

# Binary Relations

## Part One

# Outline for Today

- ***Binary Relations***
  - Reasoning about connections between objects.
- ***Equivalence Relations***
  - Reasoning about clusters.
- ***A Fundamental Theorem***
  - How do we know we have the “right” definition for something?

# Relationships

- In CS103, you've seen examples of relationships

- between sets:

$$A \subseteq B$$

- between numbers:

$$x < y \quad x \equiv_k y \quad x \leq y$$

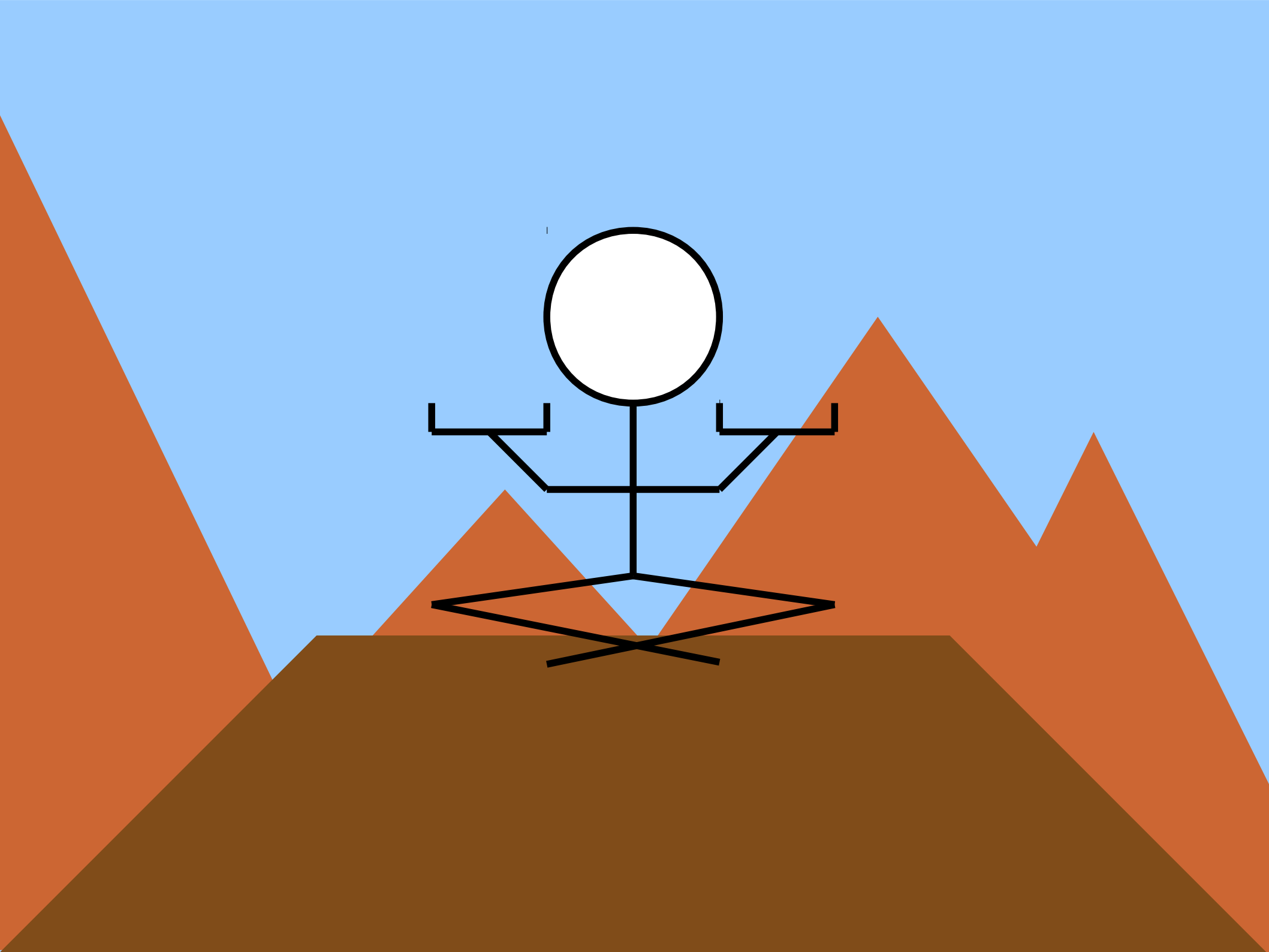
- between people:

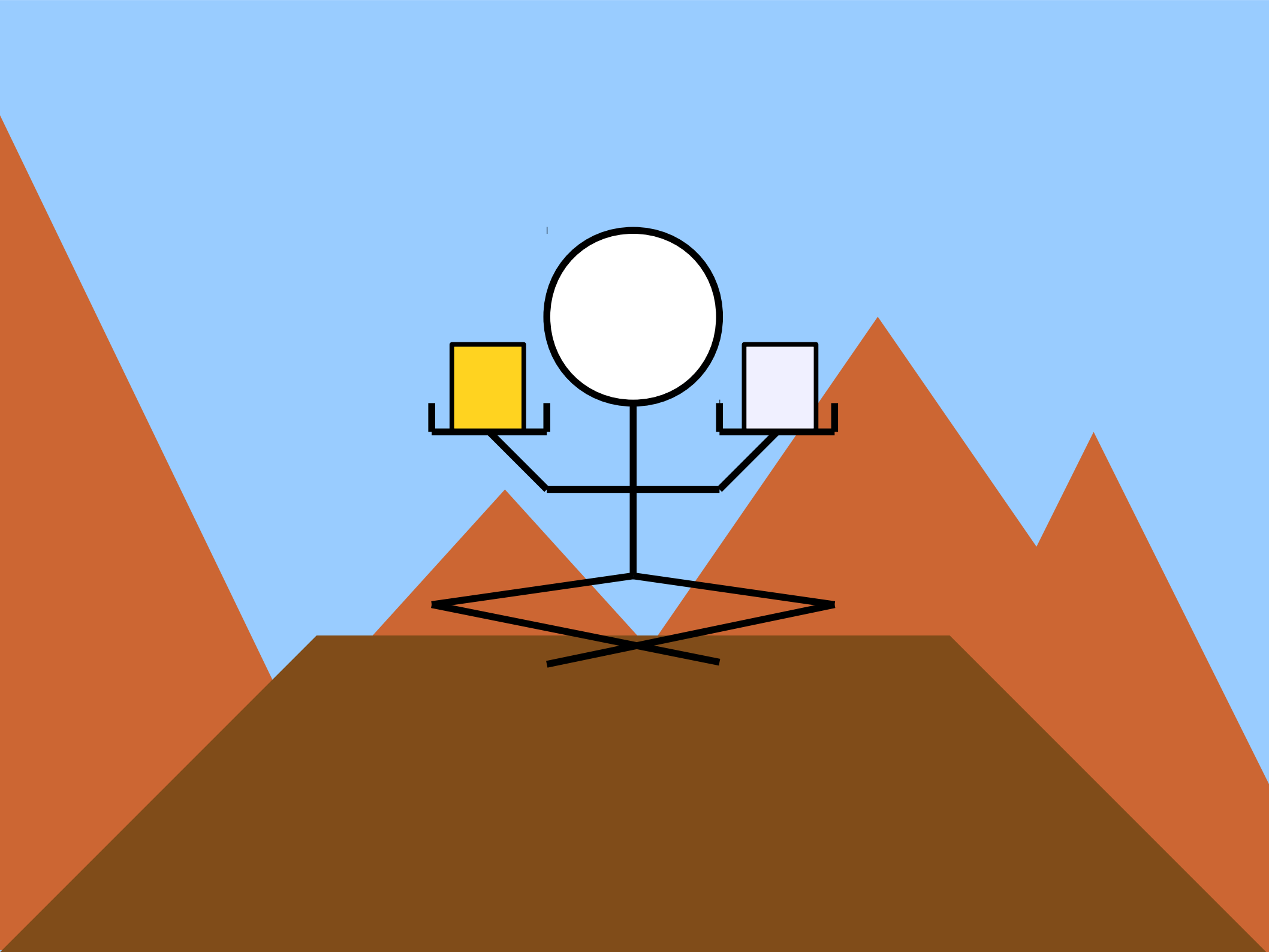
$$p \text{ loves } q$$

- Since these relations focus on connections between two objects, they are called **binary relations**.

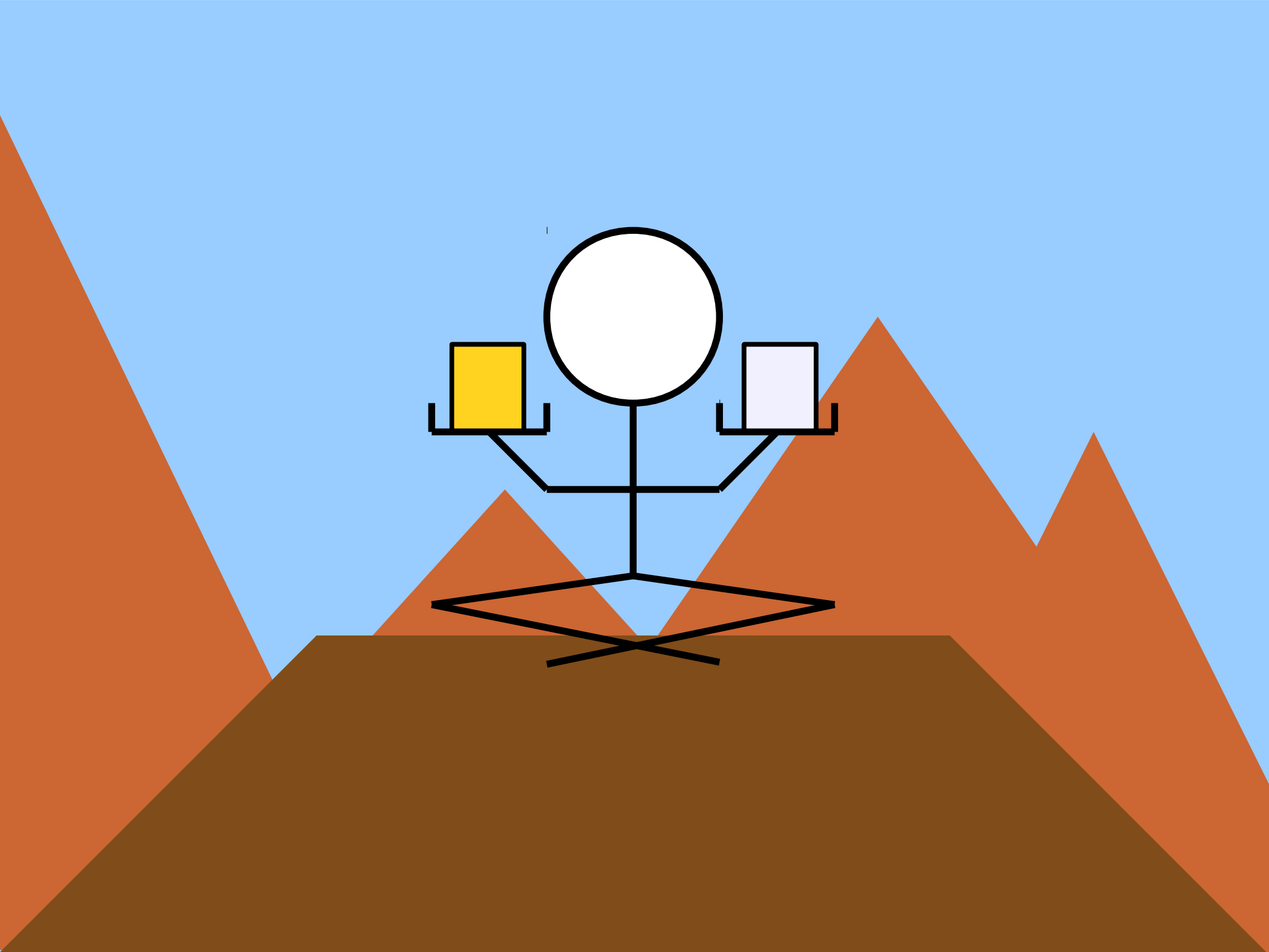
- The “binary” here means “pertaining to two things,” not “made of zeros and ones.”

What exactly is a binary relation?

