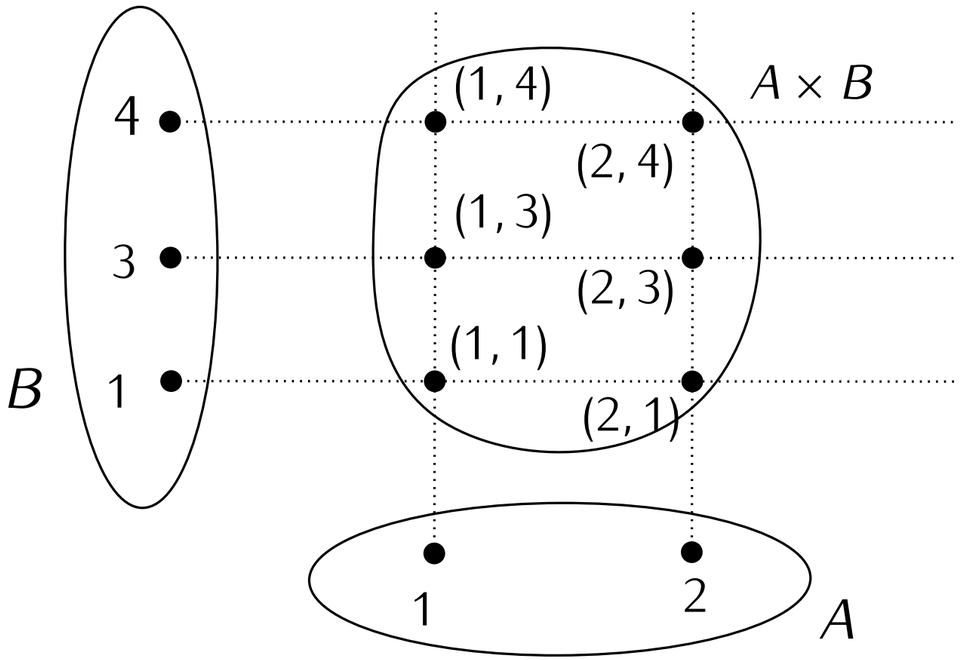


- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

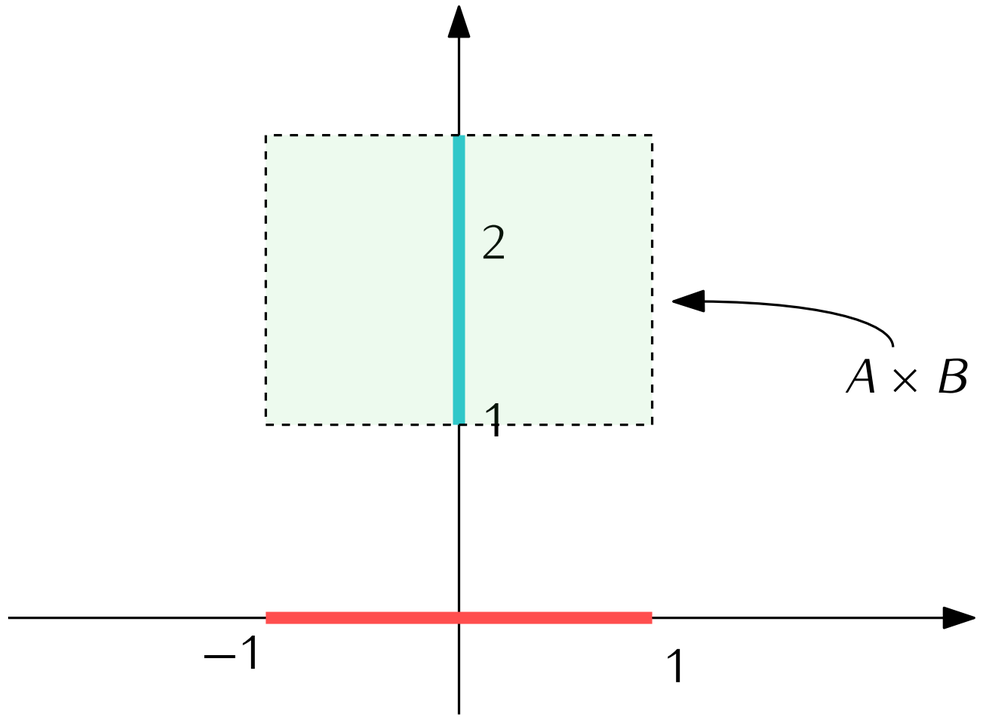
- $A = \{1, 2\}, B = \{1, 3, 4\}$

$A \times B = \{(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4)\}$



- $A = [-1, 1], B = [1, 2.5]$

$A \times B$: all points (x, y) with $x \in [-1, 1], y \in [1, 2.5]$



- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

- We also write S^2 for $S \times S$
- S^3 for triples
- S^n for n-tuples

- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

- We also write S^2 for $S \times S$
- S^3 for triples
- S^n for n-tuples

```
from itertools import product

S = {0,1}
# Runs through all elements of SxSxSxS
for s in product(S,repeat=4):
    print(s)
```

- Common class/structure in programming: tuples
  Tuple: ordered list that can contain repetitions: $(1, 1, 2), (\text{'antoine'}, 20, 10), \ldots$
- Pair: tuple of length 2
- Triple/Triplet: tuple of length 3. n-tuple: tuple of length n

Definition. If S and T are sets, then $S \times T$ is the set of all pairs (s, t) where $s \in S$ and $t \in T$.