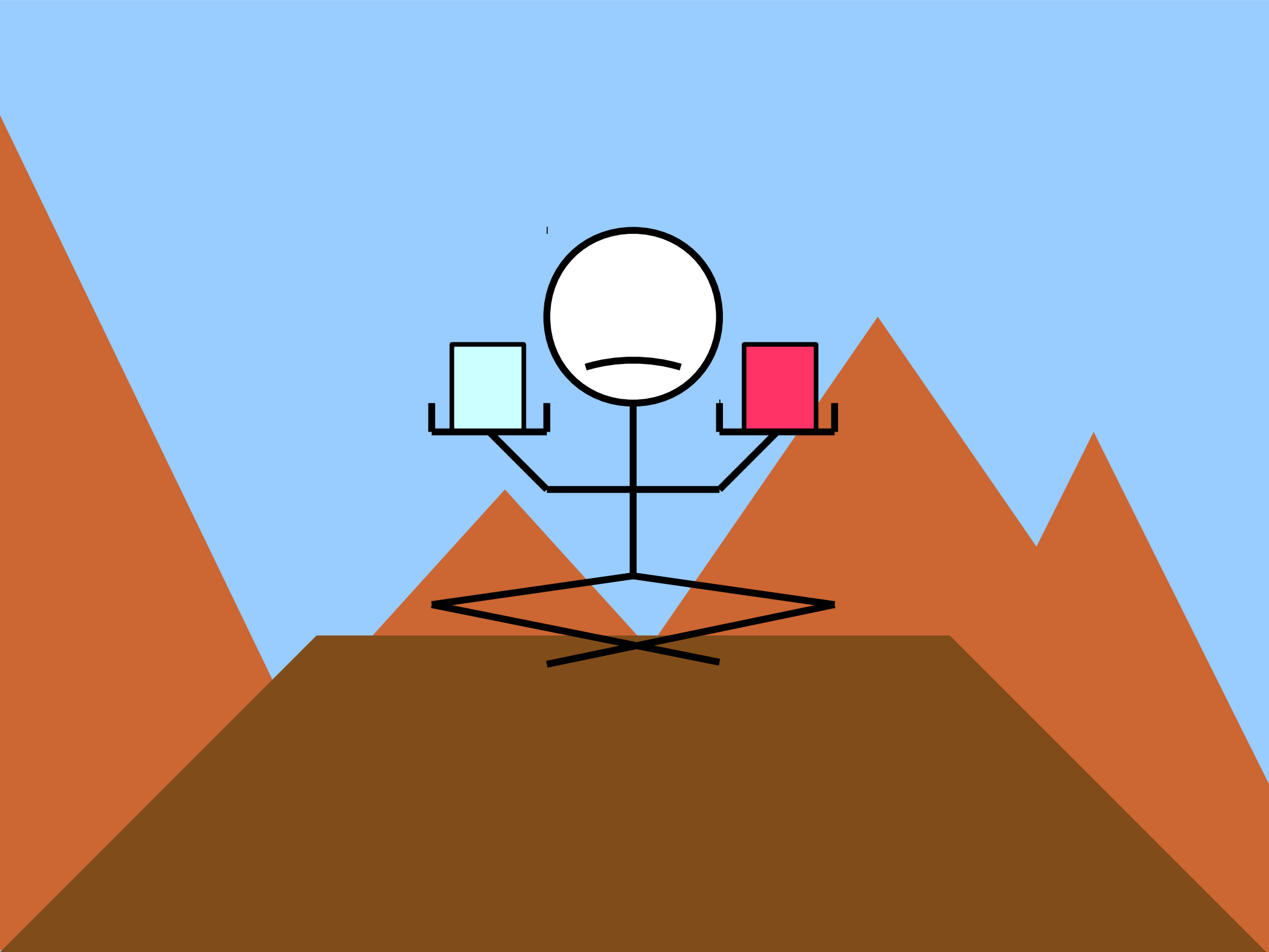


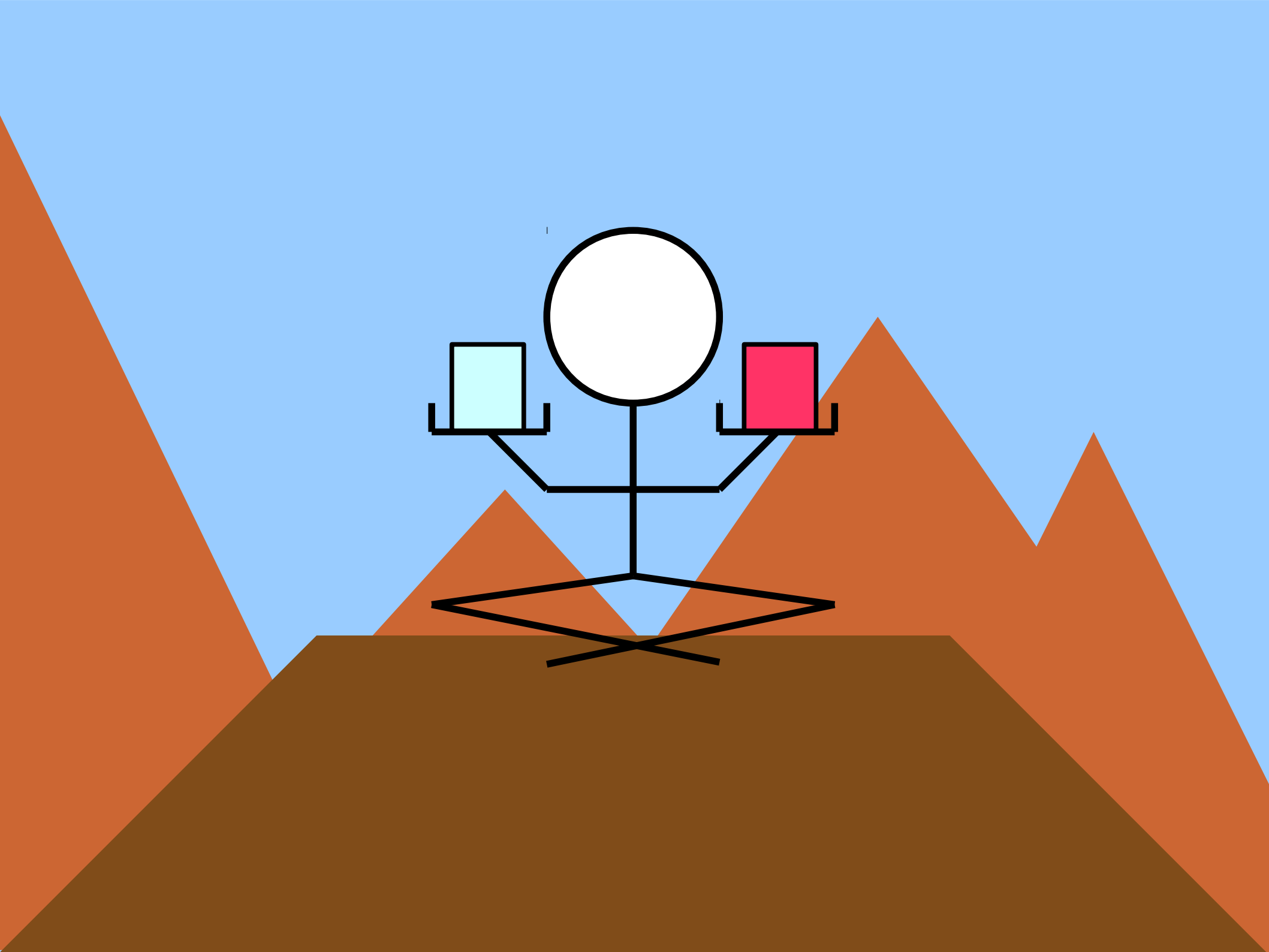




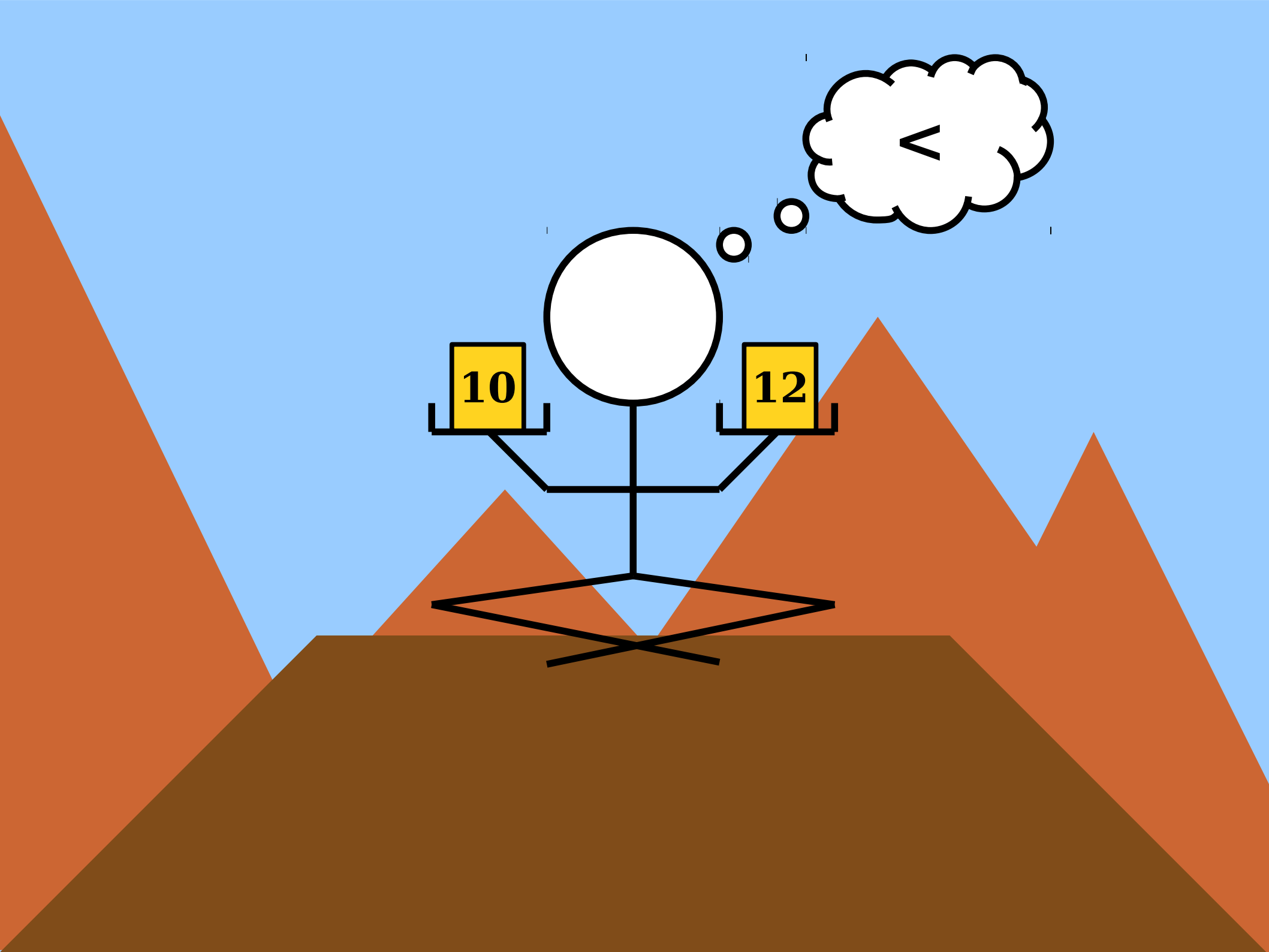








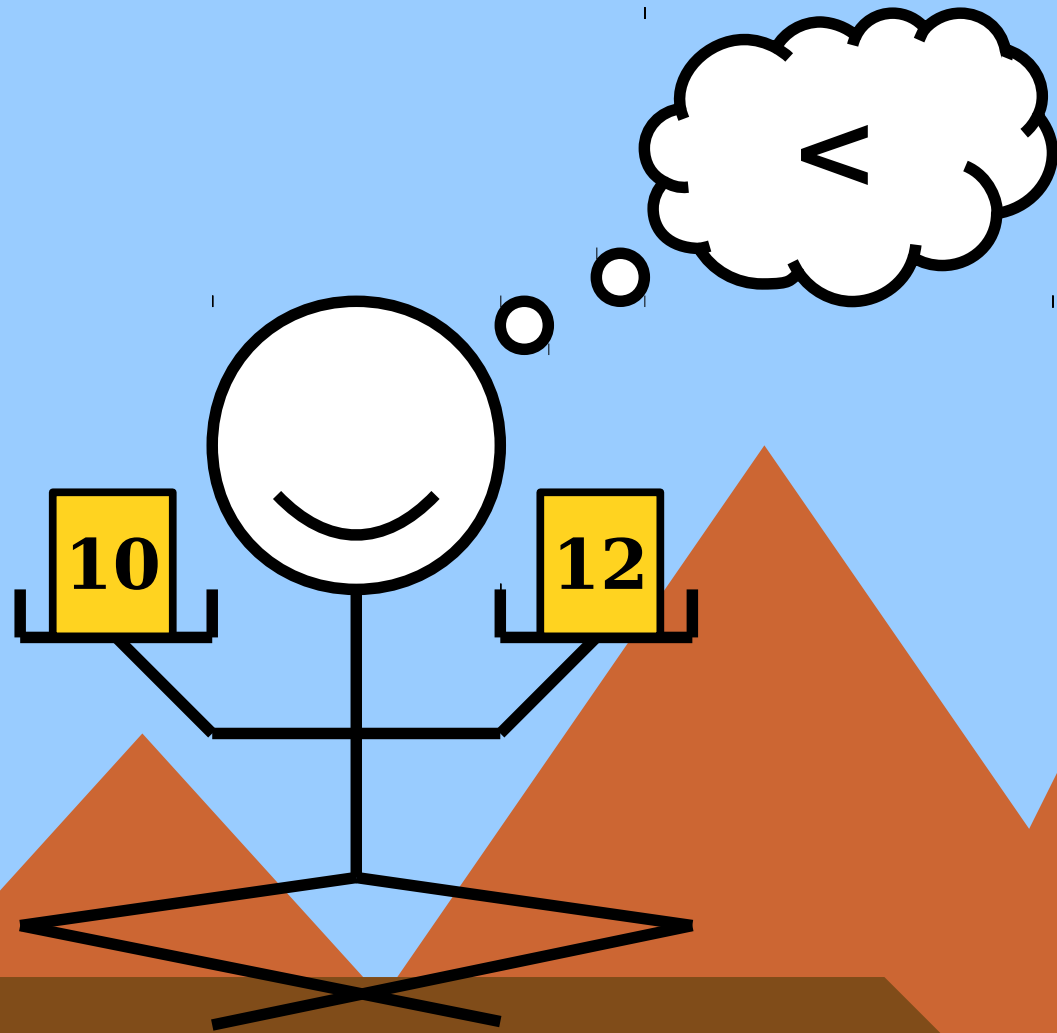




10

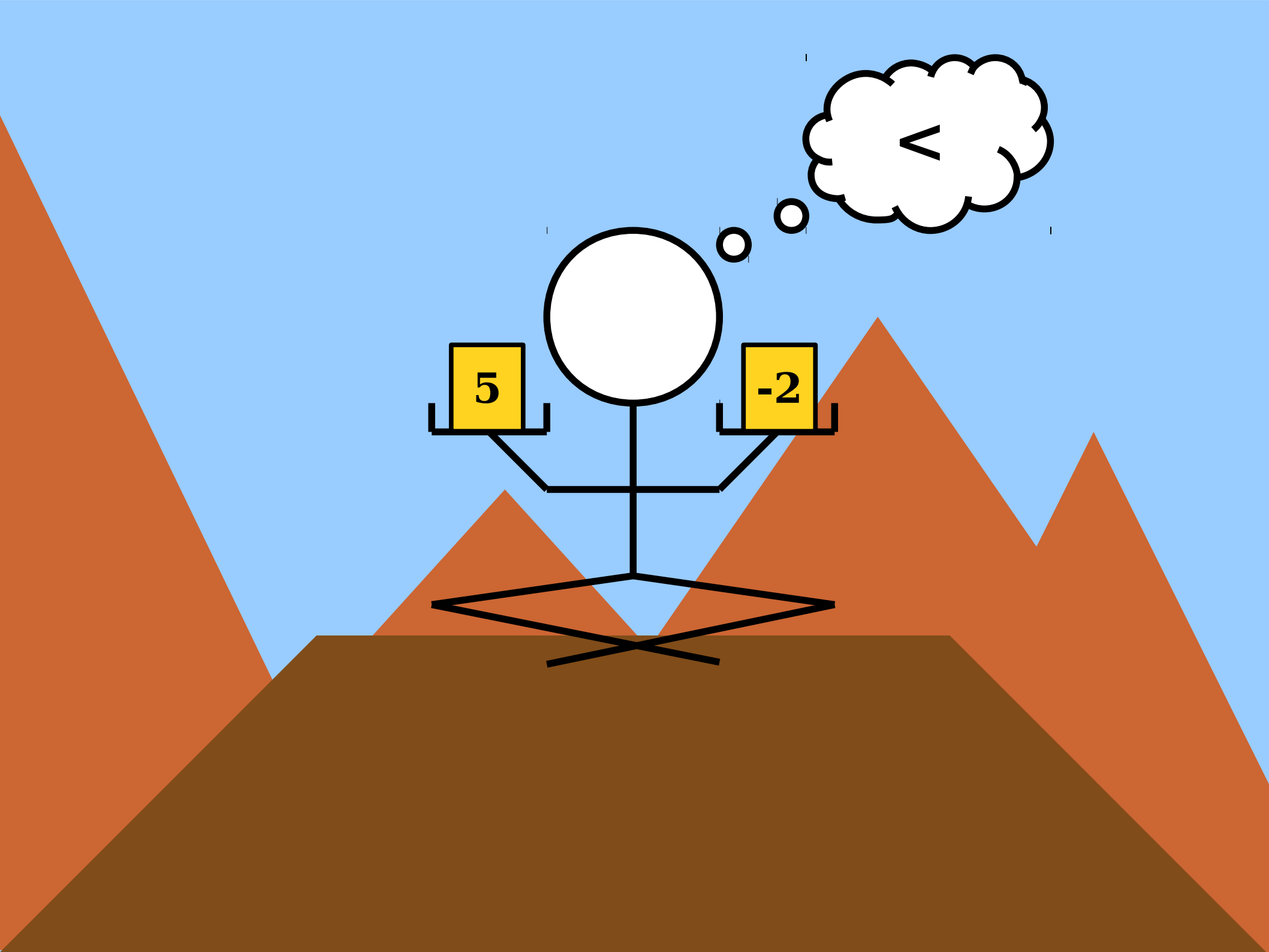
12

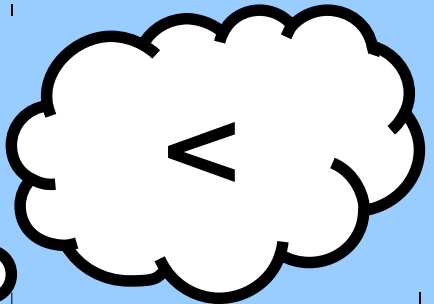
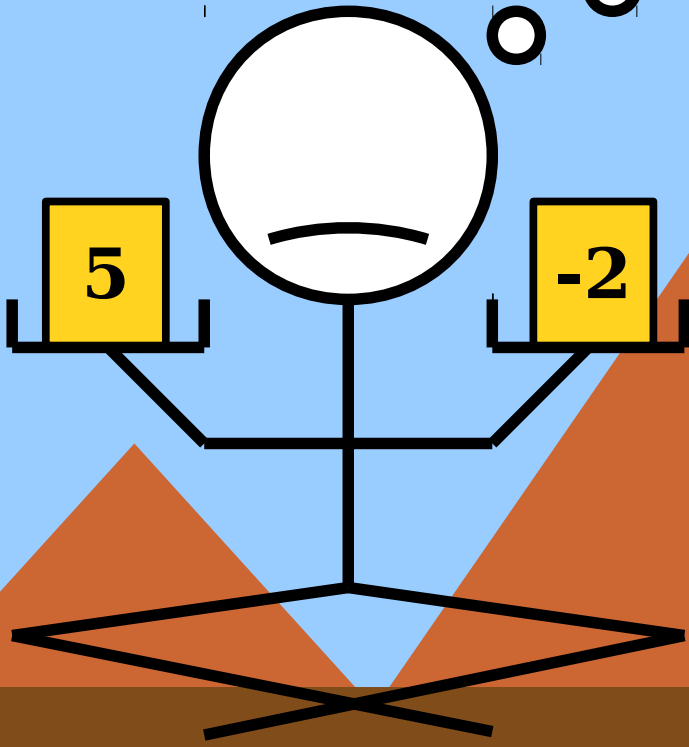
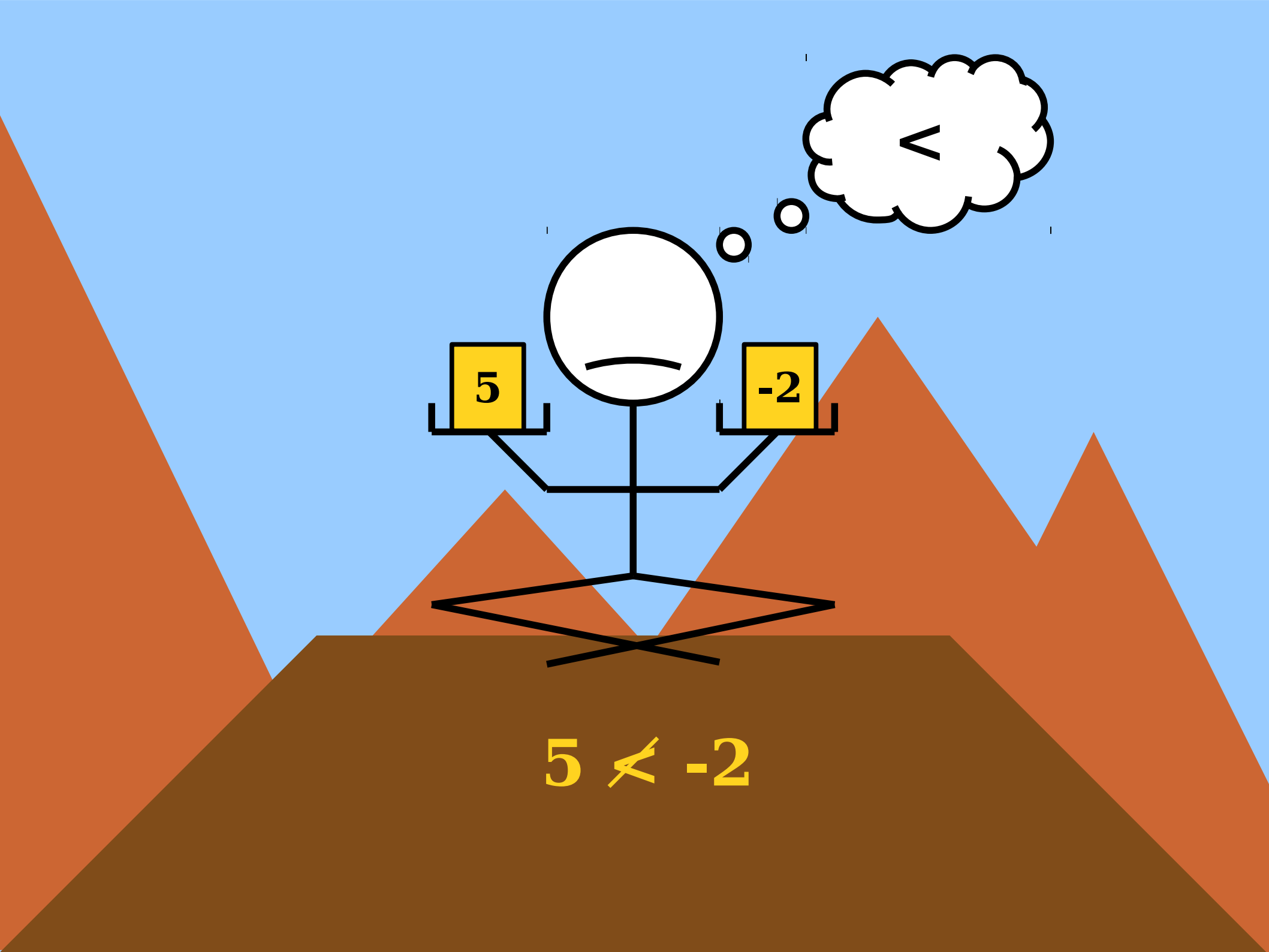
>



$$10 < 12$$

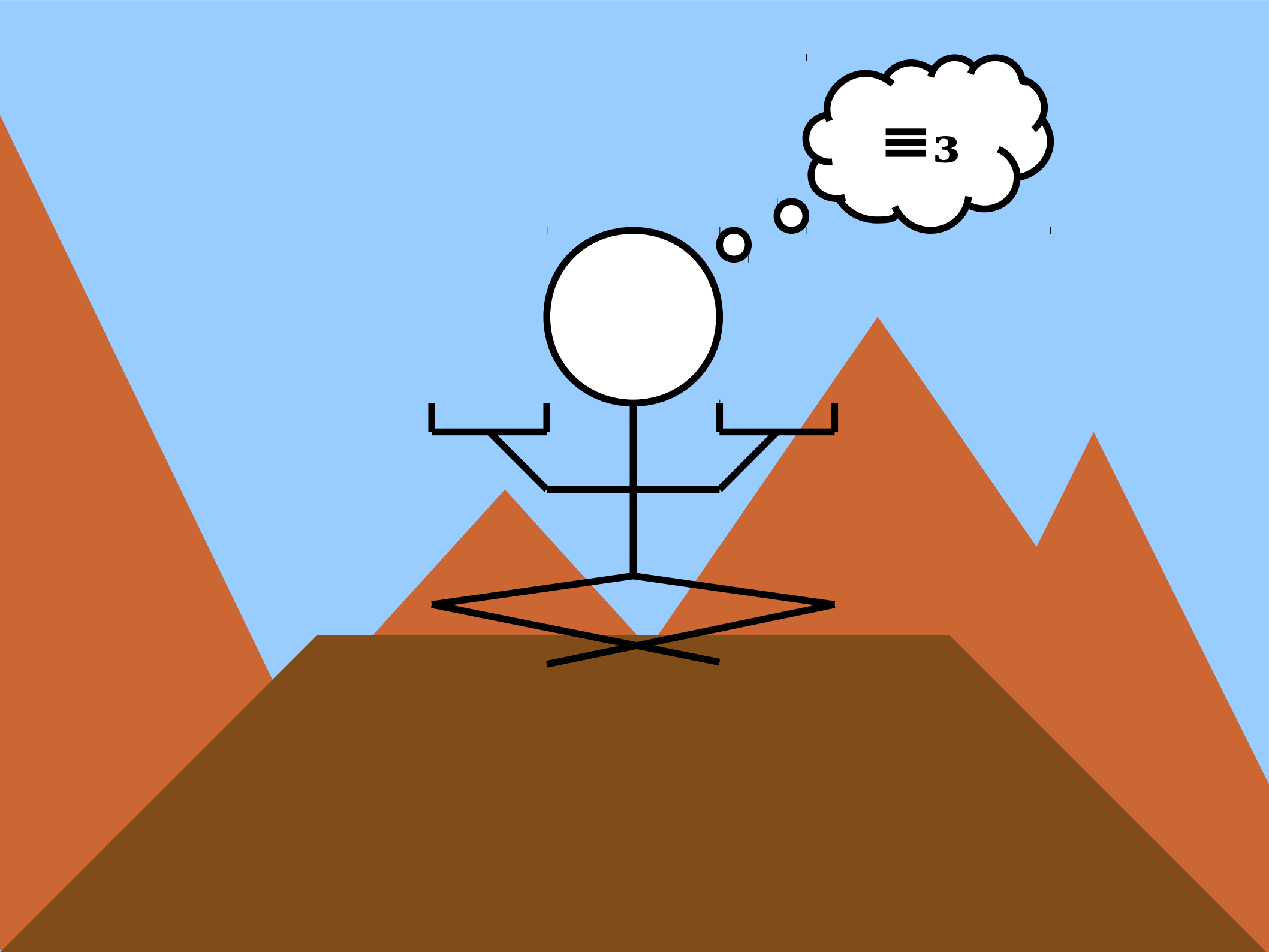




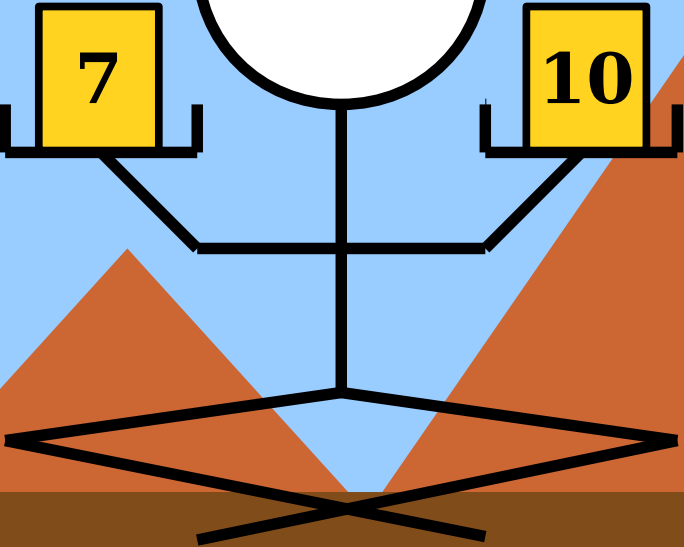
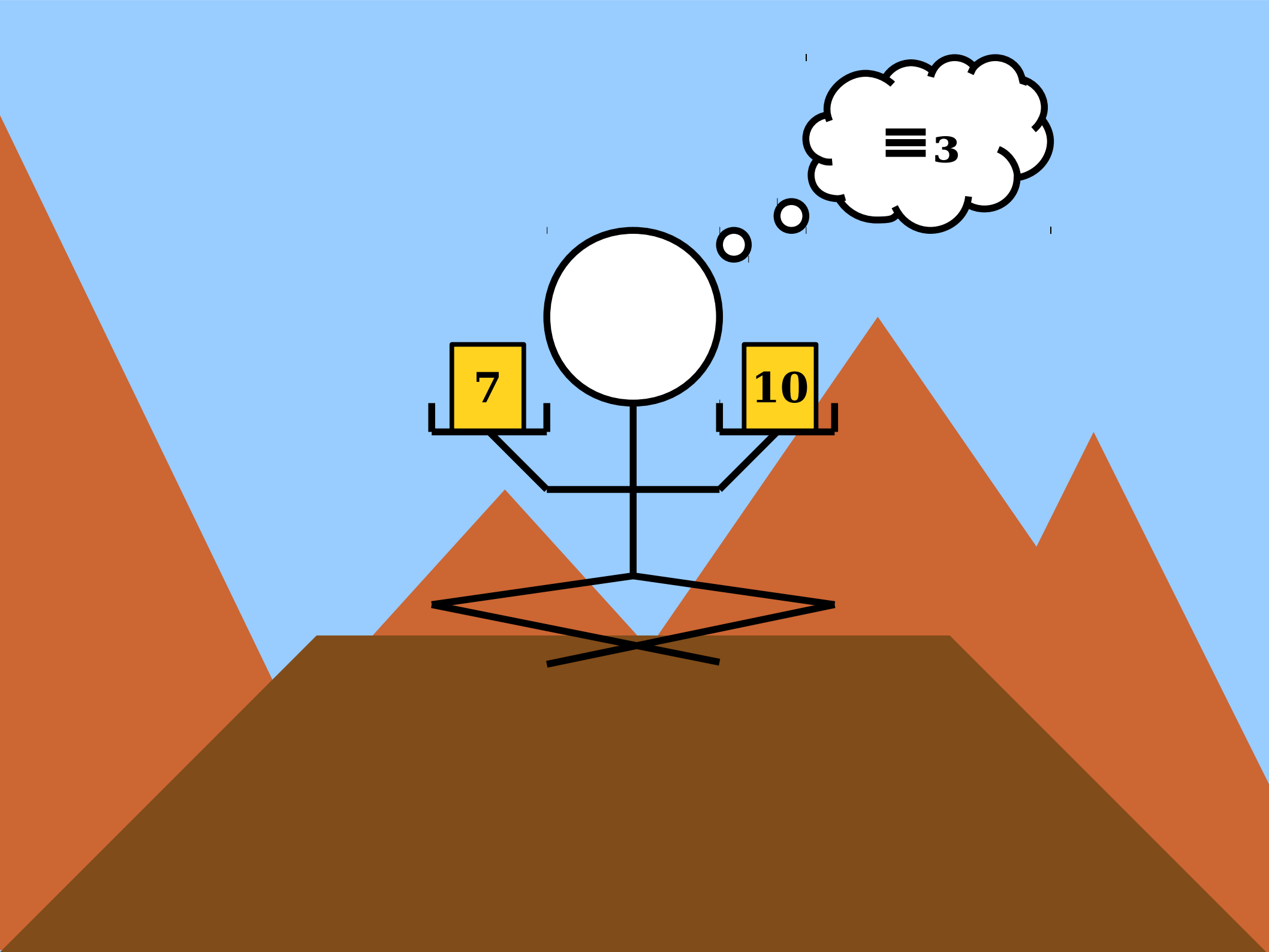


$5 < -2$

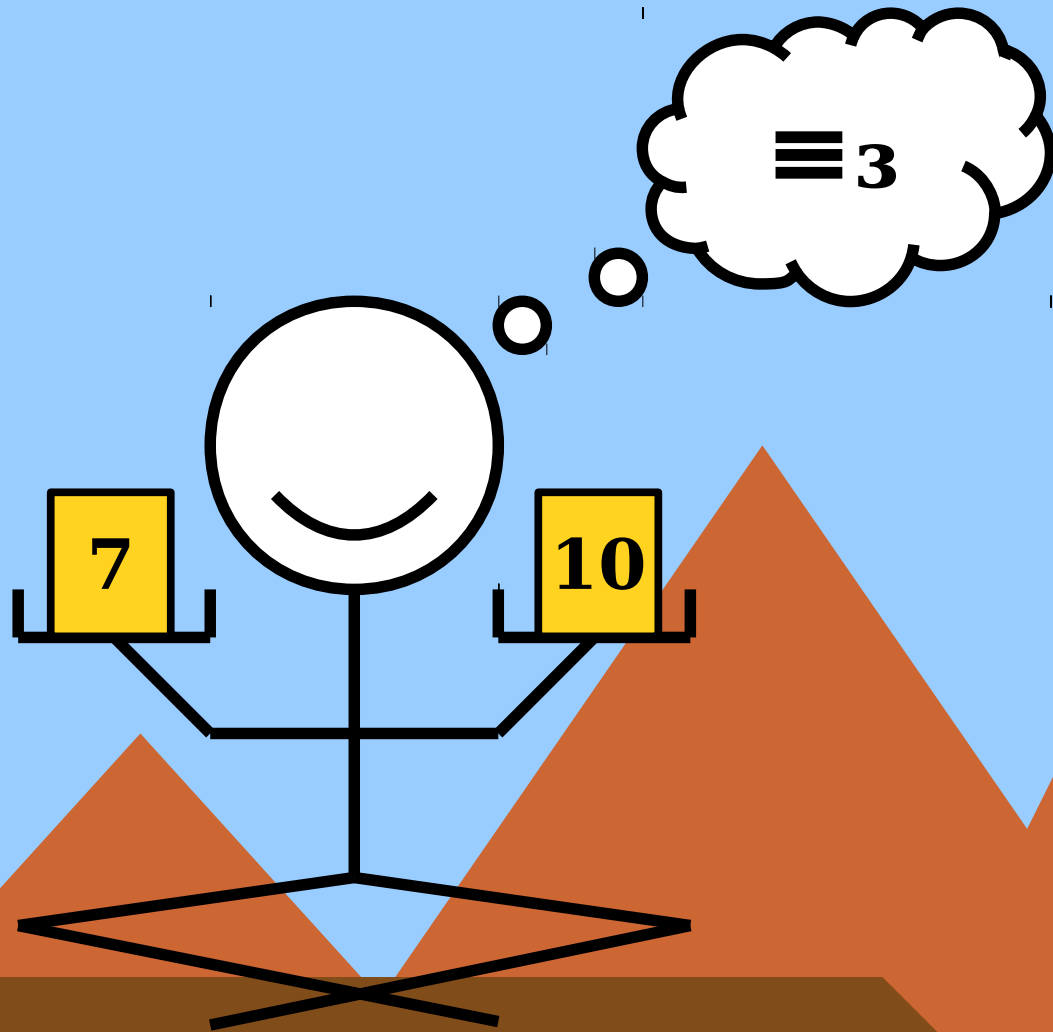




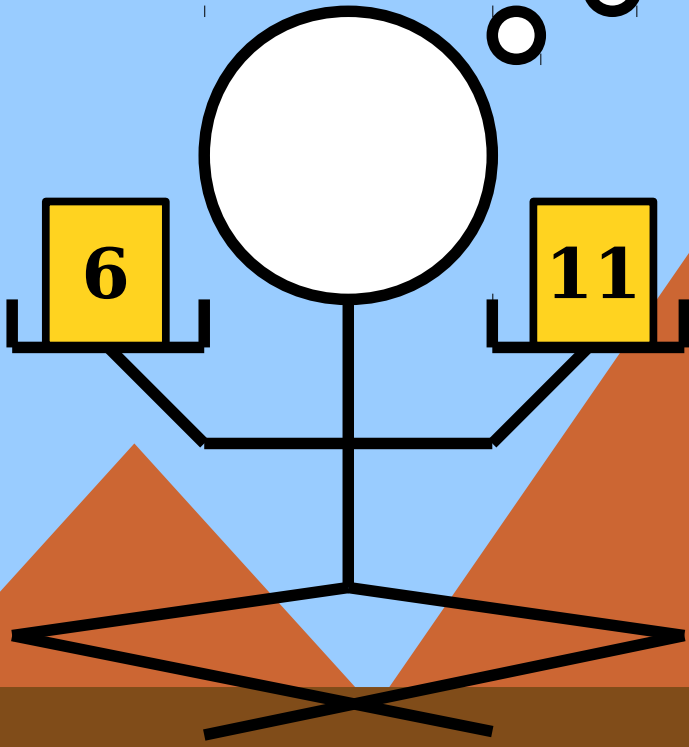
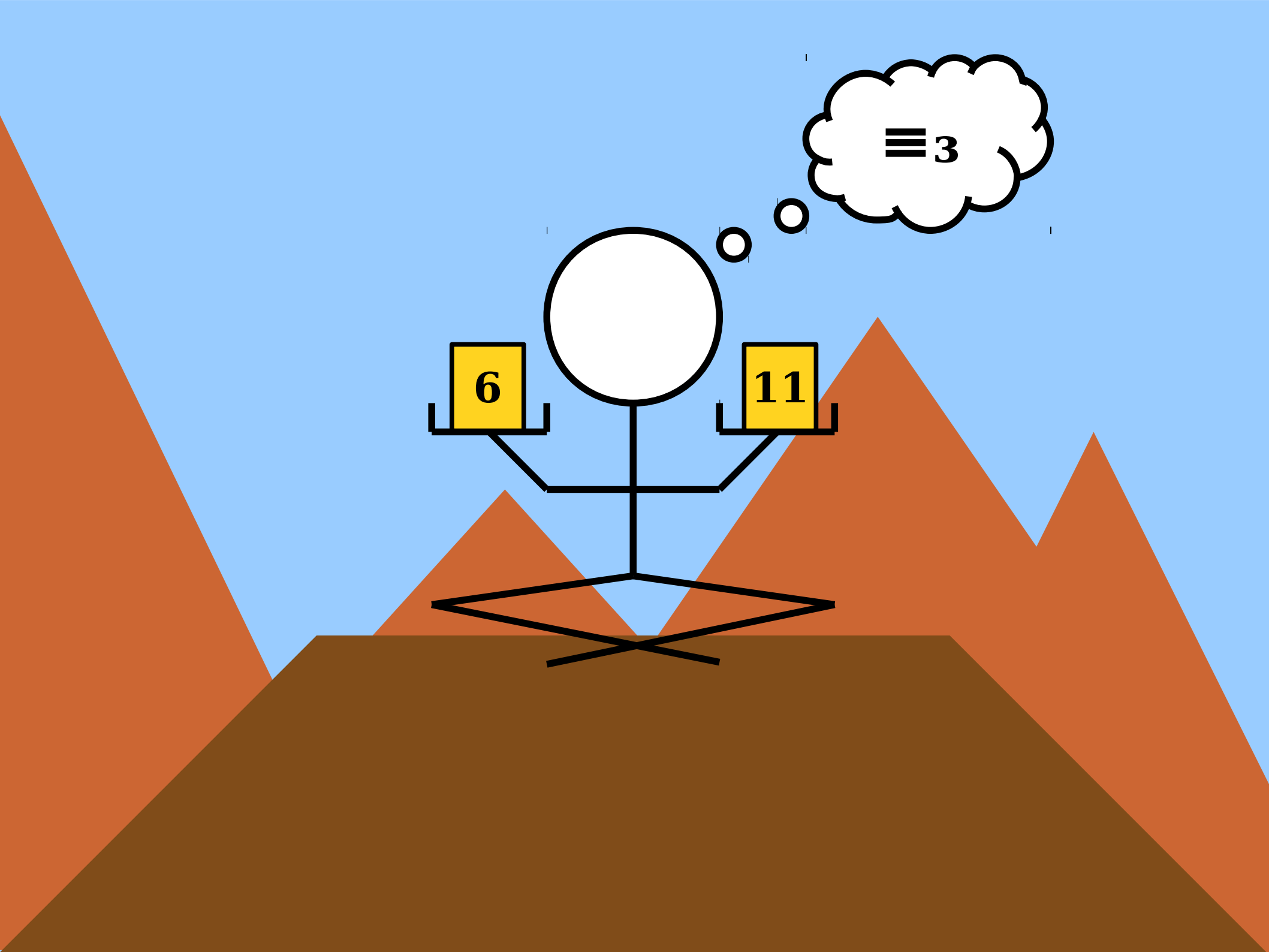
$$\equiv 3$$