- We also write S^2 for $S \times S$
- S^3 for triples
- S^n for n-tuples

```
from itertools import product

S = {0,1}
# Runs through all elements of SxSxSxS
for s in product(S,repeat=4):
    print(s)
```

- Binary strings with n bits = elements of $\{0, 1\}^n$
- Character strings = elements of $\{\text{'a'}, \text{'b'}, \ldots, \text{'z'}\}^n$
- Vectors in n dimensions = elements of $\mathbb{R}^n$

Definition. The powerset of S is the set $\mathcal{P}(S)$ that contains all the subsets of S.

So $A \in \mathcal{P}(S)$ is the same as $A \subseteq S$.
In particular we always have $S \in \mathcal{P}(S)$ and $\emptyset \in \mathcal{P}(S)$.

Definition. The powerset of S is the set $\mathcal{P}(S)$ that contains all the subsets of S.

So $A \in \mathcal{P}(S)$ is the same as $A \subseteq S$.
In particular we always have $S \in \mathcal{P}(S)$ and $\emptyset \in \mathcal{P}(S)$.

Example. $\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

Definition. The powerset of S is the set $\mathcal{P}(S)$ that contains all the subsets of S.

So $A \in \mathcal{P}(S)$ is the same as $A \subseteq S$.
In particular we always have $S \in \mathcal{P}(S)$ and $\emptyset \in \mathcal{P}(S)$.

Example. $\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

```
from more-itertools import powerset

S = {1,2,3}
from A in powerset(S):
    print(A)
# ()
# (1,), (2,), (3,)
# (1,2), (1,3), (2,3)
# (1,2,3)
```

Definition. The powerset of S is the set $\mathcal{P}(S)$ that contains all the subsets of S.

So $A \in \mathcal{P}(S)$ is the same as $A \subseteq S$.
In particular we always have $S \in \mathcal{P}(S)$ and $\emptyset \in \mathcal{P}(S)$.

Example. $\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

How many elements does $\mathcal{P}(\{e, \pi\})$ have?

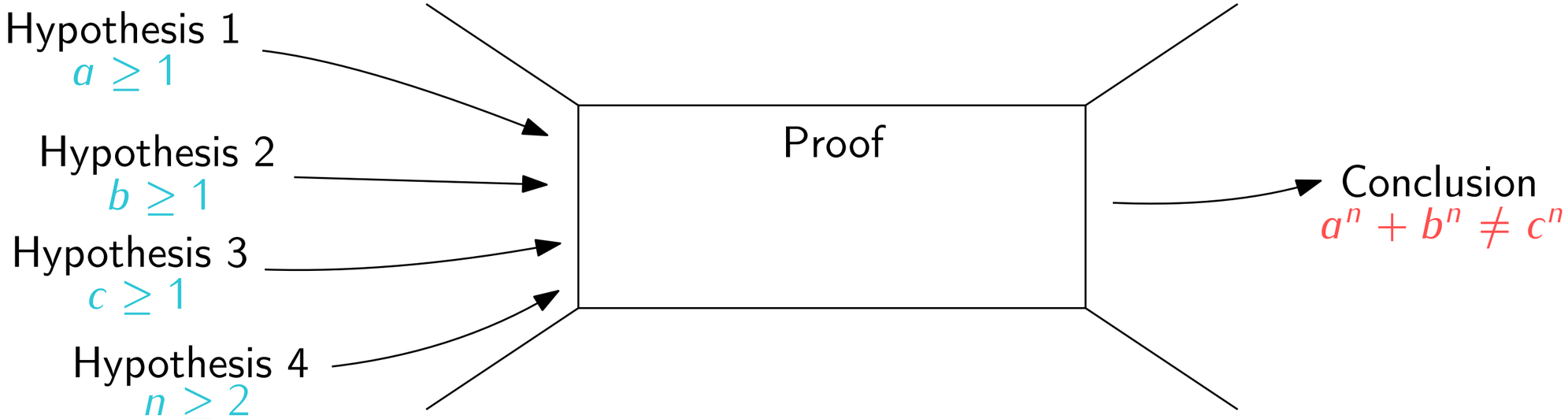
```
from more-itertools import powerset

S = {1,2,3}
from A in powerset(S):
    print(A)
# ()
# (1,), (2,), (3,)
# (1,2), (1,3), (2,3)
# (1,2,3)
```

- Higher mathematics is not about computing
- It's about proving, making deductions