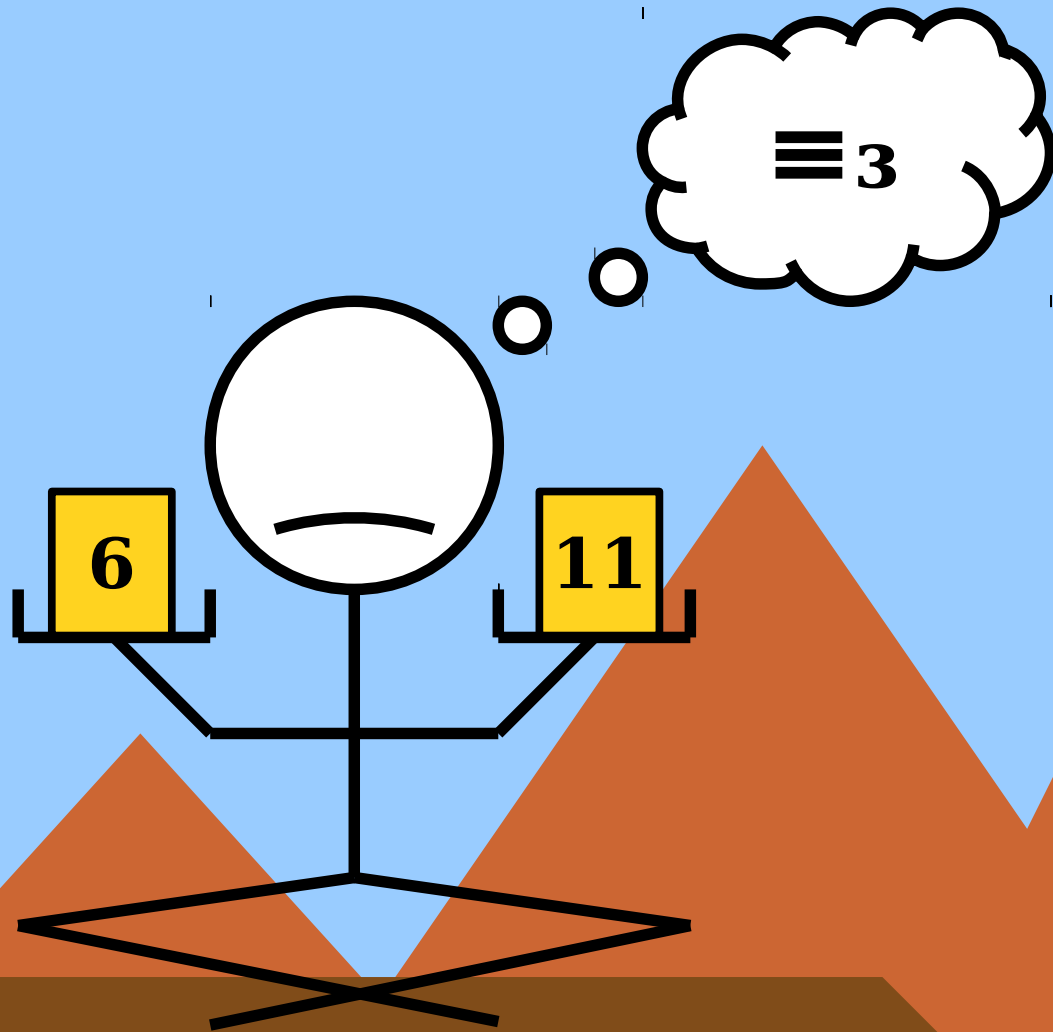
$3=3$



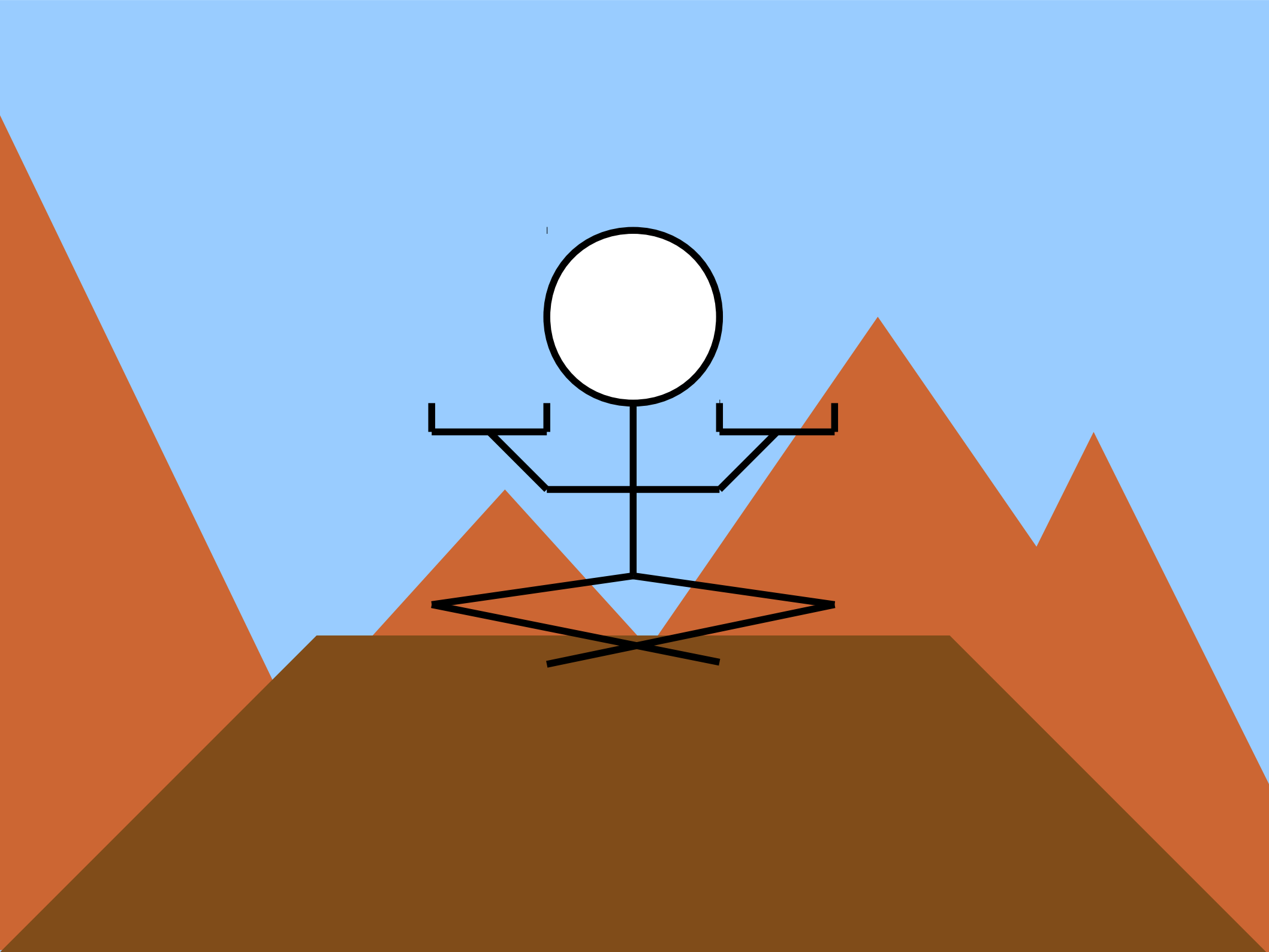
$$7 \equiv_3 10$$

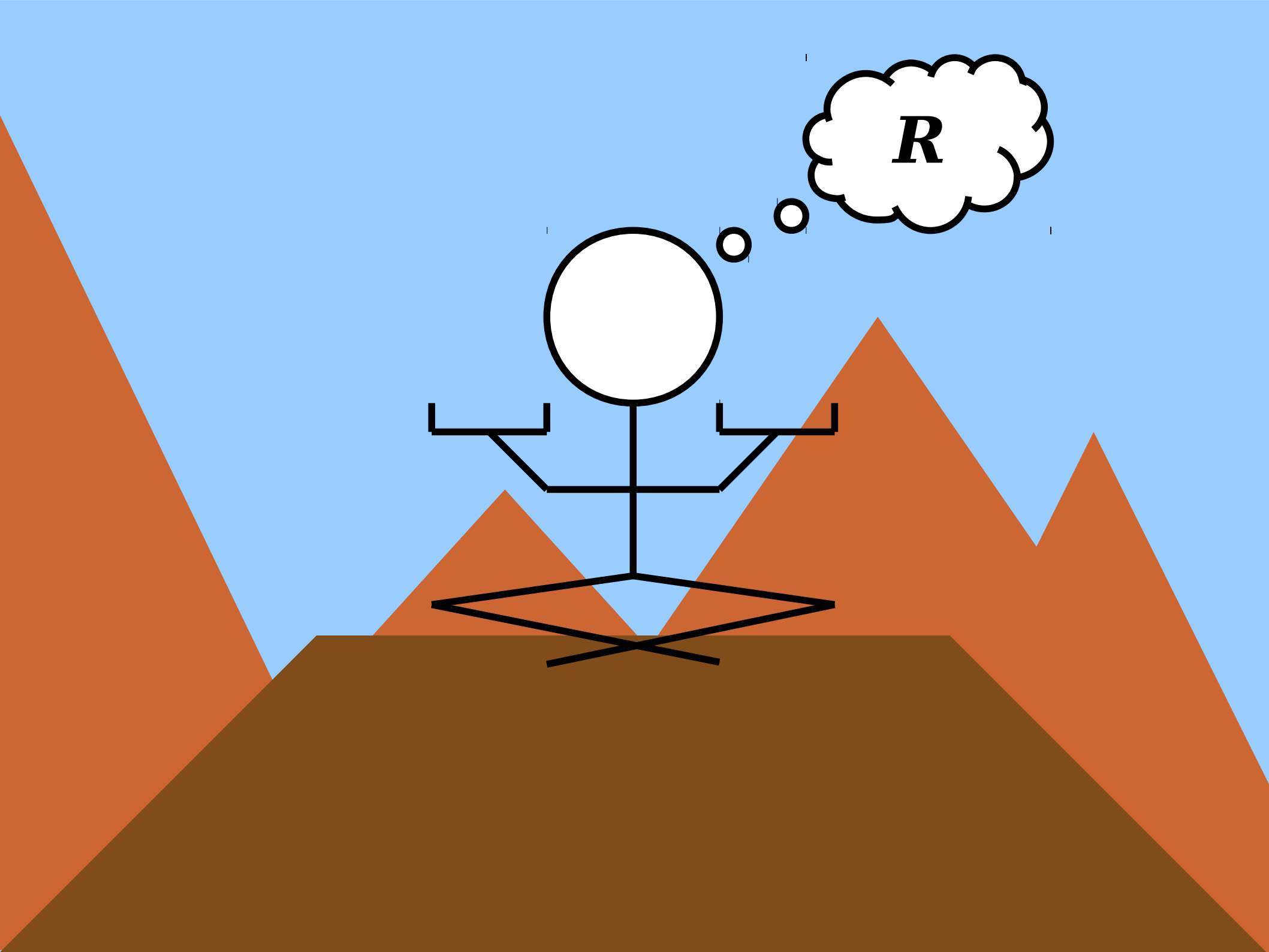


$3=3$

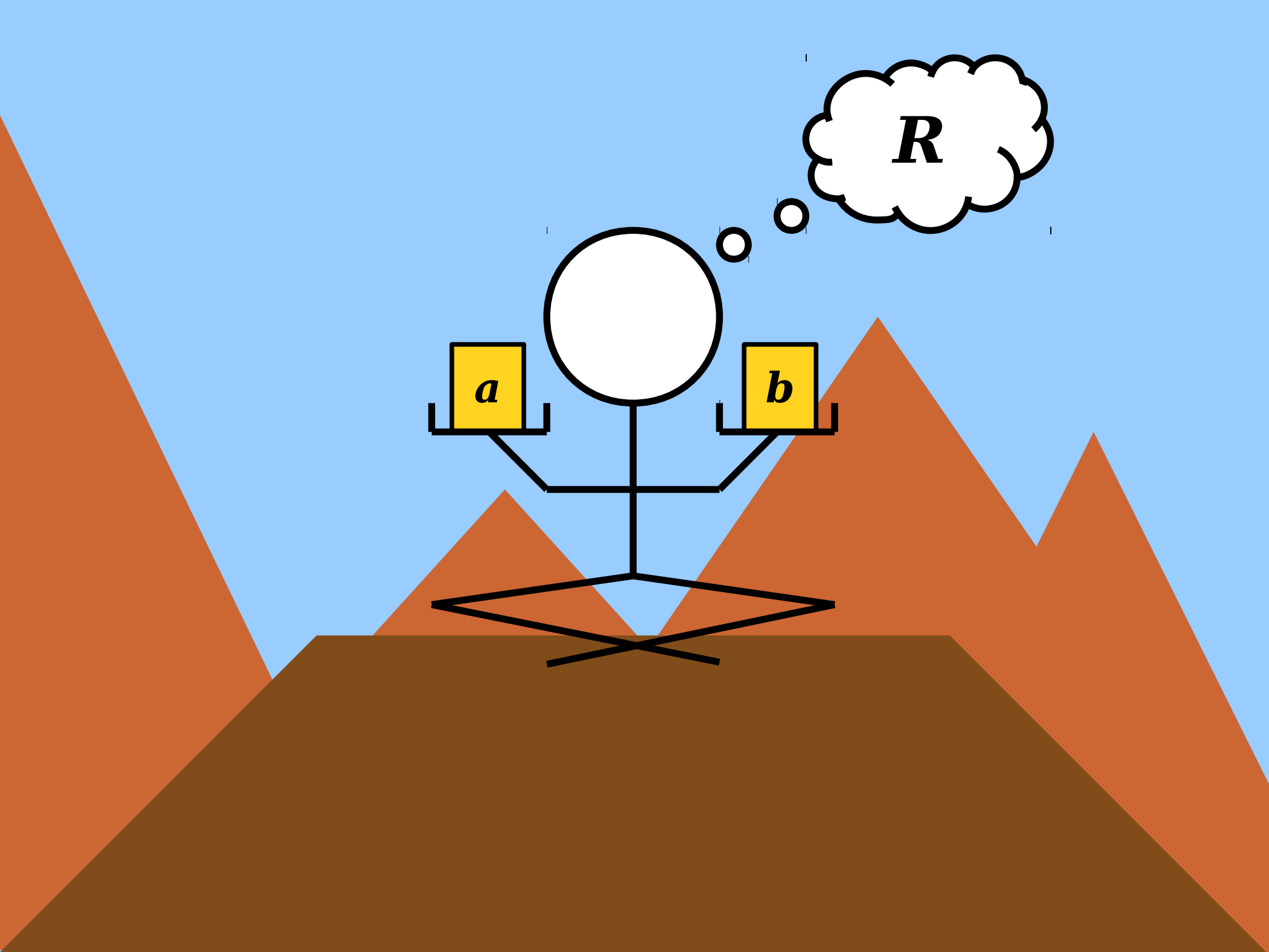


$$6 \neq_3 11$$





*R*

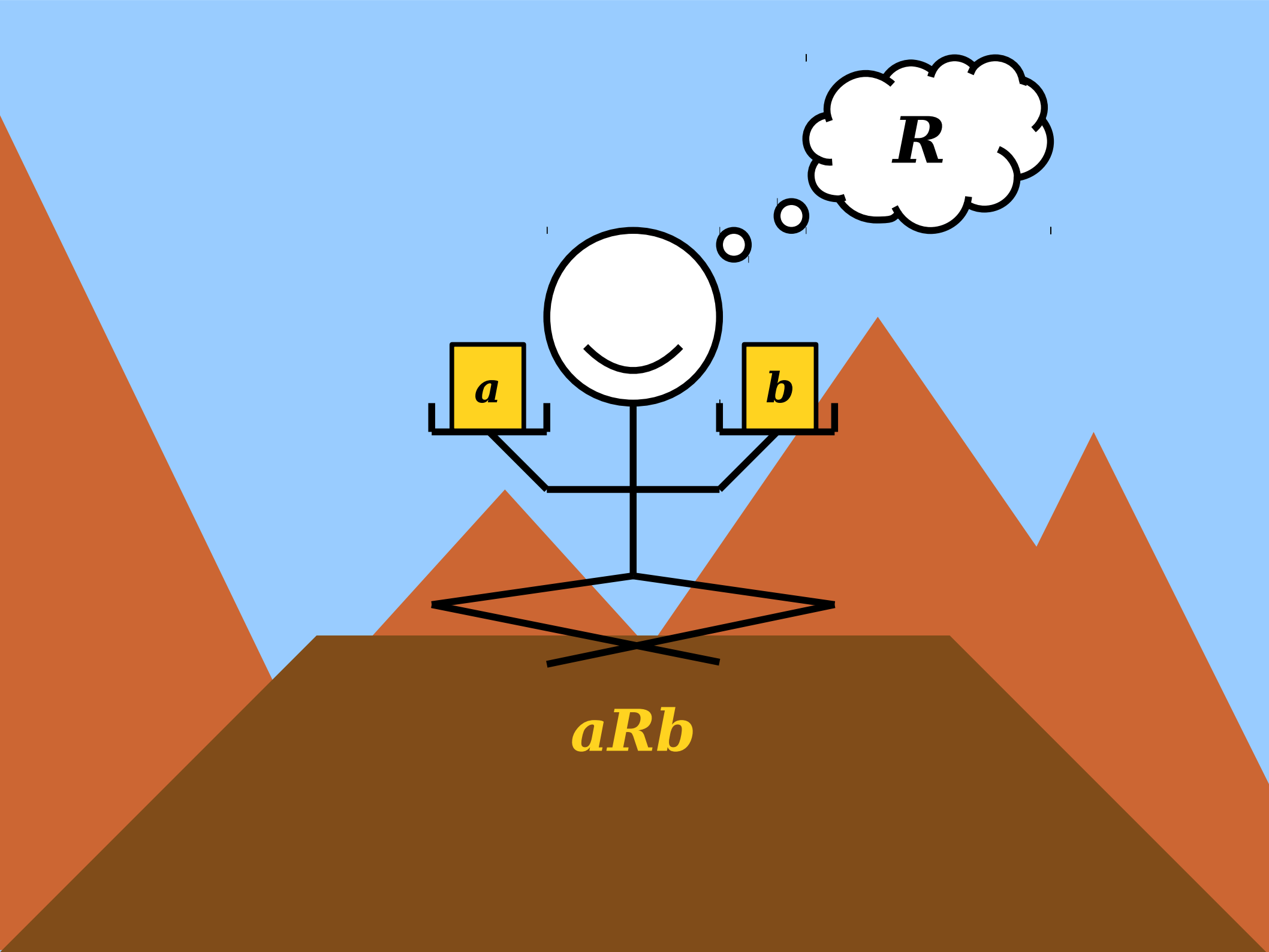


*R*

*a*

*b*



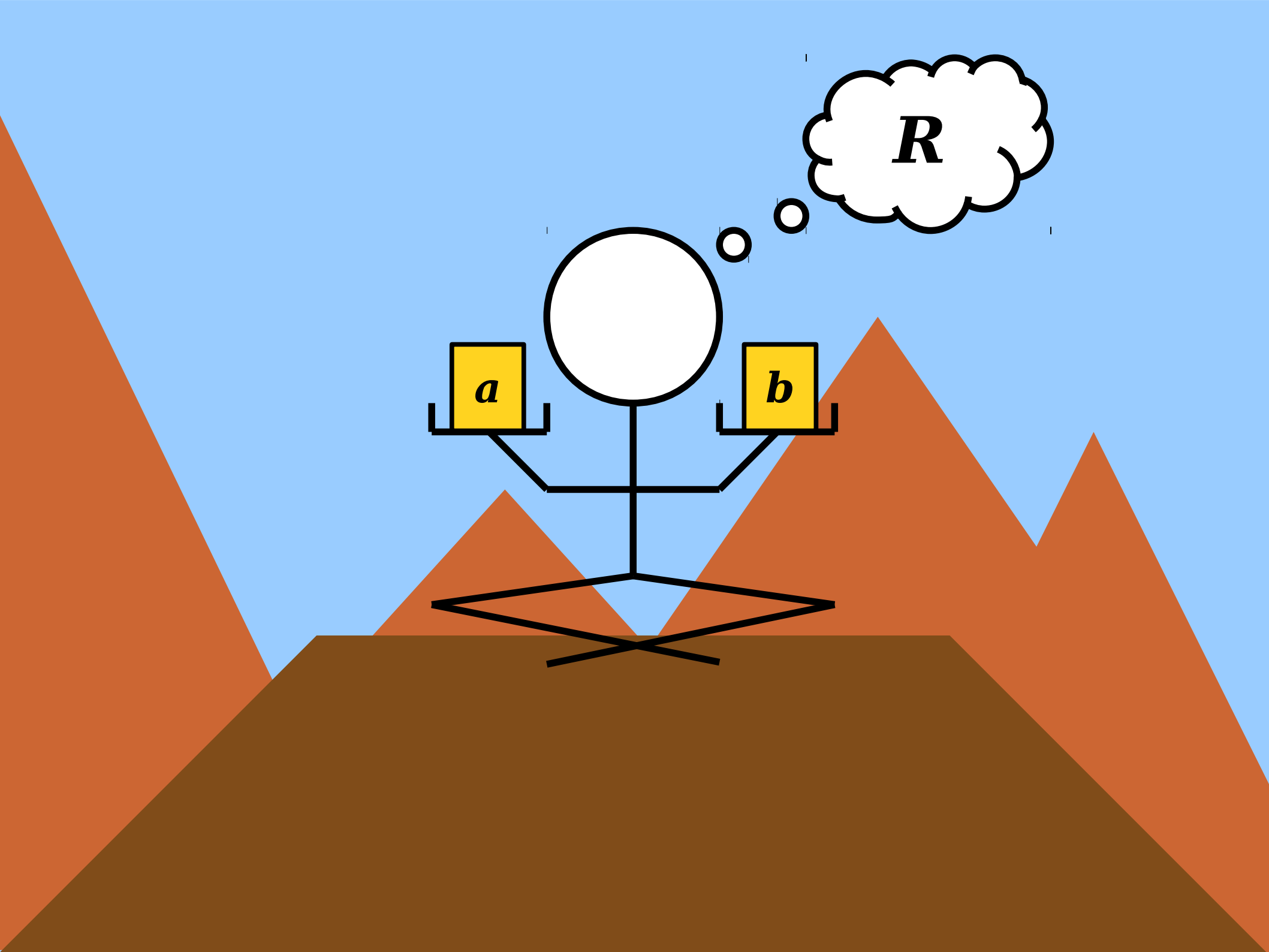


*R*

*a*

*b*

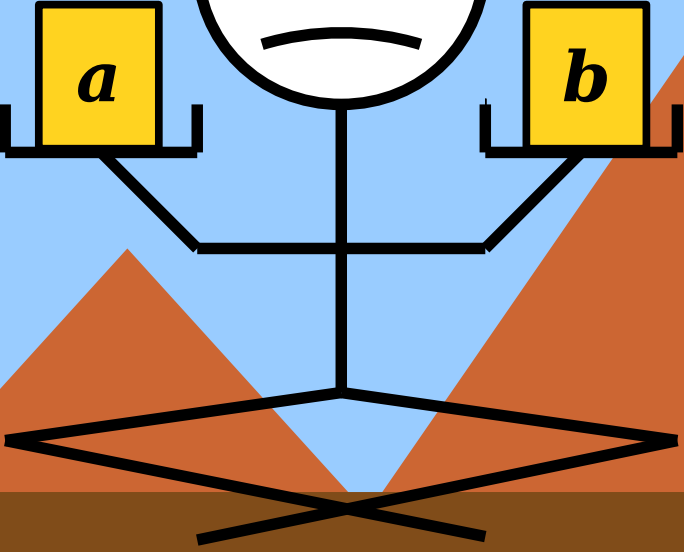
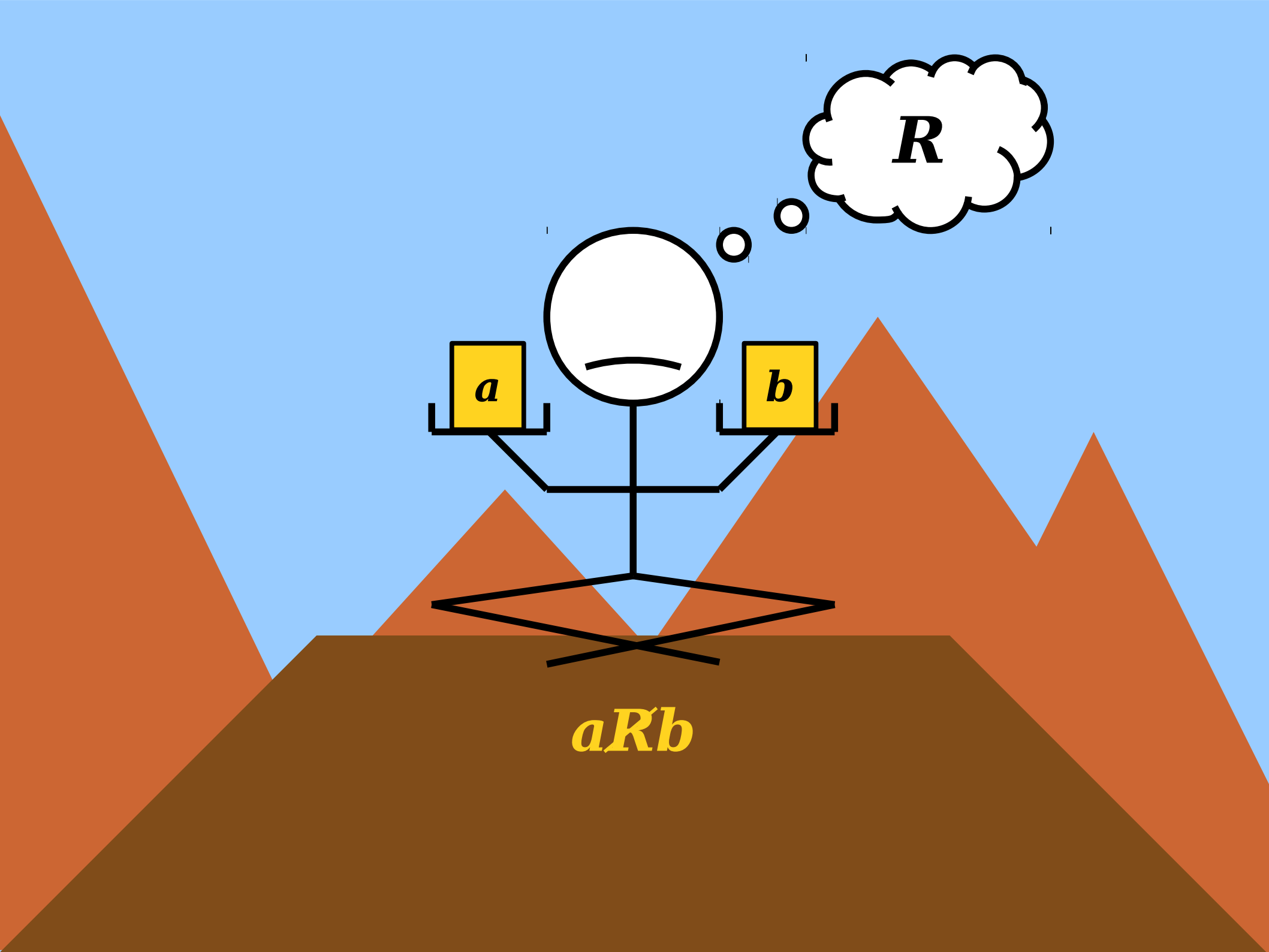
*aRb*



*R*

*a*

*b*



*R*

*a*

*b*

*aRb*

# Binary Relations

- A **binary relation over a set  $A$**  is a predicate  $R$  that can be applied to ordered pairs of elements drawn from  $A$ .
- If  $R$  is a binary relation over  $A$  and it holds for the pair  $(a, b)$ , we write  **$aRb$** .

$$3 = 3$$

$$5 < 7$$

$$\emptyset \subseteq \mathbb{N}$$

- If  $R$  is a binary relation over  $A$  and it does not hold for the pair  $(a, b)$ , we write  **$a \not R b$** .

$$4 \neq 3$$

$$4 \not< 3$$

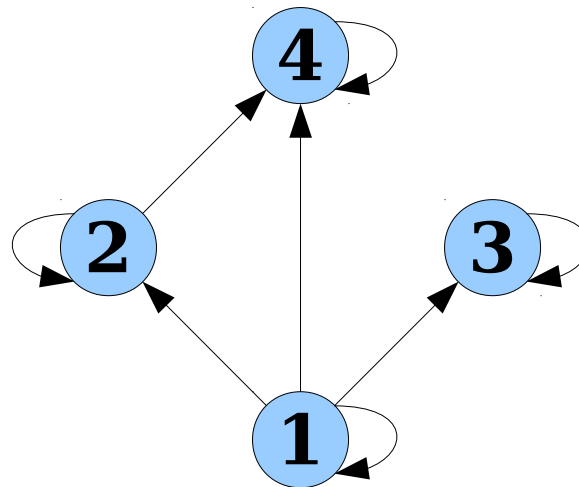
$$\mathbb{N} \not\subseteq \emptyset$$

# Properties of Relations

- Generally speaking, if  $R$  is a binary relation over a set  $A$ , the order of the operands is significant.
  - For example,  $3 < 5$ , but  $5 \not< 3$ .
  - In some relations order is irrelevant; more on that later.
- Relations are always defined relative to some underlying set.
  - It's not meaningful to ask whether  $\odot \subseteq 15$ , for example, since  $\subseteq$  is defined over sets, not arbitrary objects.