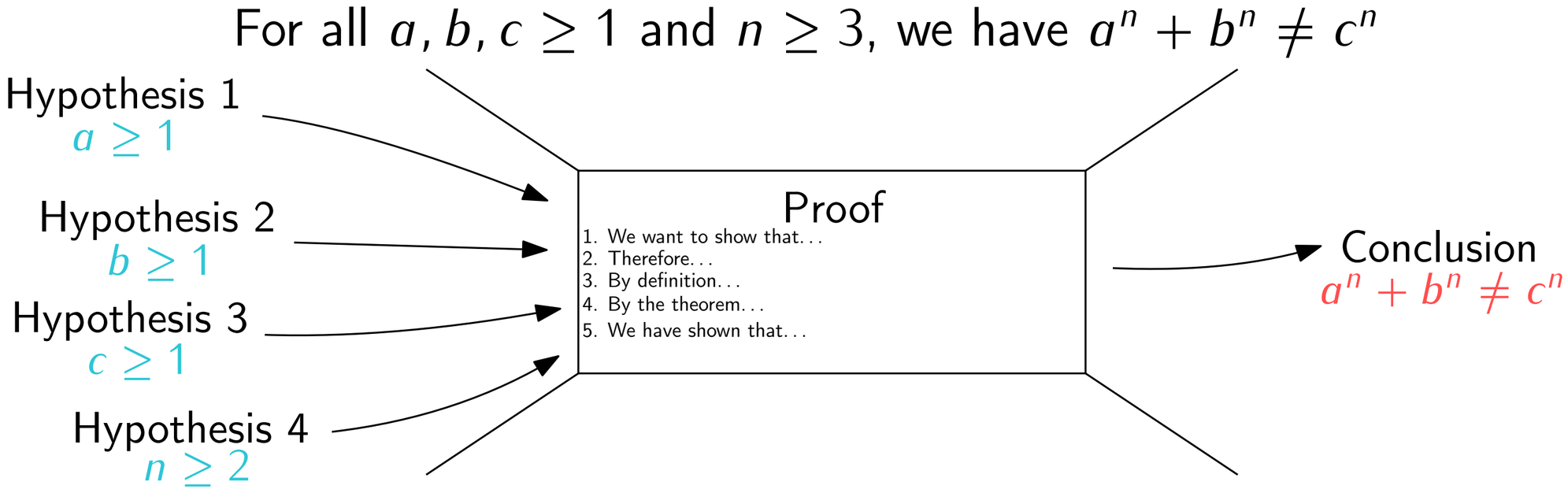
- Higher mathematics is not about computing
- It's about proving, making deductions
- A proof guarantees that in every scenario where the hypotheses are true, the conclusion is also true

- Higher mathematics is not about computing
- It's about proving, making deductions
- A proof guarantees that in every scenario where the hypotheses are true, the conclusion is also true

For all $a, b, c \geq 1$ and $n \geq 3$, we have $a^n + b^n \neq c^n$

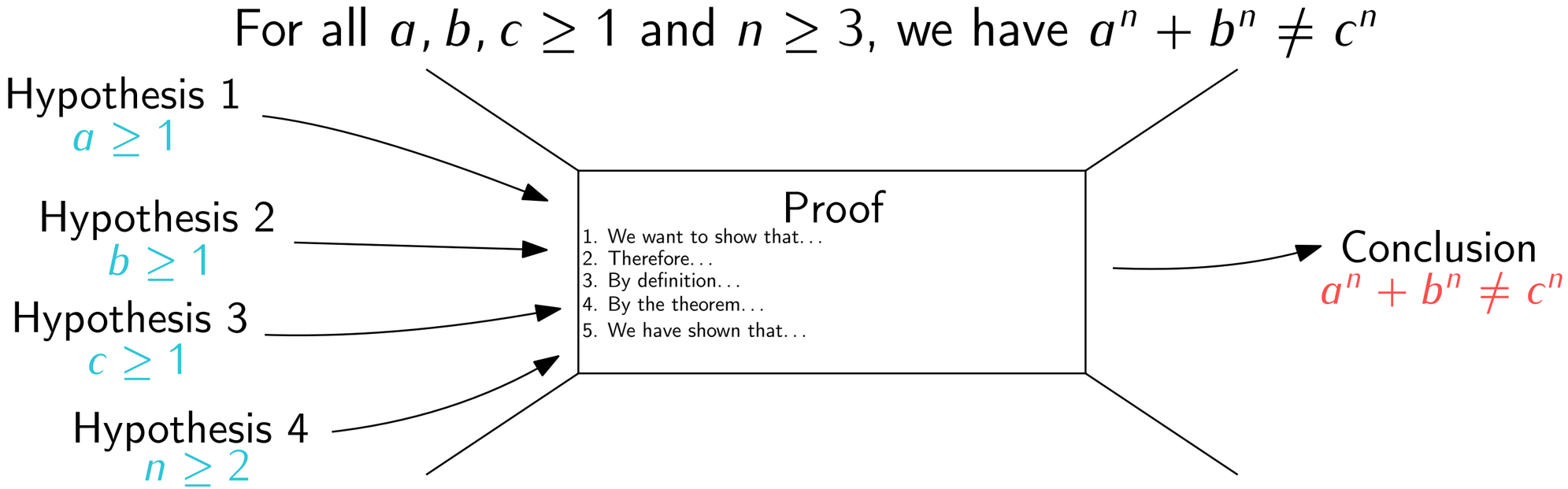


- Higher mathematics is not about computing
- It's about proving, making deductions
- A proof guarantees that in every scenario where the hypotheses are true, the conclusion is also true



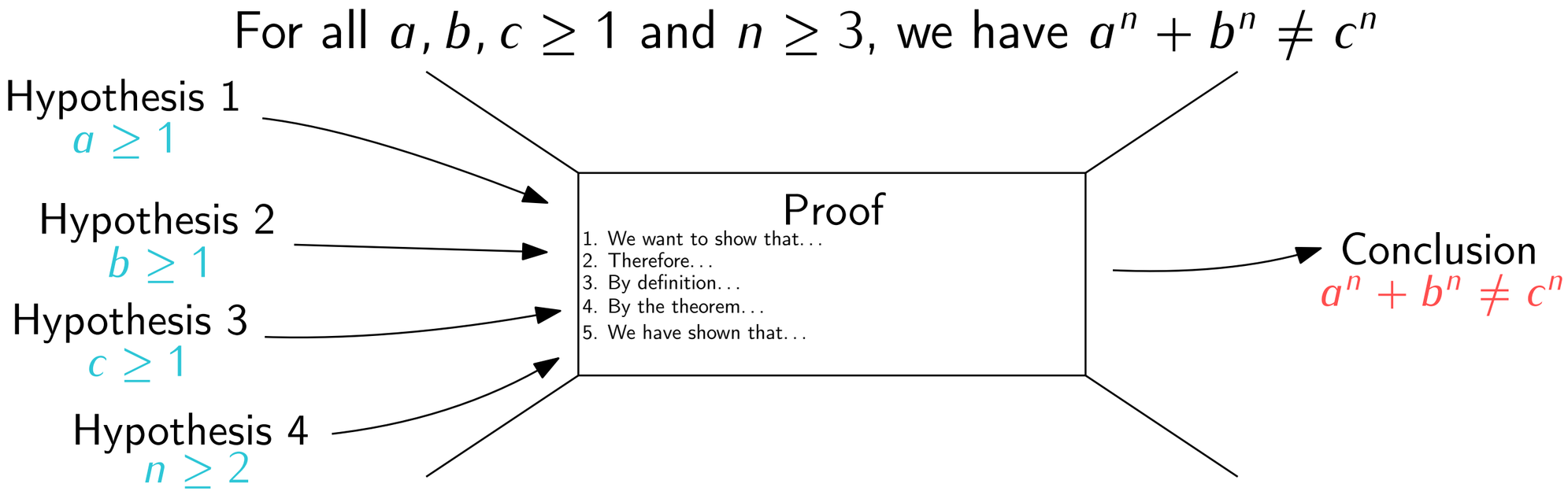
- A proof is a sequence of steps, like a cooking recipe. At each step:
  - Apply a definition
  - Apply a hypothesis
  - Apply a logical rule

- Higher mathematics is not about computing
- It's about proving, making deductions
- A proof guarantees that in every scenario where the hypotheses are true, the conclusion is also true



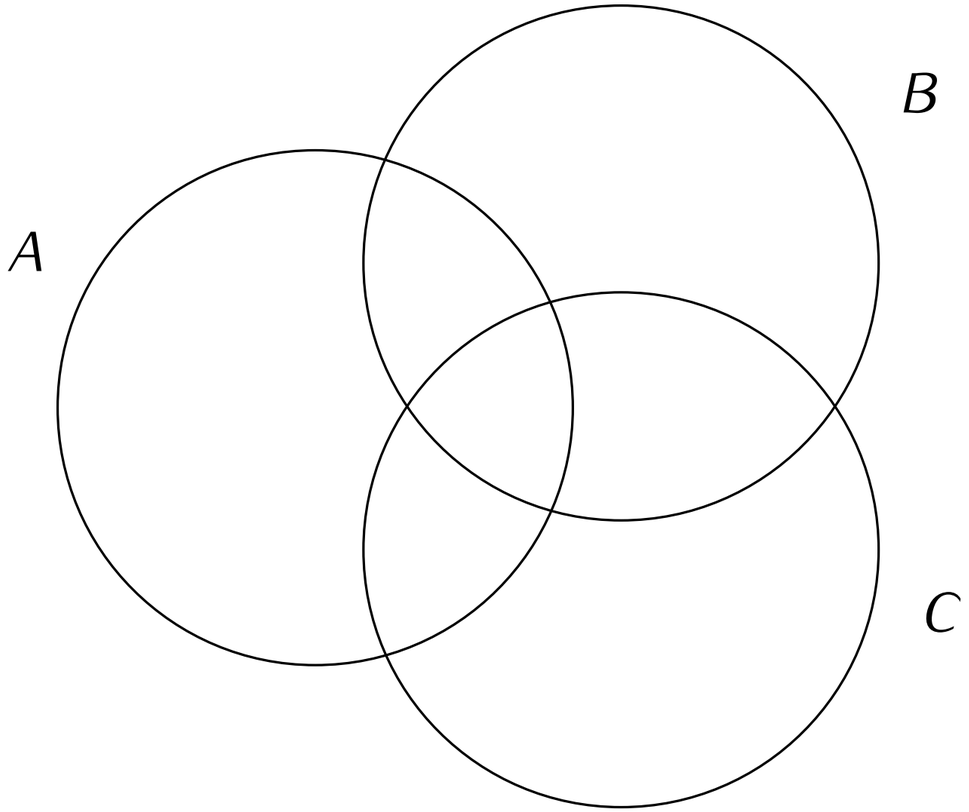
- A proof is a sequence of steps, like a cooking recipe. At each step:
  - Apply a definition
  - Apply a hypothesis
  - Apply a logical rule
- No recipe to learn proofs other than practice