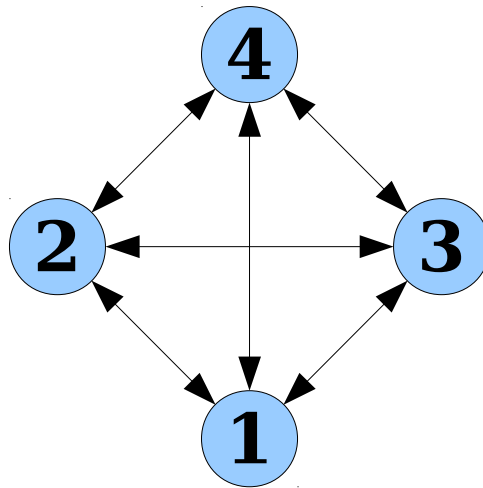
# Visualizing Relations

- We can visualize a binary relation  $R$  over a set  $A$  by drawing the elements of  $A$  and drawing an arrow between an element  $a$  and an element  $b$  if  $aRb$  is true.
- Example: the relation  $a \mid b$  (meaning “ $a$  divides  $b$ ”) over the set  $\{1, 2, 3, 4\}$  looks like this:



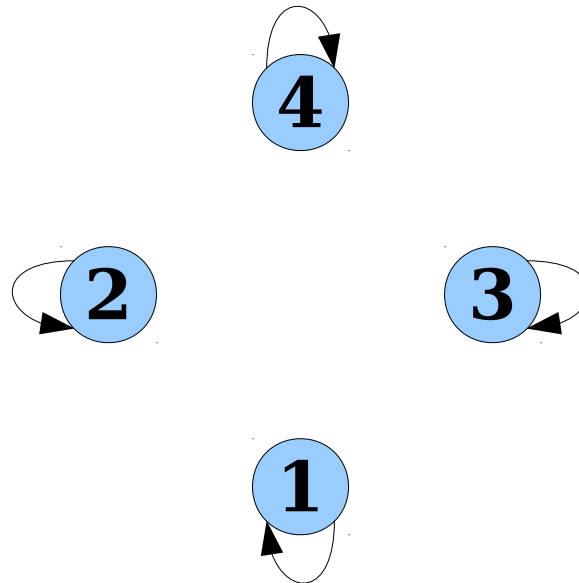
# Visualizing Relations

- We can visualize a binary relation  $R$  over a set  $A$  by drawing the elements of  $A$  and drawing an arrow between an element  $a$  and an element  $b$  if  $aRb$  is true.
- Example: the relation  $a \neq b$  over the set  $\{1, 2, 3, 4\}$  looks like this:



# Visualizing Relations

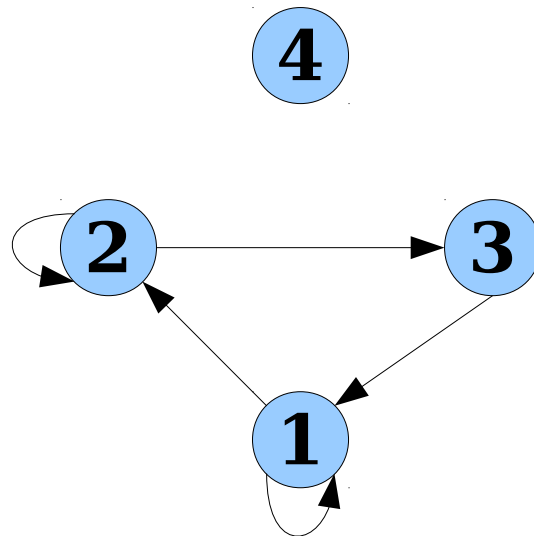
- We can visualize a binary relation  $R$  over a set  $A$  by drawing the elements of  $A$  and drawing an arrow between an element  $a$  and an element  $b$  if  $aRb$  is true.
- Example: the relation  $a = b$  over the set  $\{1, 2, 3, 4\}$  looks like this:





# Visualizing Relations

- We can visualize a binary relation  $R$  over a set  $A$  by drawing the elements of  $A$  and drawing an arrow between an element  $a$  and an element  $b$  if  $aRb$  is true.
- Example: below is some relation over  $\{1, 2, 3, 4\}$  that's a totally valid relation even though there doesn't appear to be a simple unifying rule.

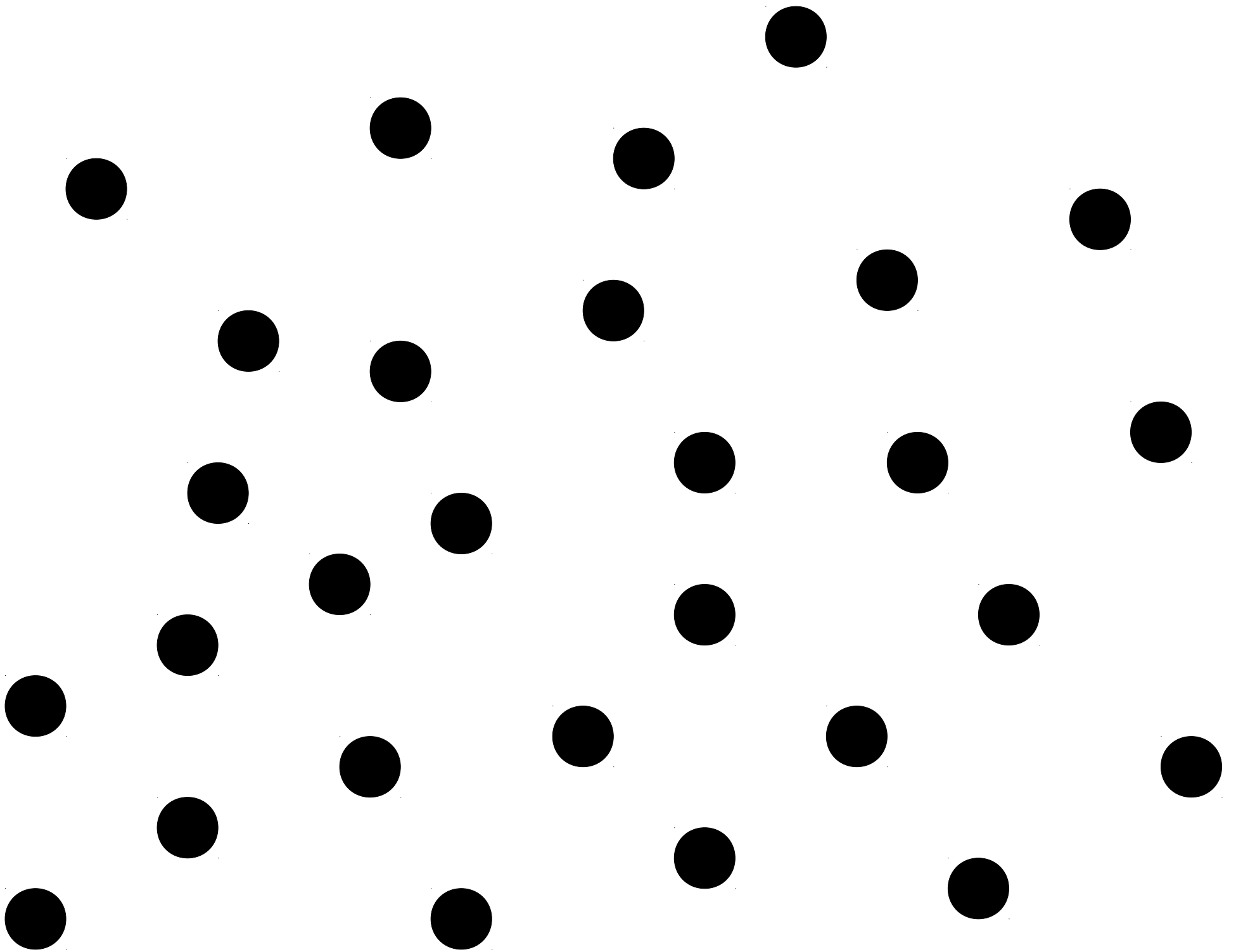


# Capturing Structure

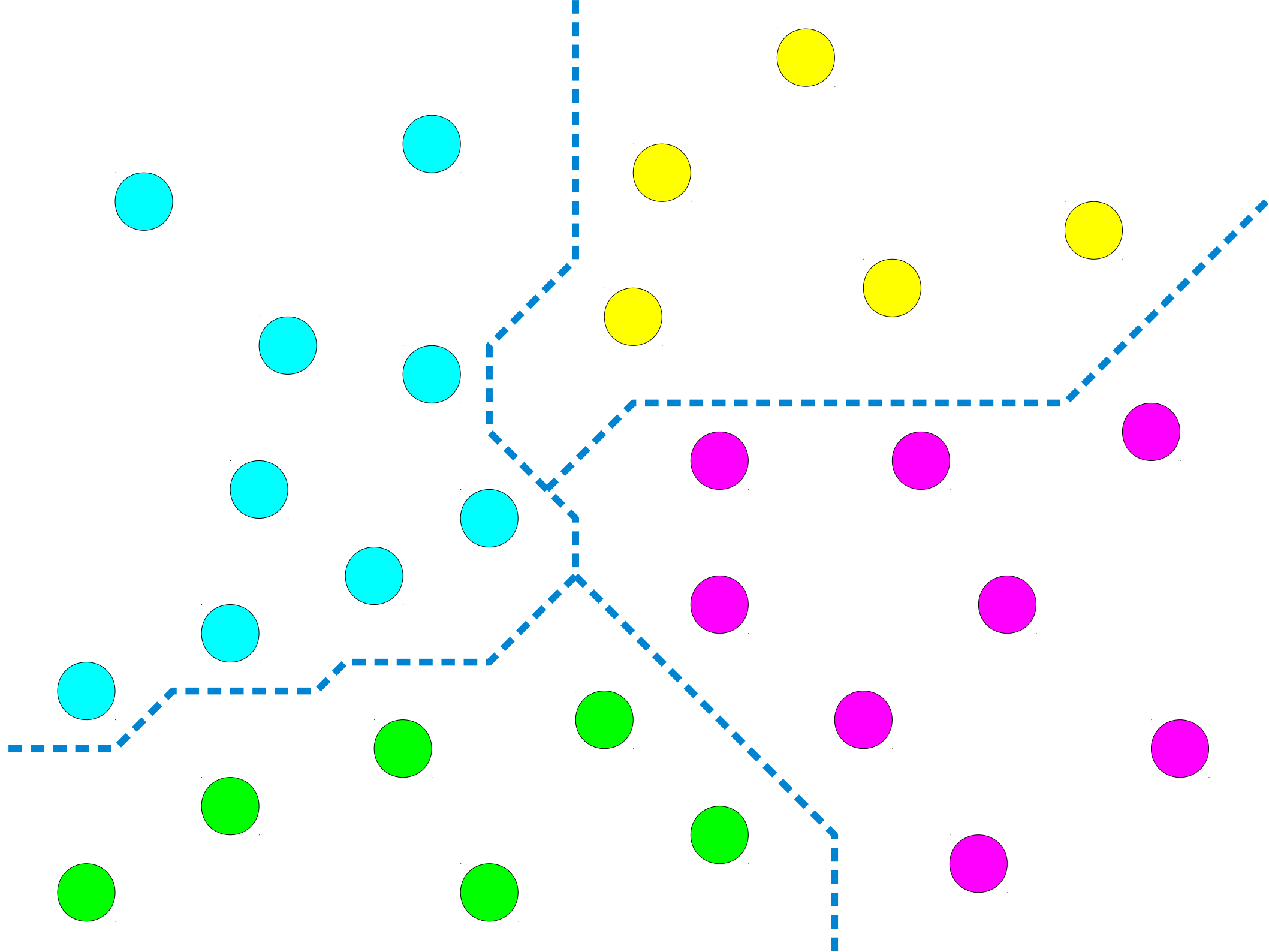
# Capturing Structure

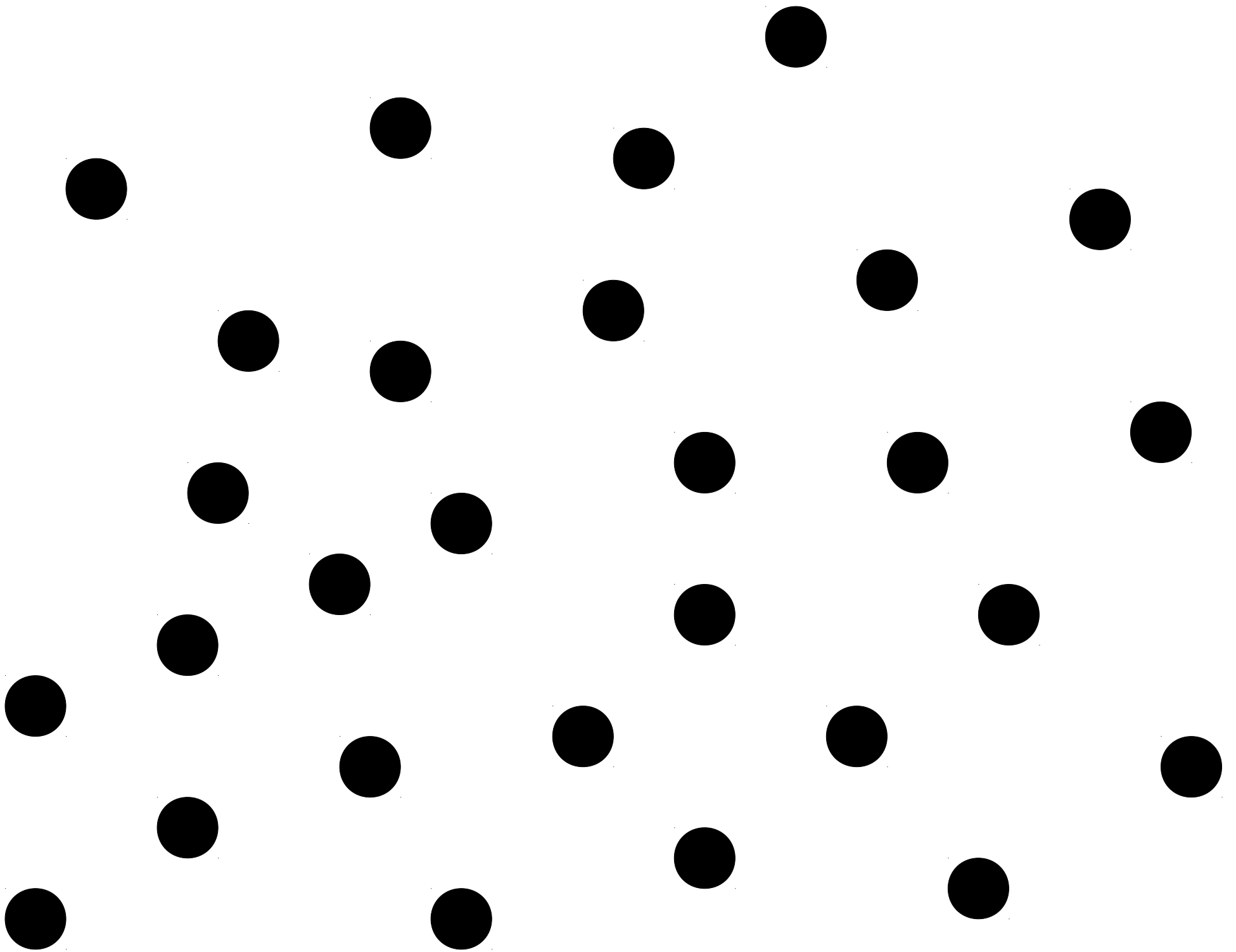
- Binary relations are an excellent way for capturing certain structures that appear in computer science.
- Today, we'll look at one of them (***partitions***), and next time we'll see another (***prerequisites***).
- Along the way, we'll explore how to write proofs about definitions given in first-order logic.

# Partitions

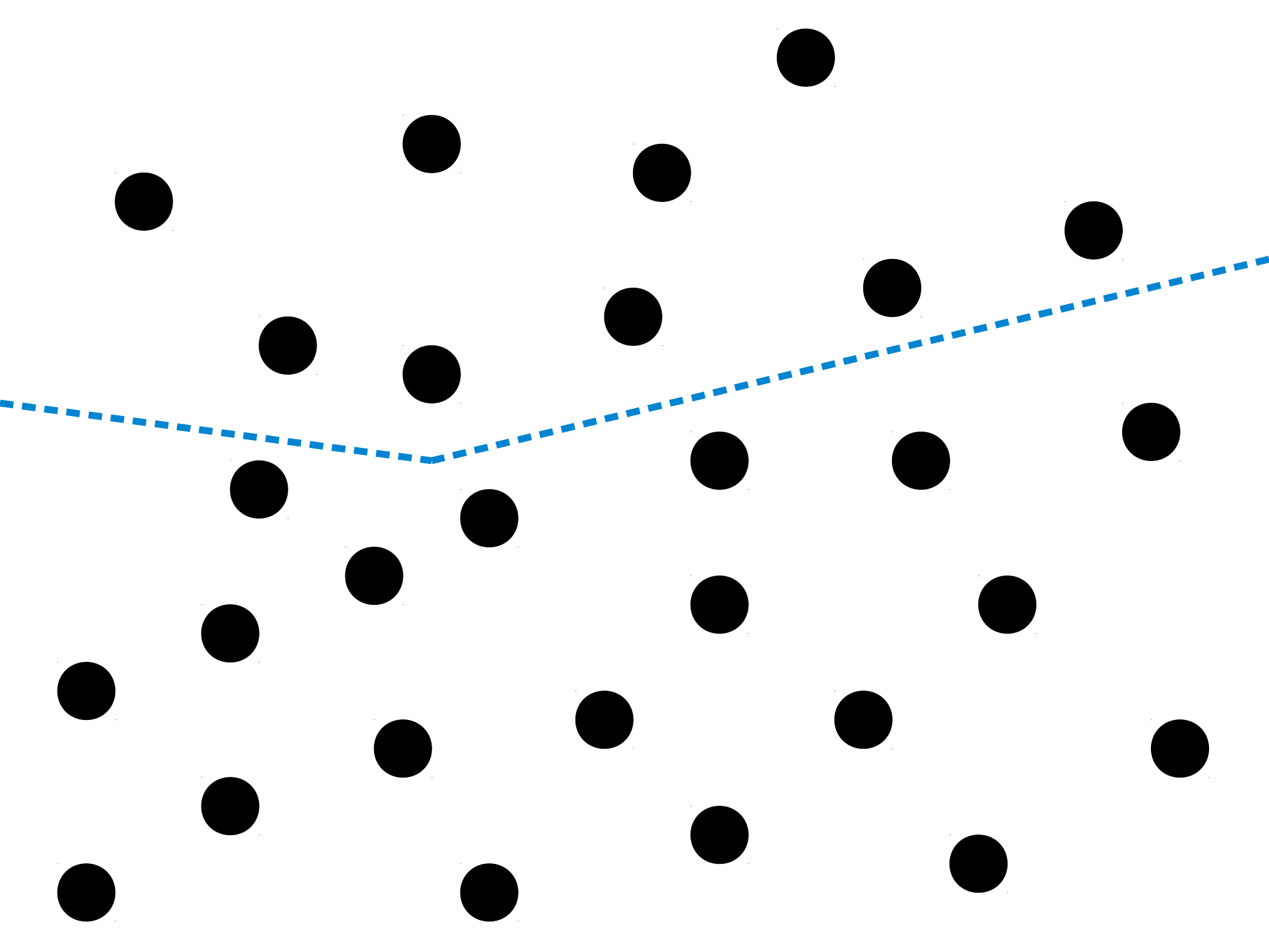


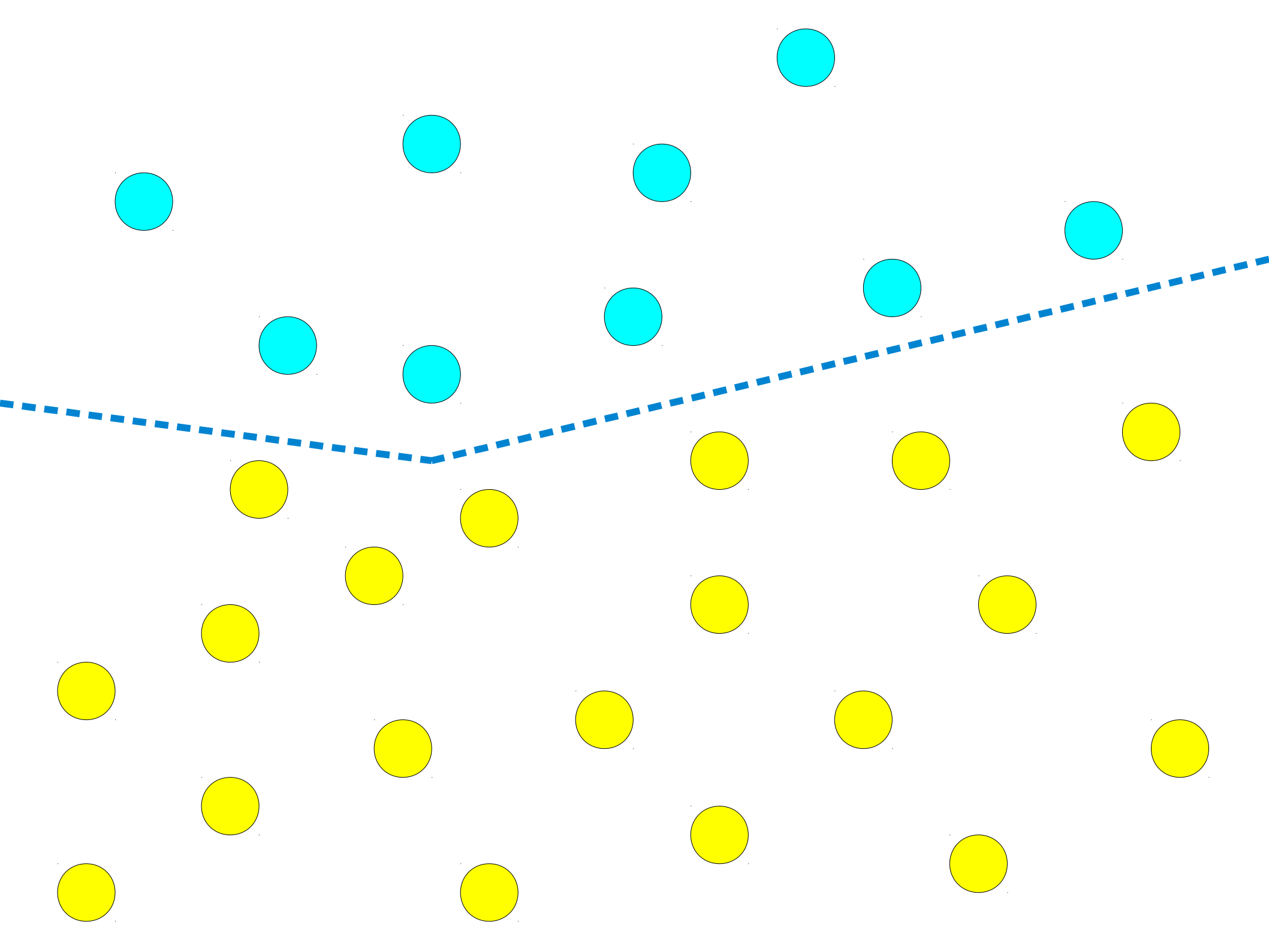


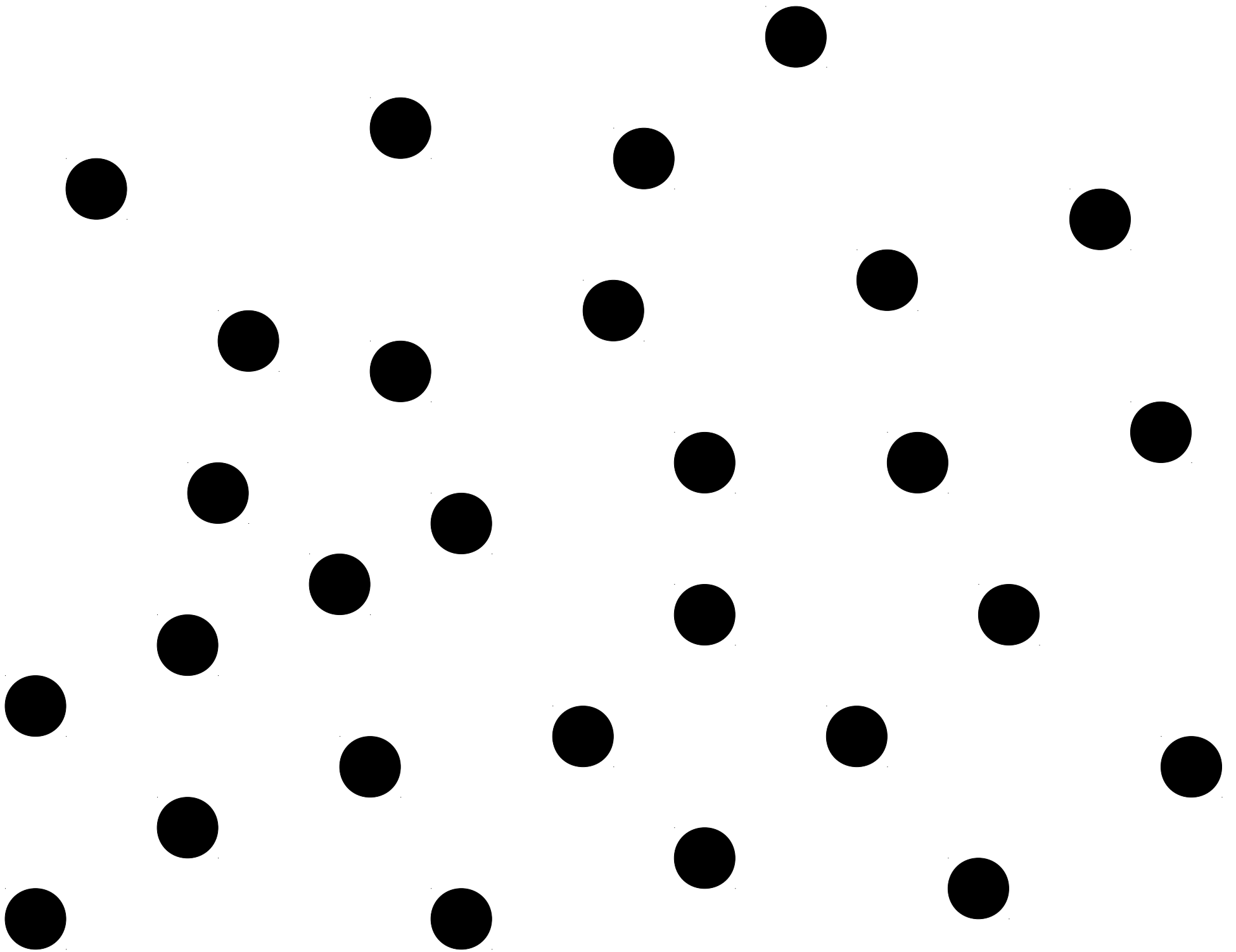


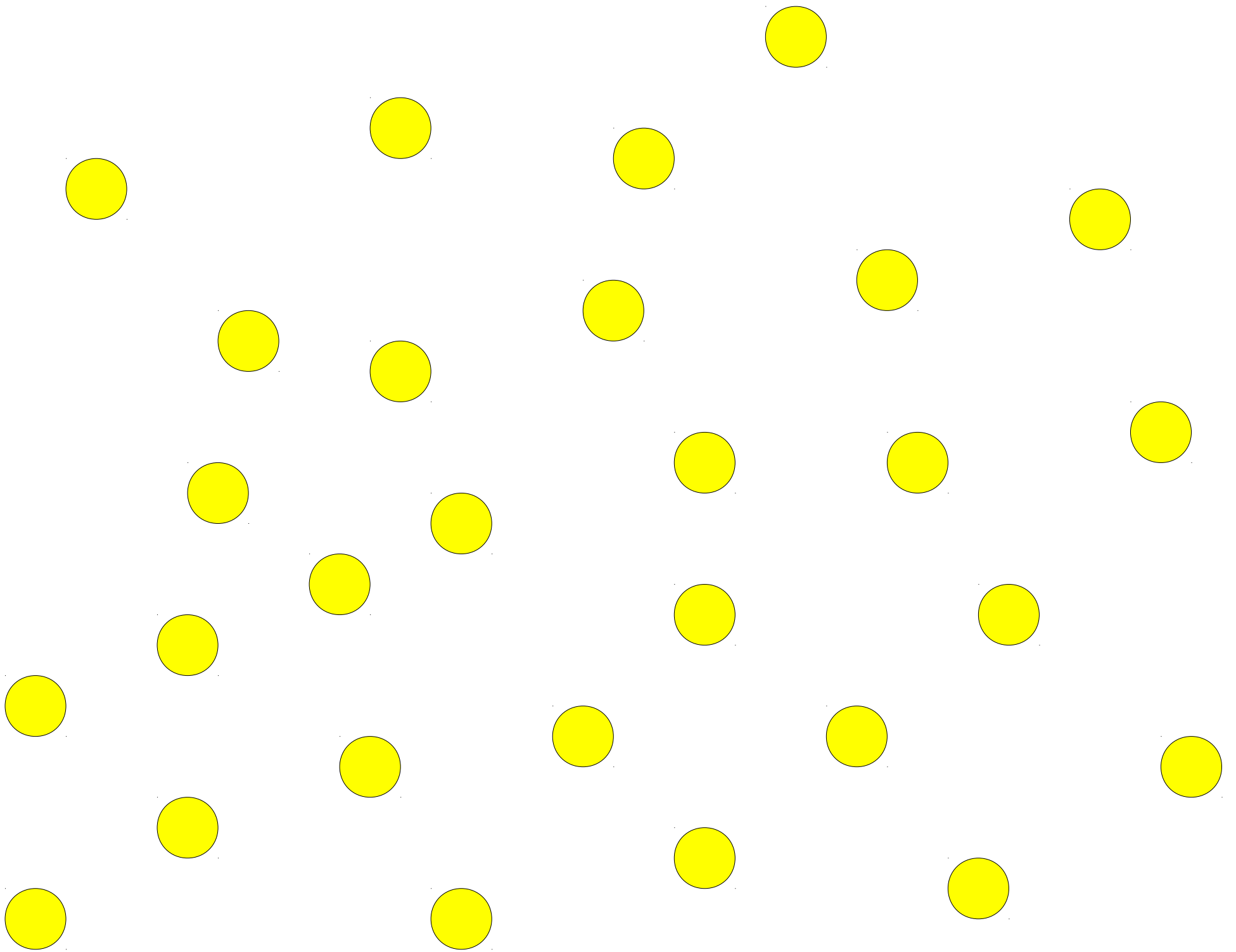


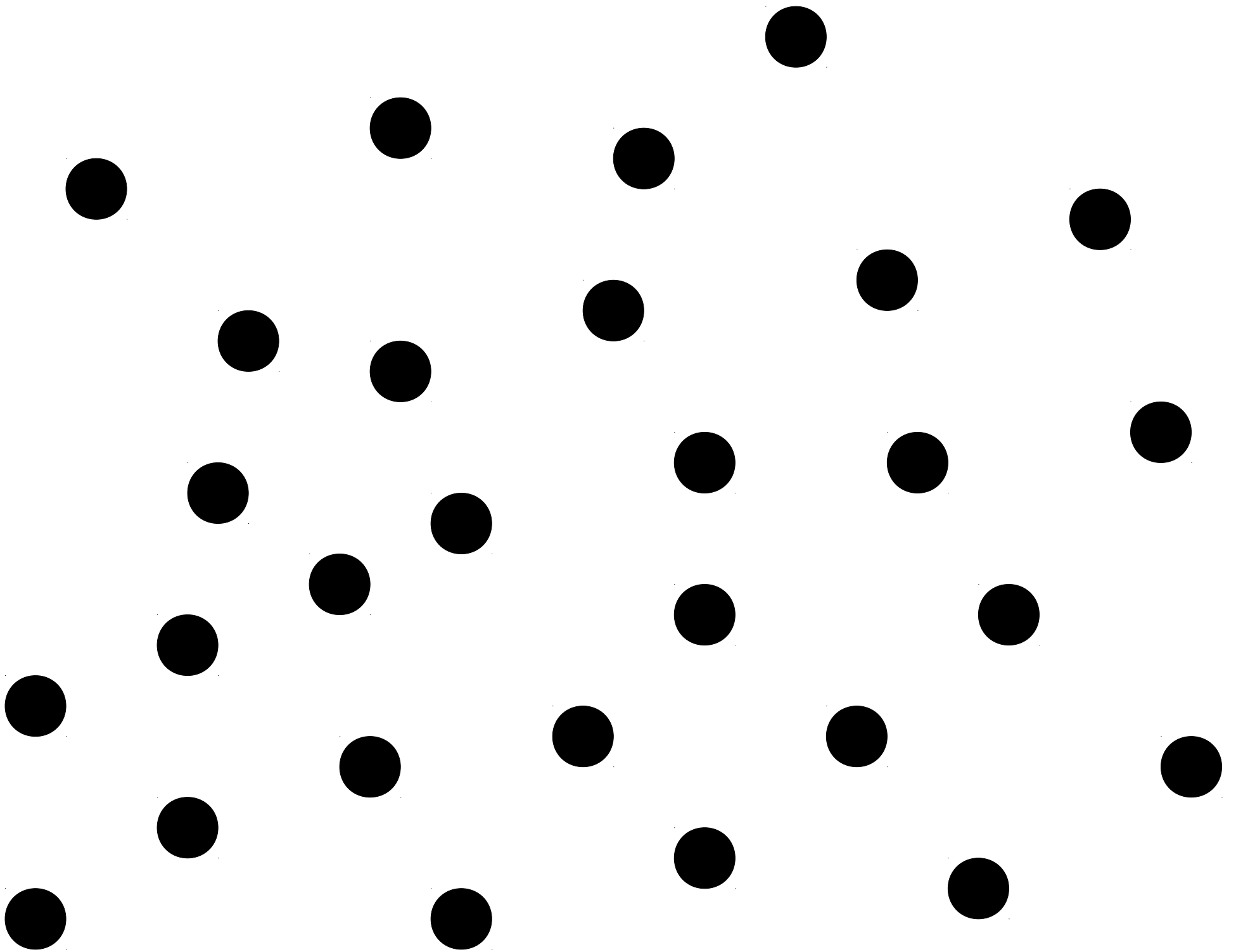


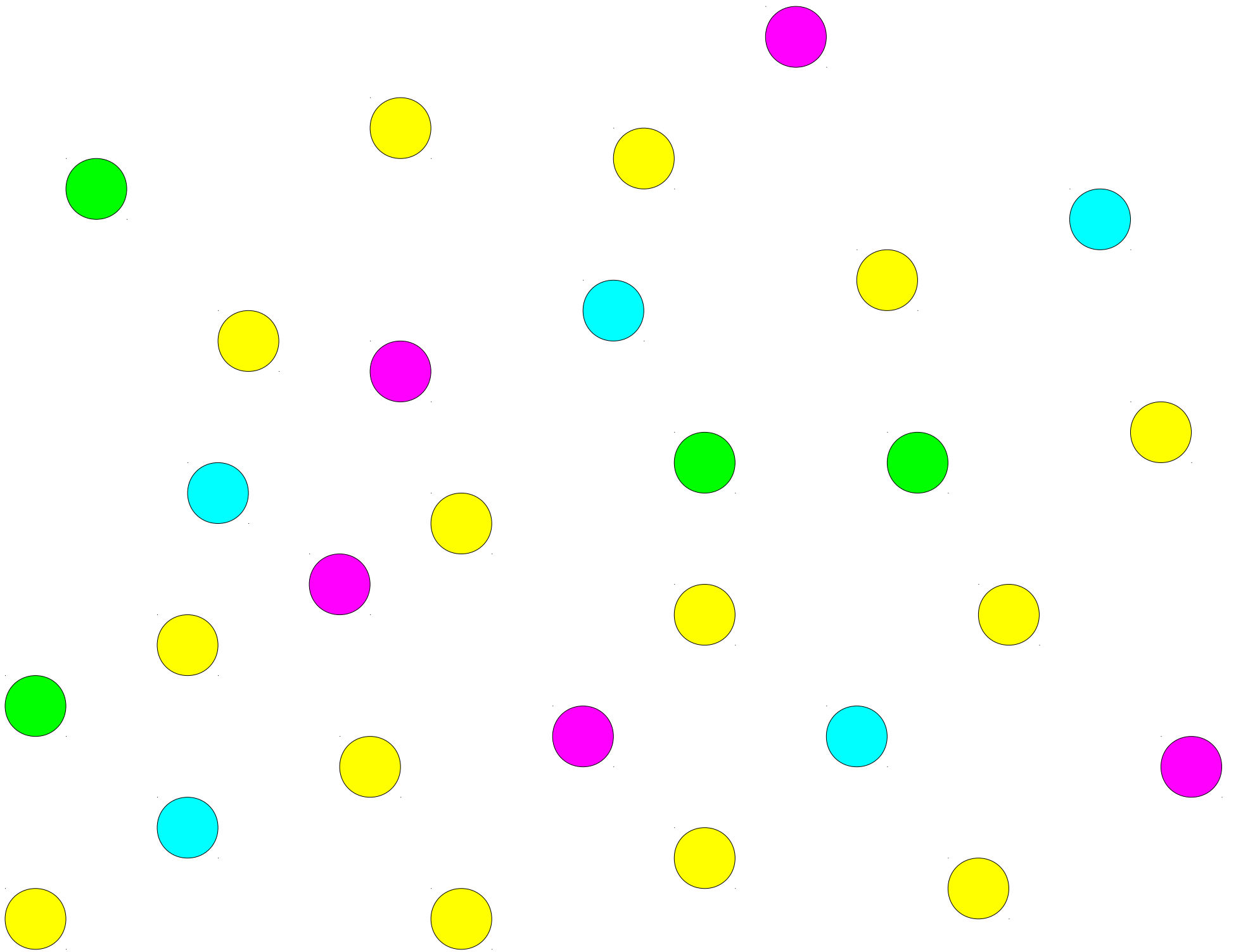












# Partitions

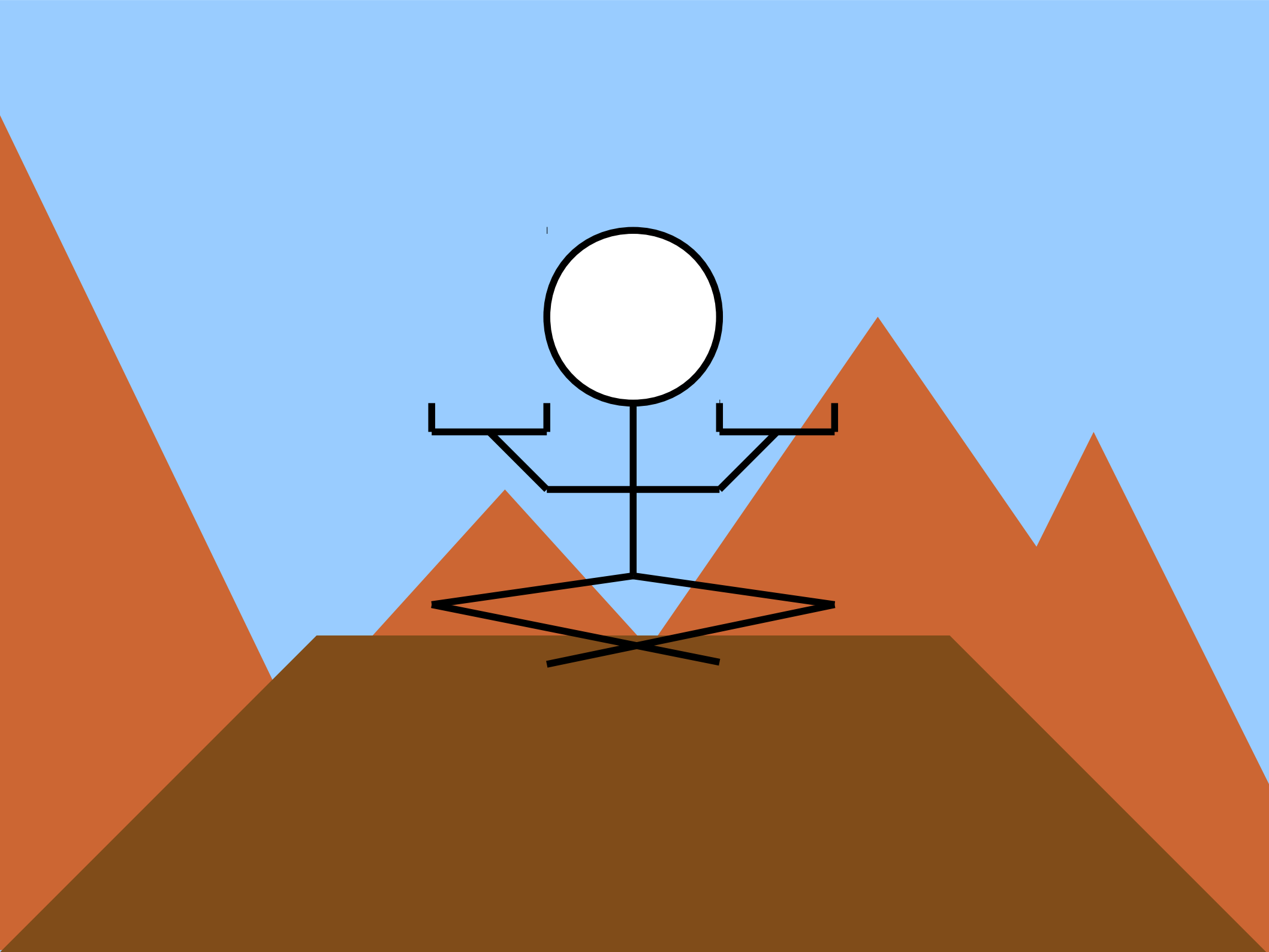
- A ***partition of a set*** is a way of splitting the set into disjoint, nonempty subsets so that every element belongs to exactly one subset.
  - Two sets are ***disjoint*** if their intersection is the empty set; formally, sets  $S$  and  $T$  are disjoint if  $S \cap T = \emptyset$ .
- Intuitively, a partition of a set breaks the set apart into smaller pieces.
- There doesn't have to be any rhyme or reason to what those pieces are, though often there is one.