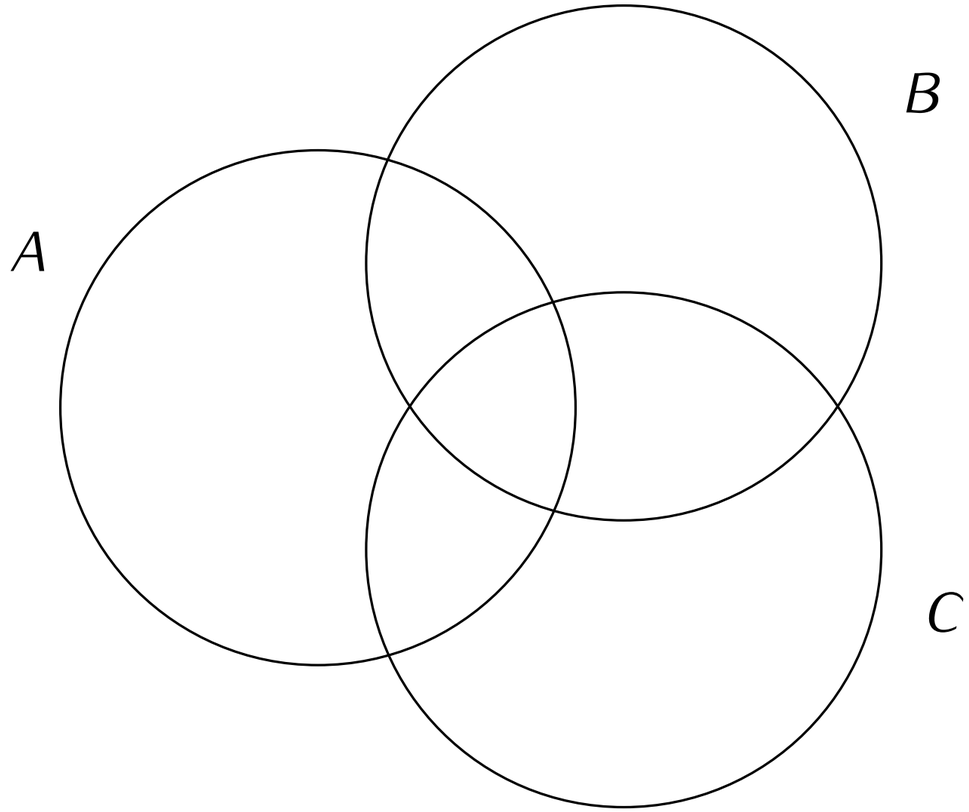
- Higher mathematics is not about computing
- It's about proving, making deductions
- A proof guarantees that in every scenario where the hypotheses are true, the conclusion is also true