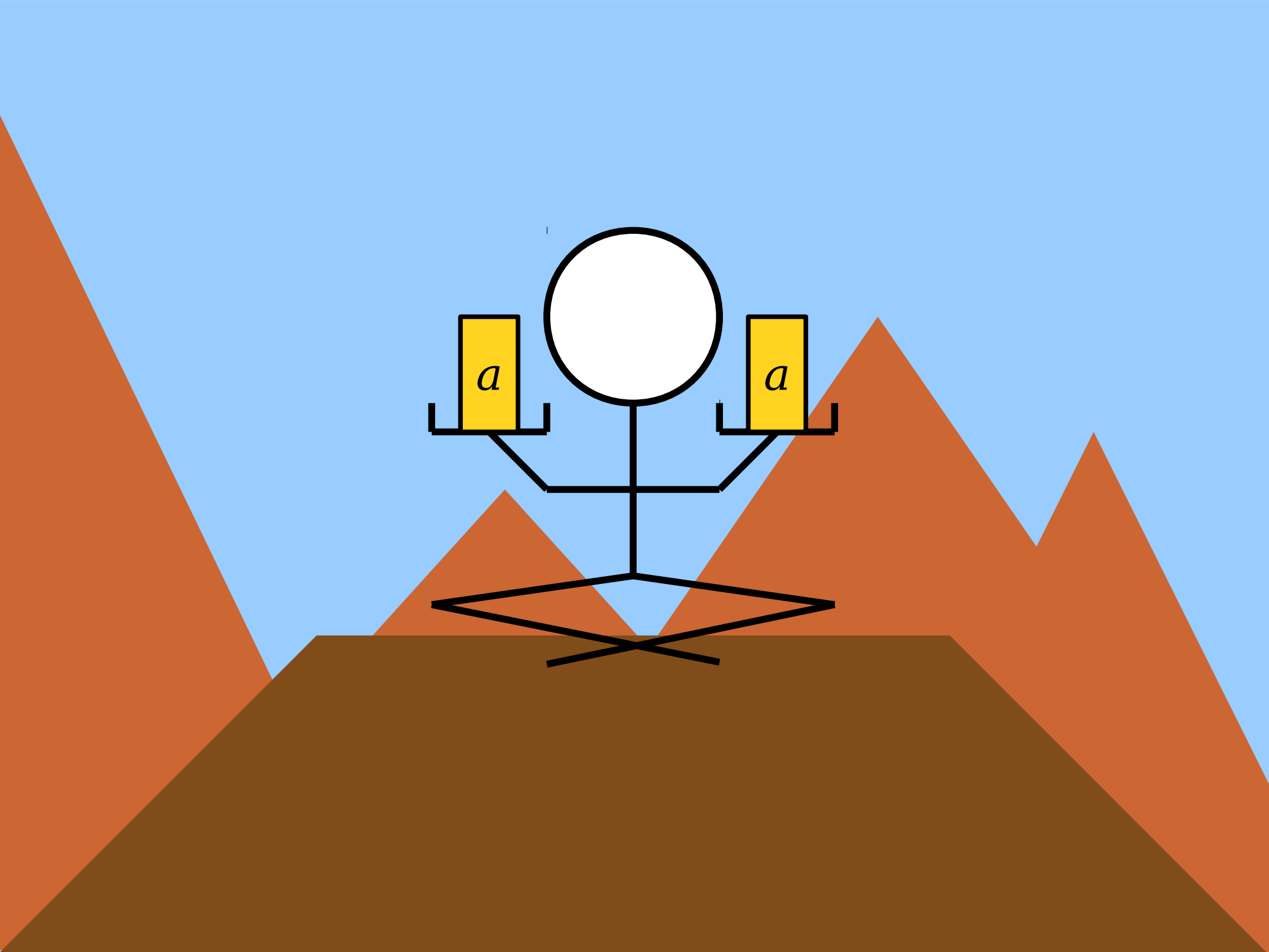
# Partitions and Clustering

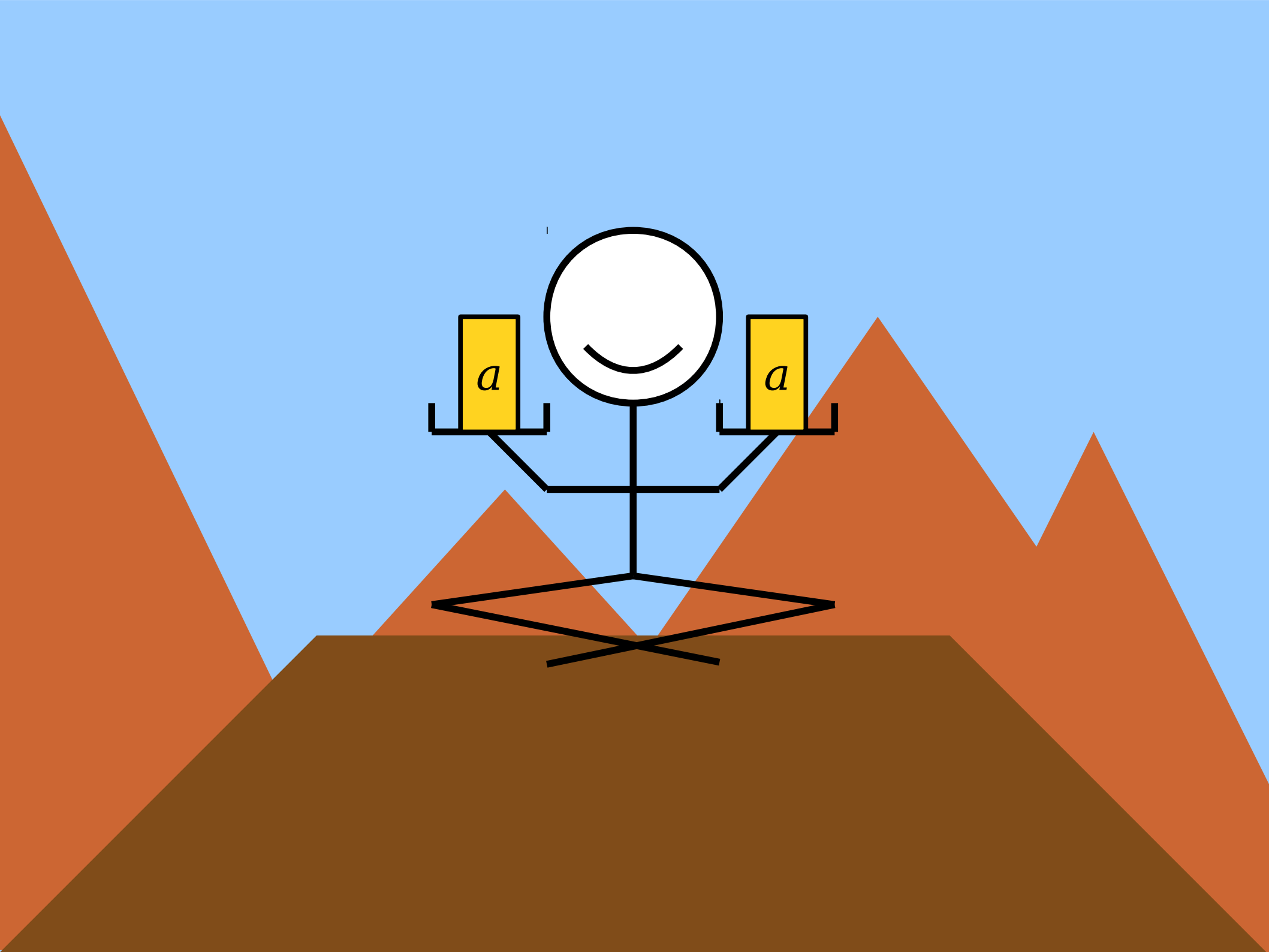
- If you have a set of data, you can often learn something from the data by finding a “good” partition of that data and inspecting the partitions.
  - Usually, the term ***clustering*** is used in data analysis rather than *partitioning*.
- Interested to learn more? Take CS161 or CS246!

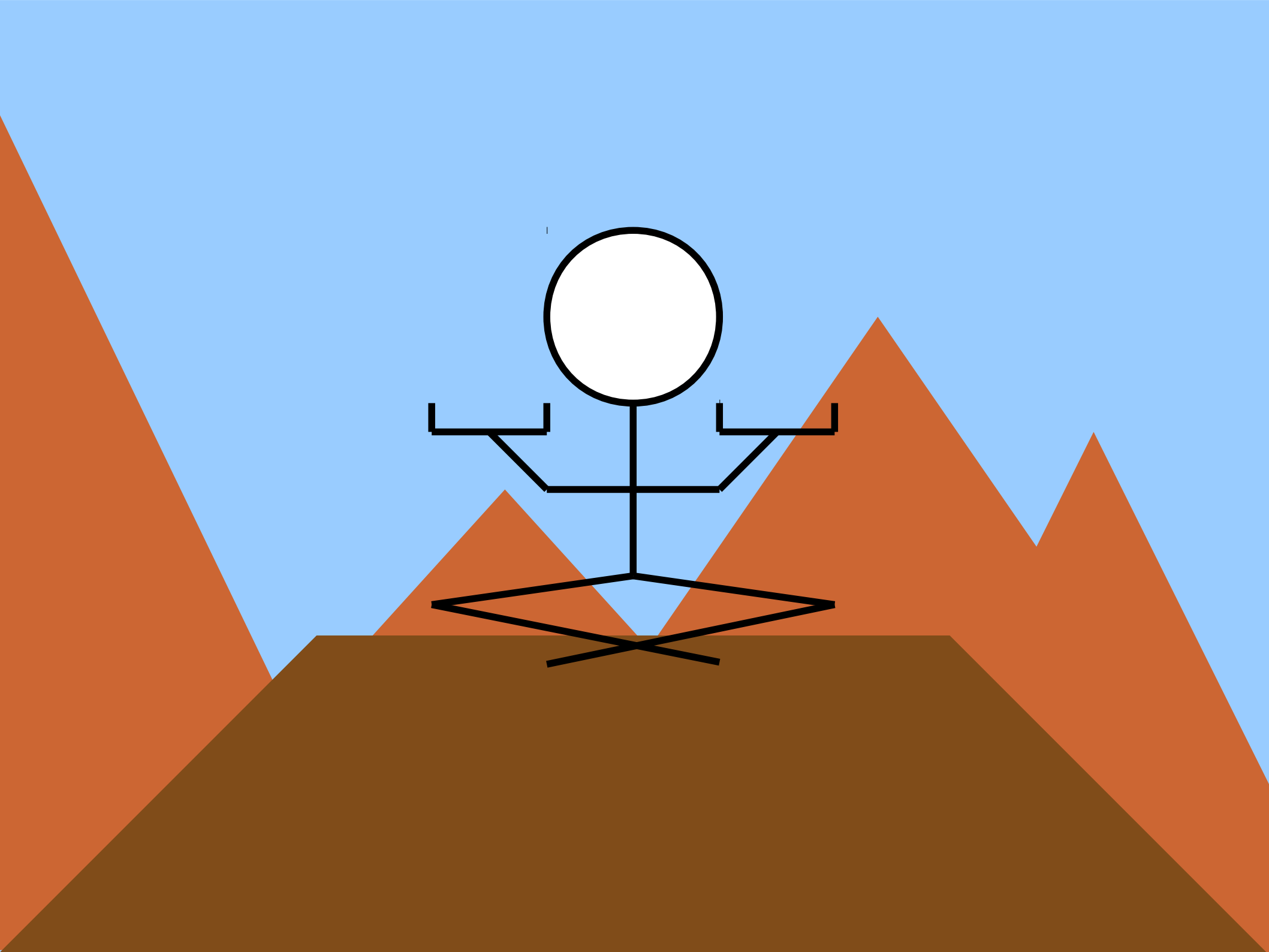


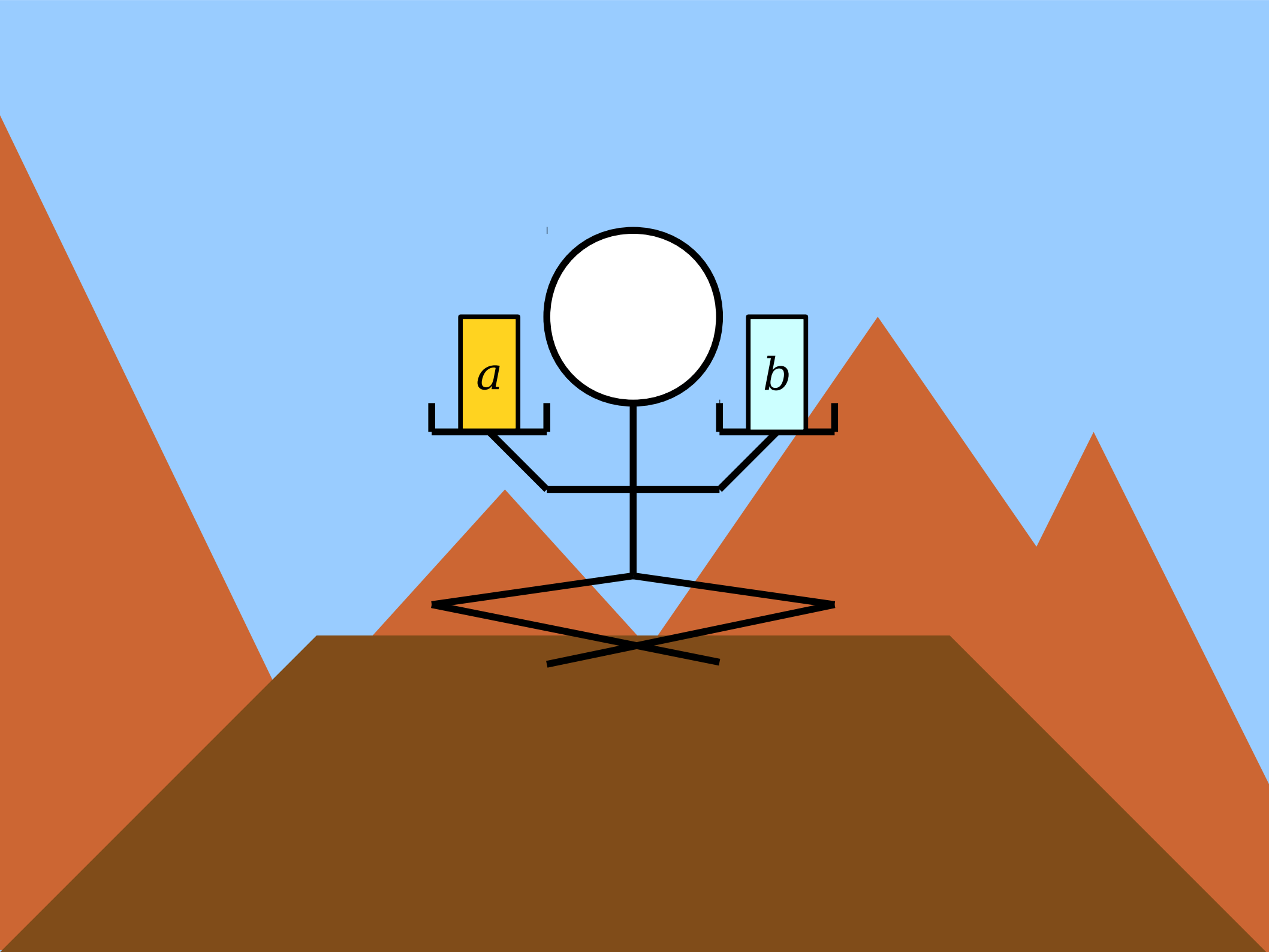
What's the connection between partitions  
and binary relations?