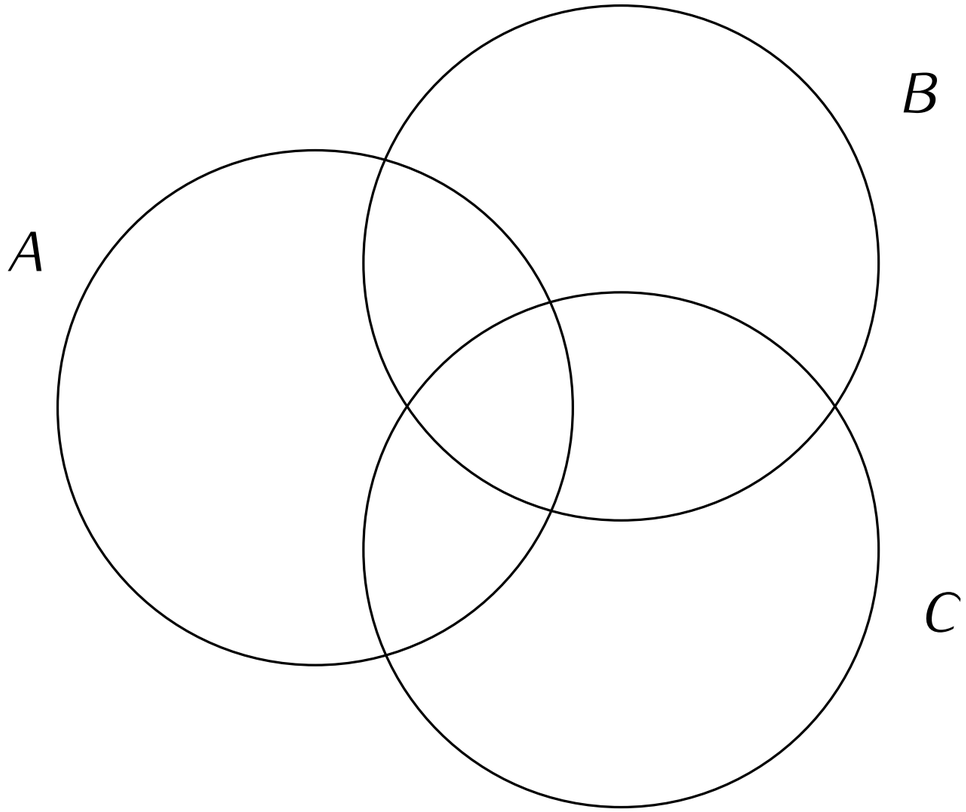
- A proof is a sequence of steps, like a cooking recipe. At each step:
  - Apply a definition
  - Apply a hypothesis
  - Apply a logical rule
- No recipe to learn proofs other than practice
- An example is **not** a proof!

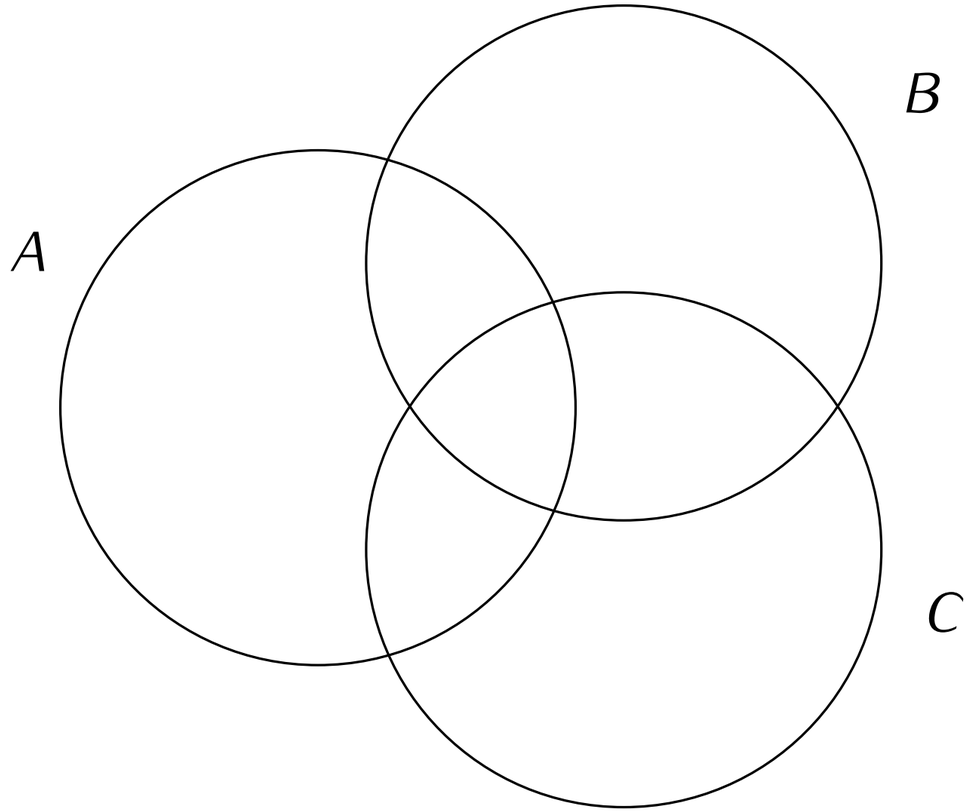
Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.





Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

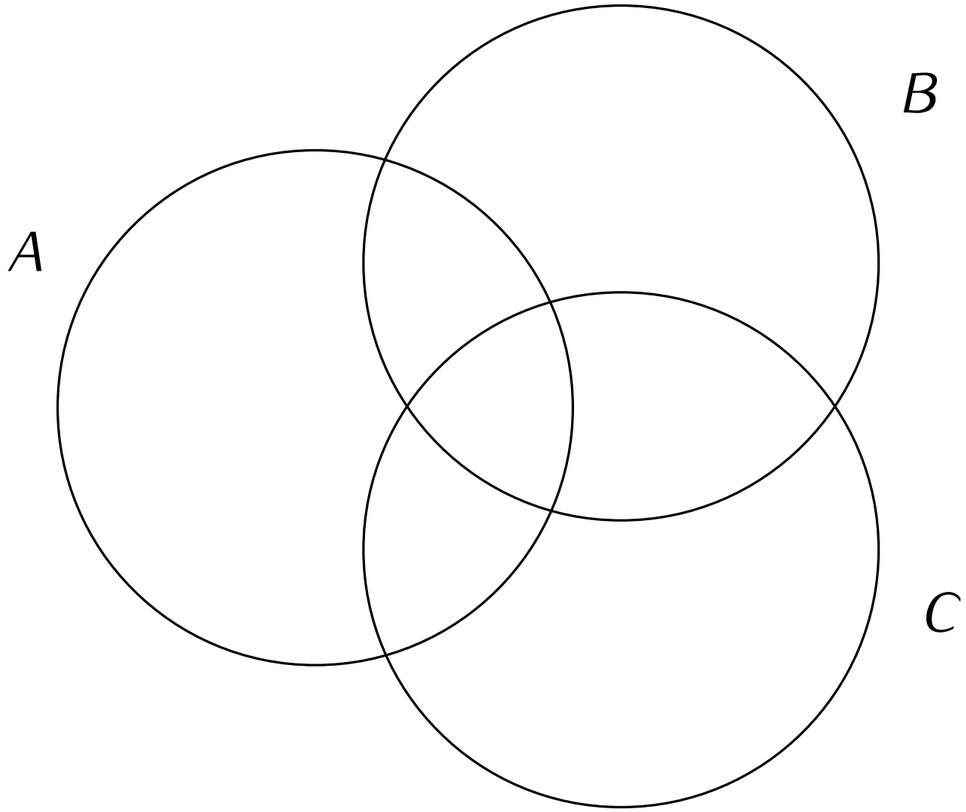


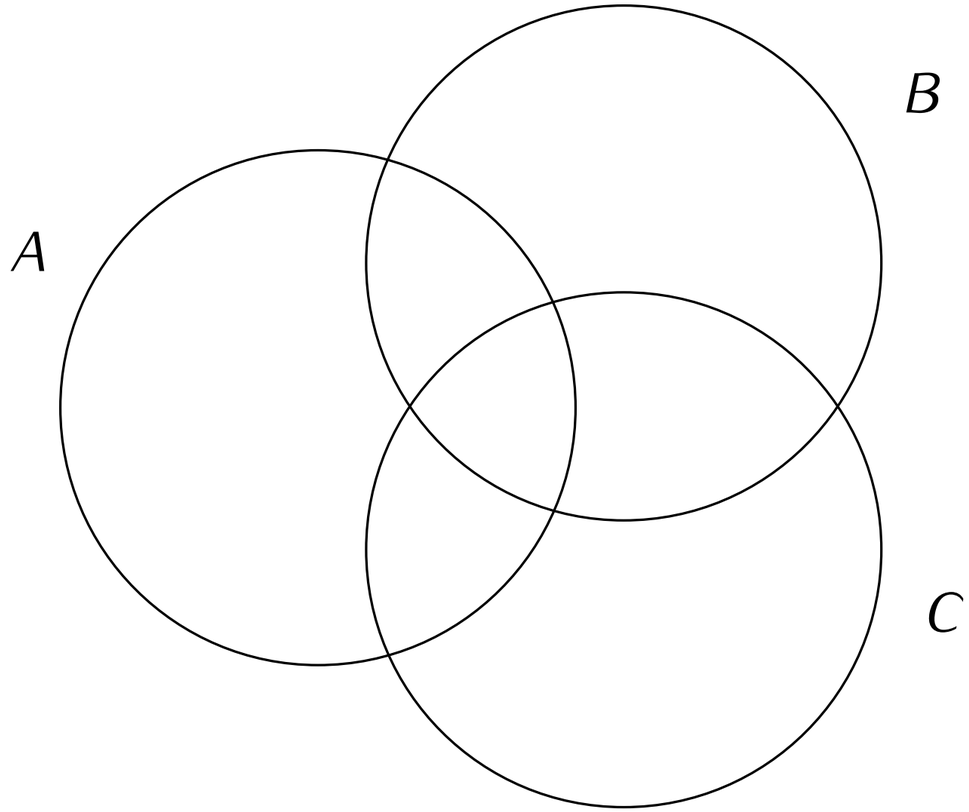


We cannot prove things just by drawings:

- how can we be sure it does not depend on how we have drawn the potatoes?
- think about the mess it would be if there were 10 sets instead

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.





We cannot prove things just by drawings:

- how can we be sure it does not depend on how we have drawn the potatoes?
- think about the mess it would be if there were 10 sets instead

$$A \cap (B_1 \cup \cdots \cup B_9) \subseteq (A \cap B_1) \cup \cdots \cup (A \cap B_9)$$

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Proof. (what you write down)

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set

Goal: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$

Proof. (what you write down)

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set

Goal: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

Proof. (what you write down)

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set

Goal: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.
  Goal is to prove $\ldots \subseteq \ldots$.
  We need to apply the definition.
  This generates a new hypothesis.

Proof. (what you write down)
1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set

Goal: if $x \in A \cap (B \cup C)$, then $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.
  Goal is to prove $\ldots \subseteq \ldots$.
  We need to apply the definition.
  This generates a new hypothesis.

Proof. (what you write down)
1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 4: $x \in A \cap (B \cup C)$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

Proof. (what you write down)
1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 4: $x \in A \cap (B \cup C)$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses
  Hyp. 4 is $x \in \ldots \cap \ldots$

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 6: $x \in B \cup C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses
  Hyp. 4 is $x \in \ldots \cap \ldots$

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 6: $x \in B \cup C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses
  Hyp. 6 is $x \in \ldots \cup \ldots$
  This means x is in at least one of the two sets
  There are 2 cases to deal with

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 7: $x \in B$ or $x \in C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses
  Hyp. 6 is $x \in \ldots \cup \ldots$
  This means x is in at least one of the two sets
  There are 2 cases to deal with

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 8: $x \in B$

Goal: $x \in (A \cap B) \cup (A \cap C)$

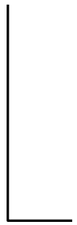
- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$



Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 9: $x \in A \cap B$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

Hyp. 5+8: $x \in A \cap B$

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 9: $x \in A \cap B$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

> Hyp. 5+8: $x \in A \cap B$
>
> Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
>
> This is what we wanted to prove

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 7: $x \in B$ or $x \in C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

> Hyp. 5+8: $x \in A \cap B$
> Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
> This is what we wanted to prove

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 5: $x \in A$
Hypothesis 10: $x \in C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

- Hyp. 5+8: $x \in A \cap B$
- Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
- This is what we wanted to prove

4b. Case 2: $x \in C$

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 11: $x \in A \cap C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

- Hyp. 5+8: $x \in A \cap B$
- Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
- This is what we wanted to prove

4b. Case 2: $x \in C$

- Hyp. 5+10: $x \in A \cap C$

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 11: $x \in A \cap C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.
- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$

> Hyp. 5+8: $x \in A \cap B$
> Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
> This is what we wanted to prove

4b. Case 2: $x \in C$

> Hyp. 5+10: $x \in A \cap C$
> Hyp. 11 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
> This is what we wanted to prove

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Notes. (how you start thinking about the problem)

- Step 0: identify the hypotheses and the conclusion

Hypothesis 1: A is a set
Hypothesis 2: B is a set
Hypothesis 3: C is a set
Hypothesis 11: $x \in A \cap C$

Goal: $x \in (A \cap B) \cup (A \cap C)$

- Step 1: understand the goal.

- Step 2: understand the hypotheses

Proof. (what you write down)

1. Def: $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ means "every element of $A \cap (B \cup C)$ must be an element of $(A \cap B) \cup (A \cap C)$."
2. Def: $x \in A \cap (B \cup C)$ means "$x \in A$ **and** $x \in B \cup C$"
3. Def: $x \in B \cup C$ means "$x \in B$ **or** $x \in C$"

4a. Case 1: $x \in B$
- Hyp. 5+8: $x \in A \cap B$
- Hyp. 9 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
- This is what we wanted to prove

4b. Case 2: $x \in C$
- Hyp. 5+10: $x \in A \cap C$
- Hyp. 11 and def. of $\cup$: $x \in (A \cap B) \cup (A \cap C)$
- This is what we wanted to prove

5. Done! □

Theorem. If A, B, C are sets, then $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

Proof.

By definition of $\subseteq$, we need to prove that every element of $A \cap (B \cup C)$ is in $(A \cap B) \cup (A \cap C)$.
By definition of $\cap$, $x \in A$ **and** $x \in B \cup C$.
By definition of $\cup$, $x \in B$ **or** $x \in C$.

Case 1: $x \in B$

Since $x \in A$ and $x \in B$, we know $x \in A \cap B$.
By definition of $\cup$: $x \in (A \cap B) \cup (A \cap C)$.
This is what we wanted to prove.

Case 2: $x \in C$

Since $x \in A$ and $x \in C$, we know $x \in A \cap C$.
By definition of $\cup$: $x \in (A \cap B) \cup (A \cap C)$.
This is what we wanted to prove. $\square$

- do not only use examples: examples are good to get intuition, but examples are not a proof
- rephrase/decompose hypotheses and goal using the definitions
- in general, to prove $S \subseteq T$ you need to
  - start with "Assume $x \in S$." // "Sei $x \in S$"
  - blablabla...
  - "Therefore $x \in T$." // "Deshalb $x \in T$."

- do not only use examples: examples are good to get intuition, but examples are not a proof
- rephrase/decompose hypotheses and goal using the definitions
- in general, to prove $S \subseteq T$ you need to
  - start with "Assume $x \in S$." // "Sei $x \in S$"
  - blablabla...
  - "Therefore $x \in T$." // "Deshalb $x \in T$."

How to check whether your proof is correct:

- Every step should be understandable and require no further explanation.
- Give your proof to your study partner. They should be able to read and understand without any help (if they know the definitions).

- Definition of $\cap, \cup, \times, \setminus, \Delta, \mathcal{P}$
- How to prove $S \subseteq T$
- How to prove $S = T$
  (prove $S \subseteq T$ and $T \subseteq S$)