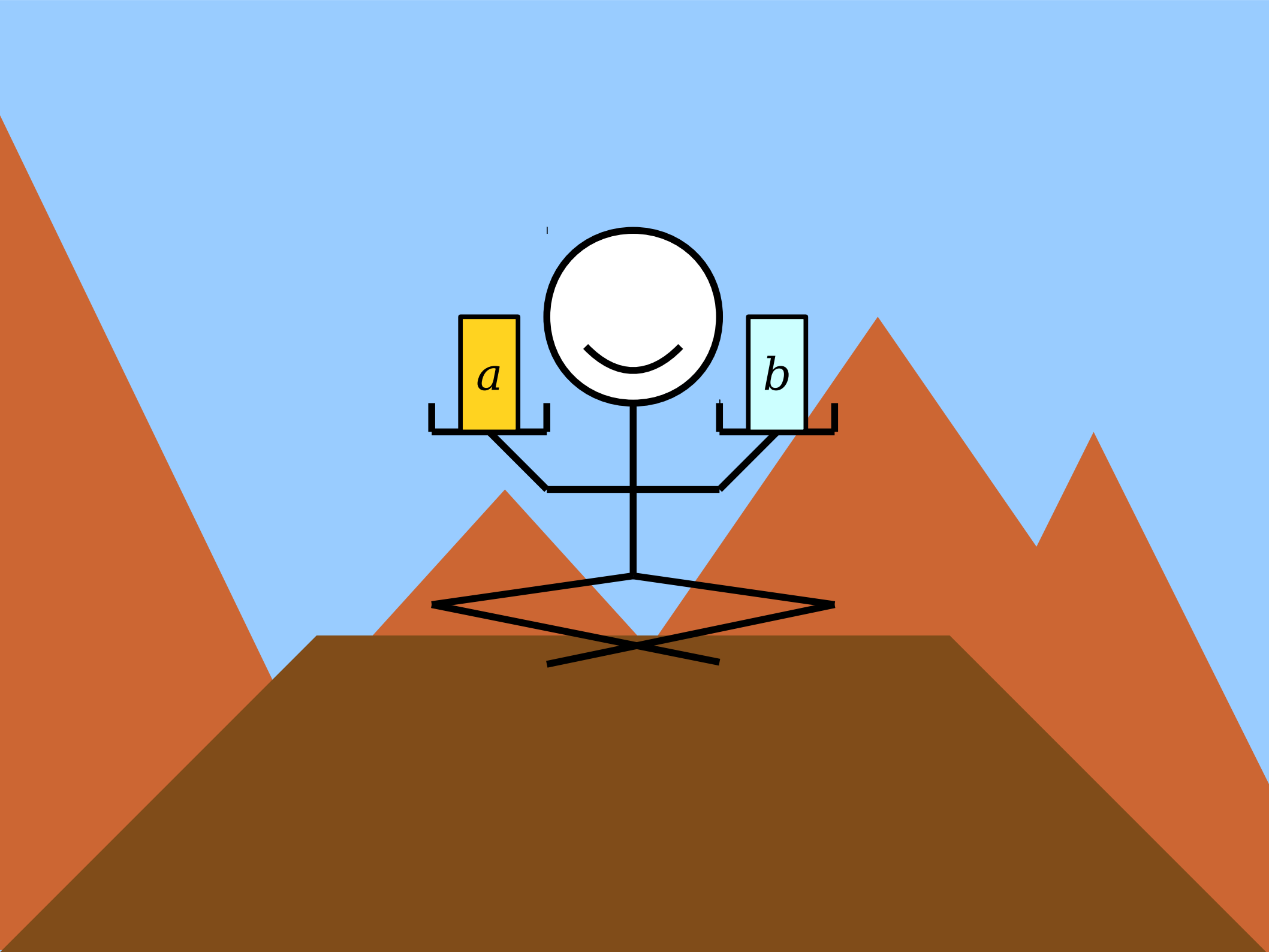


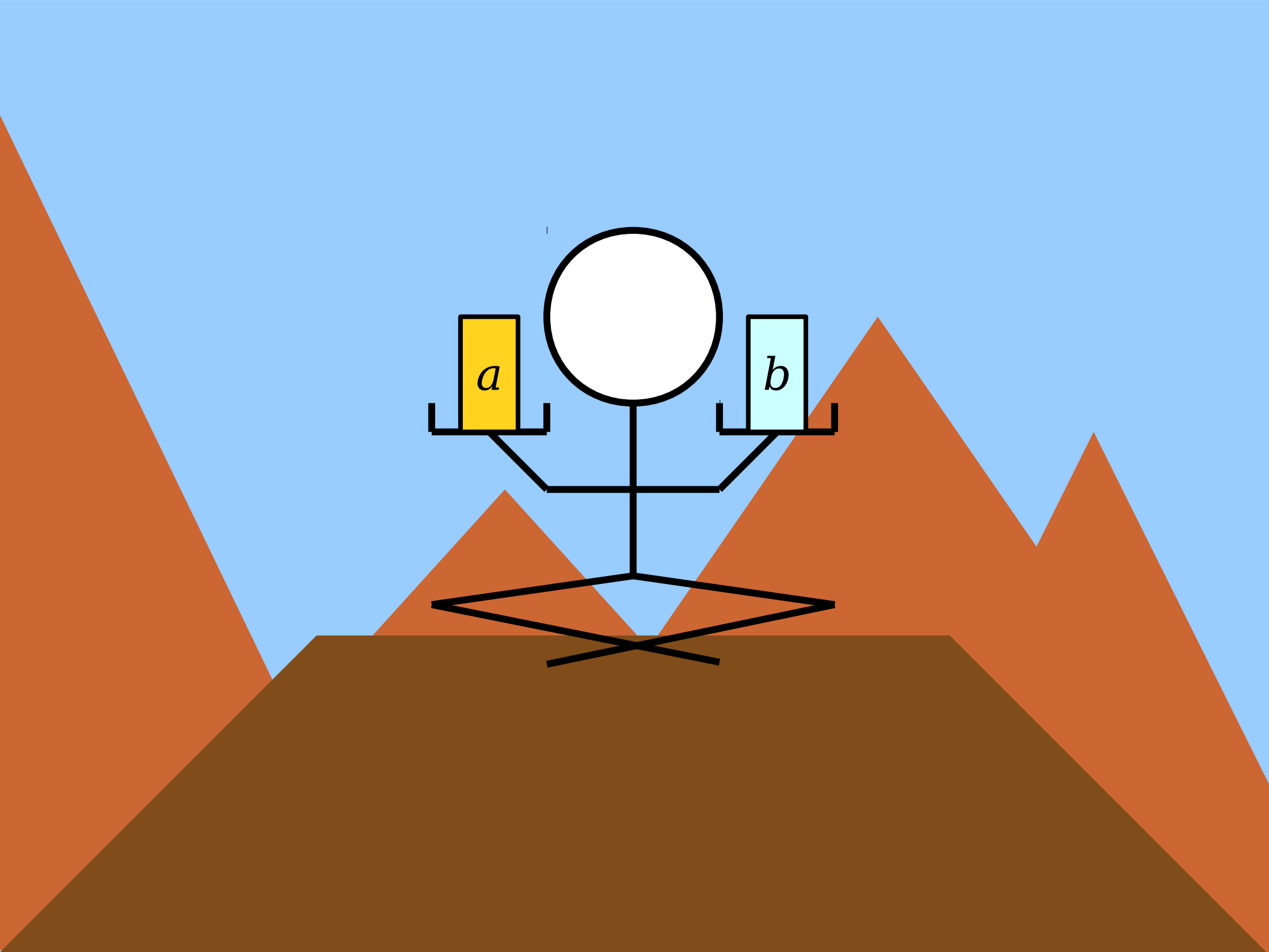






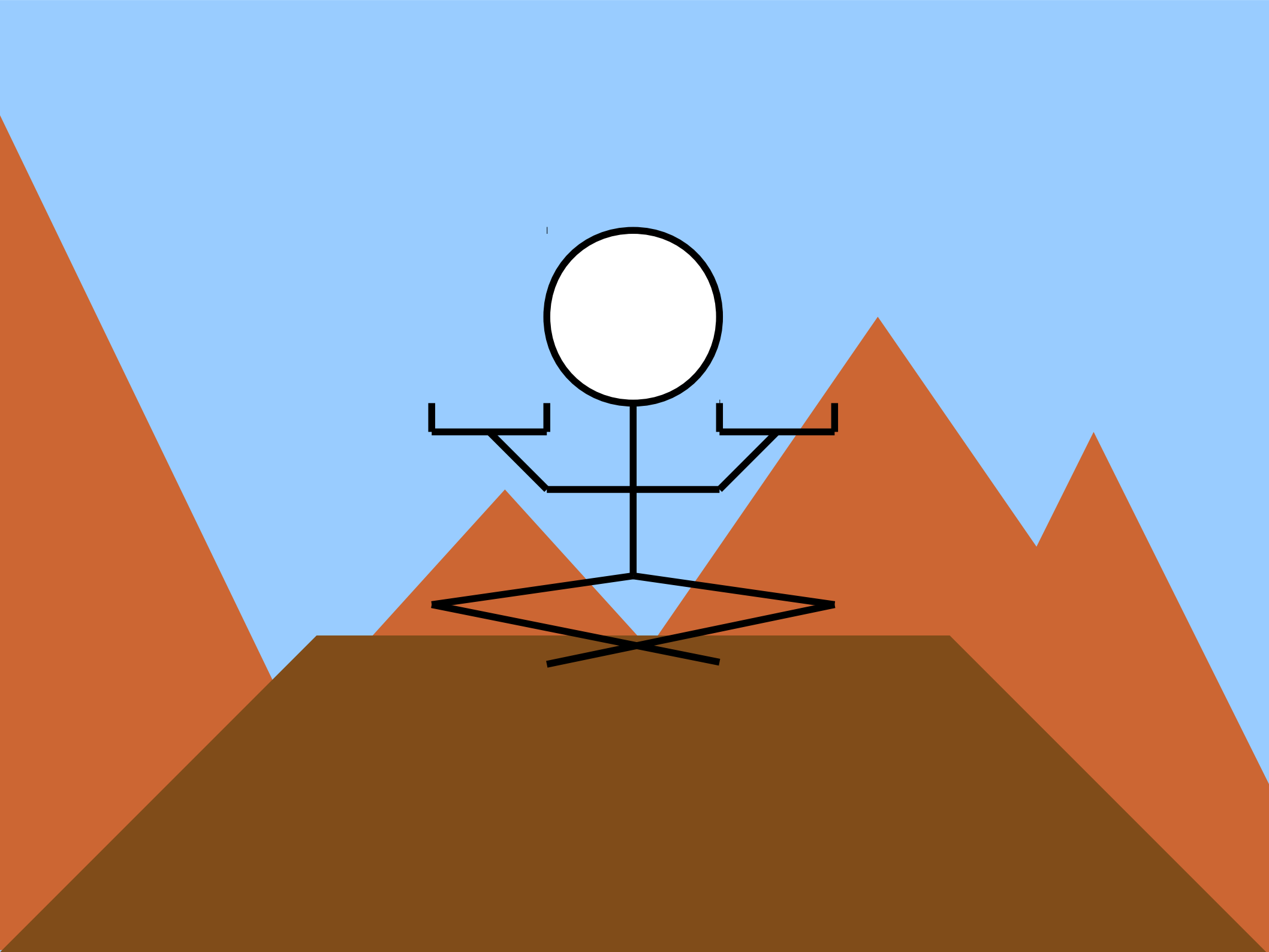


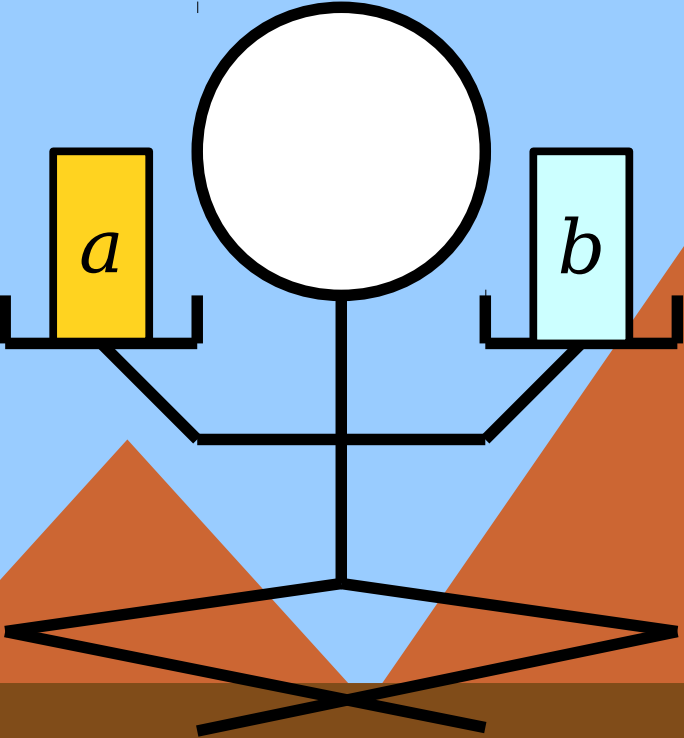
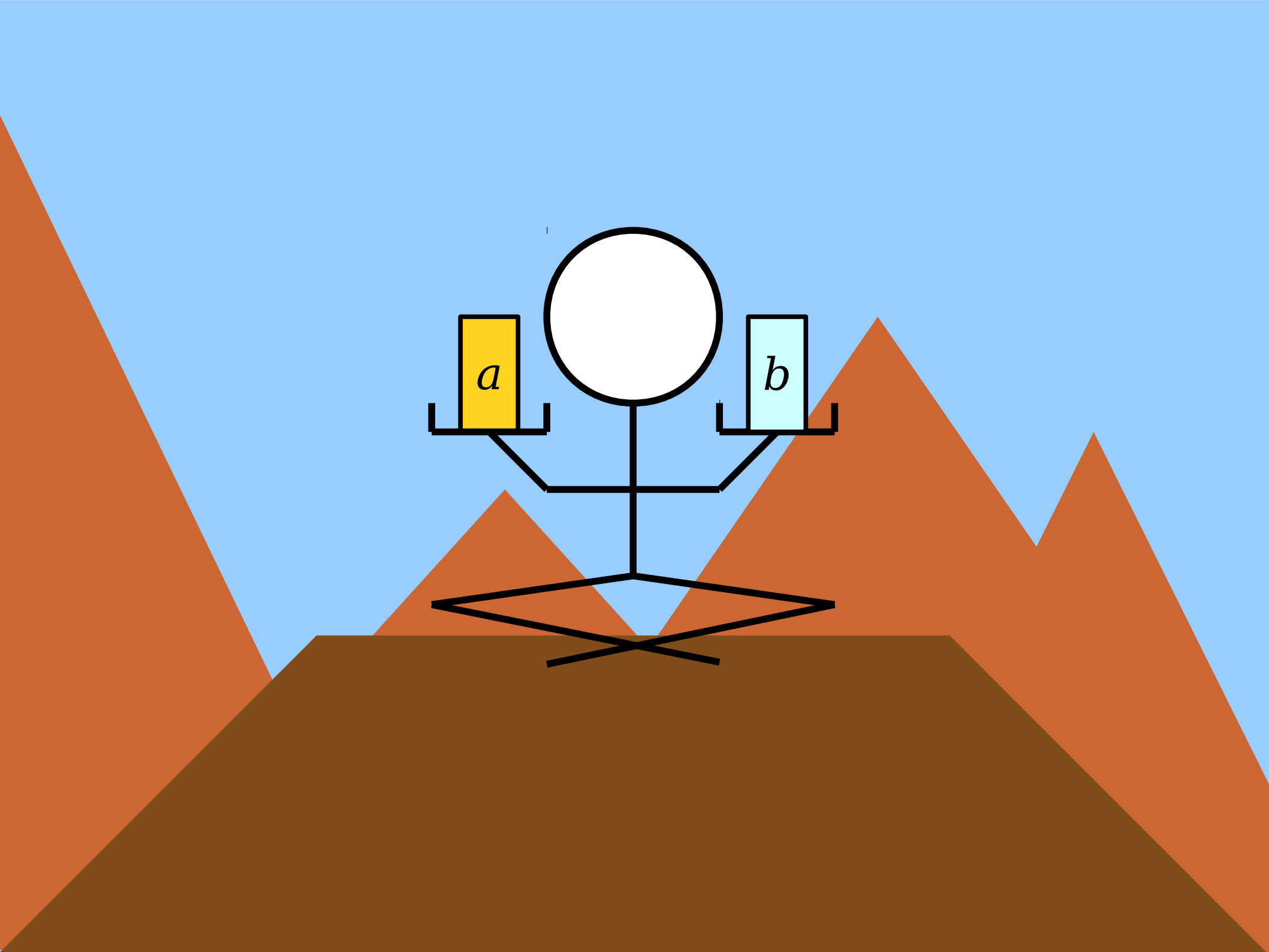


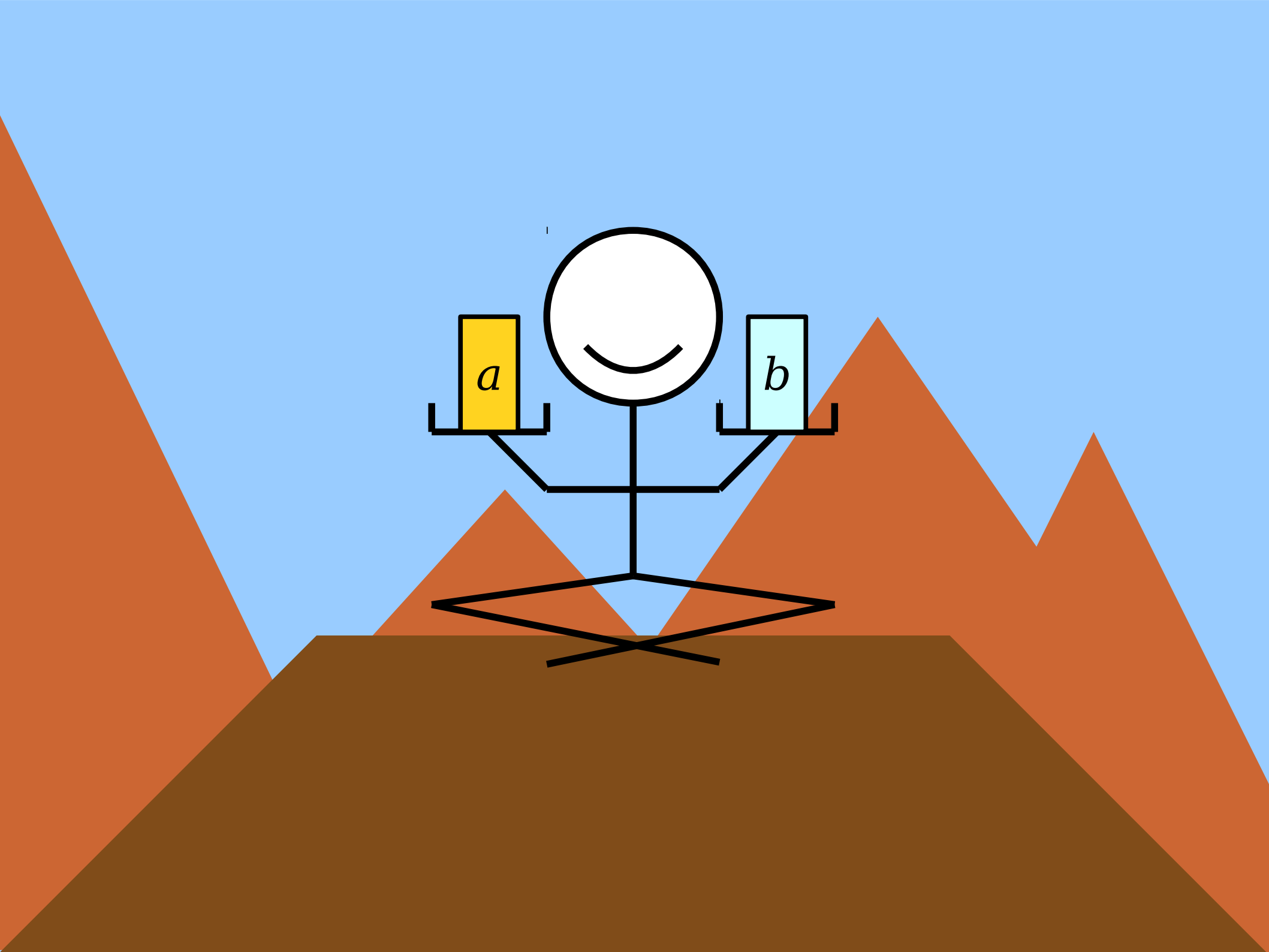






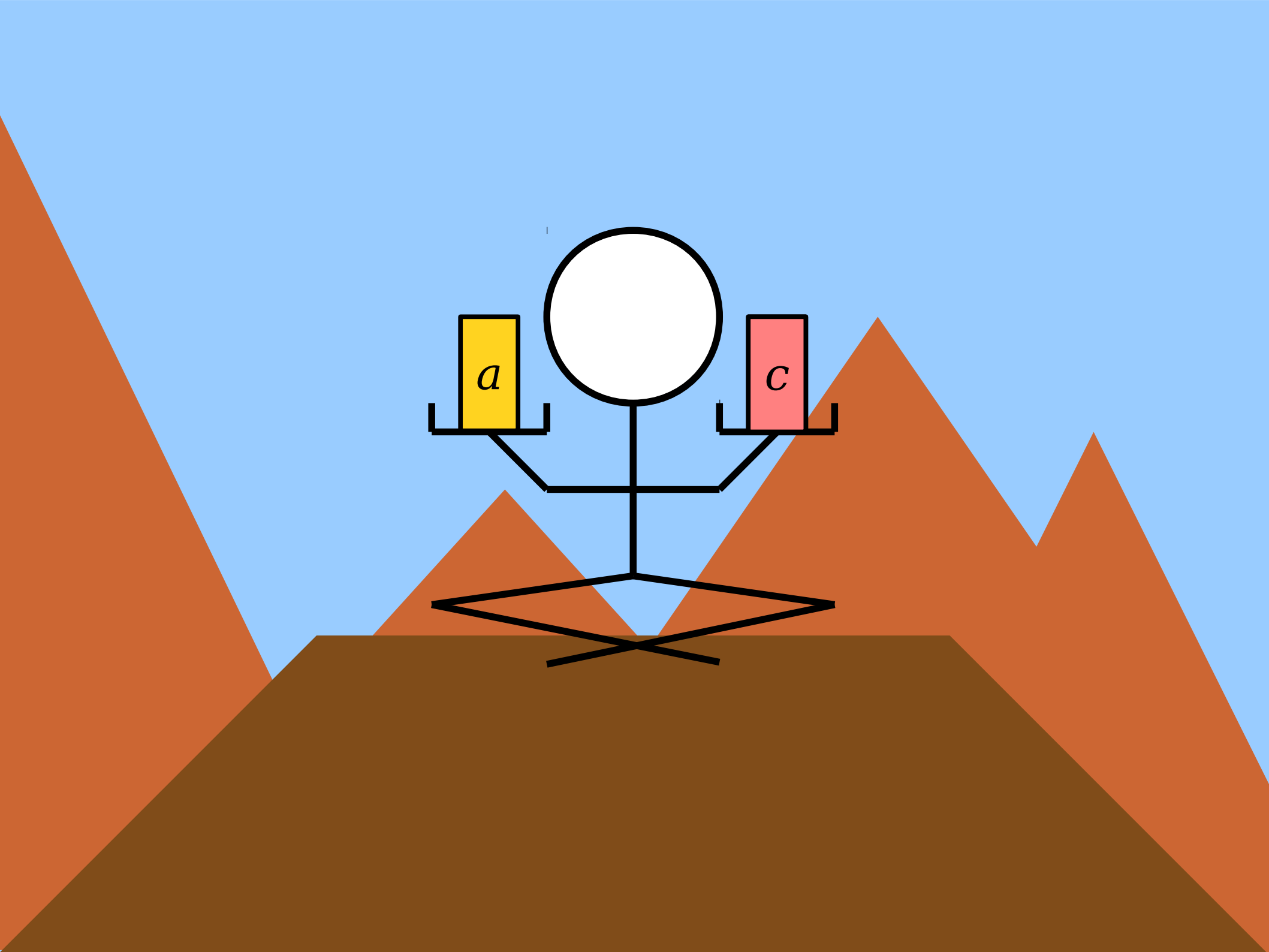


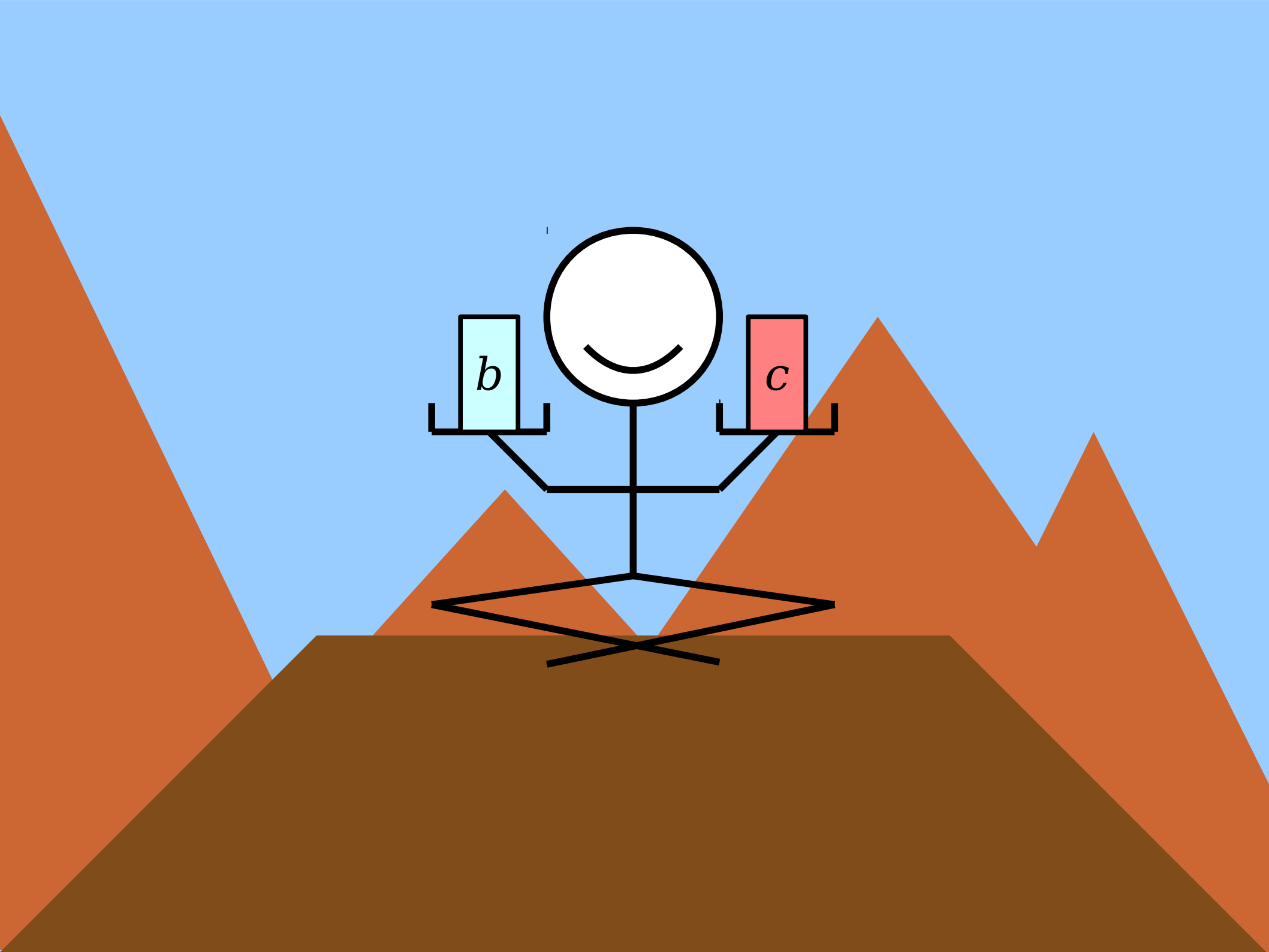


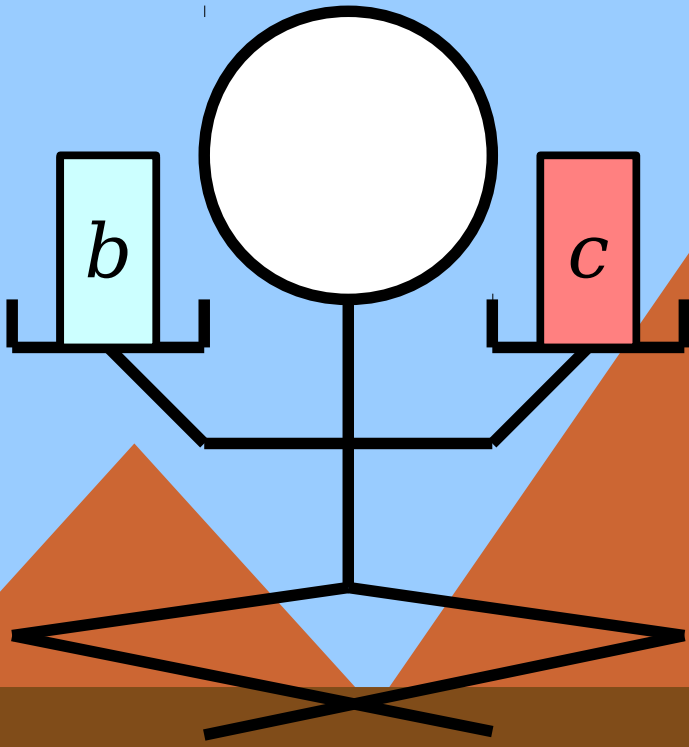
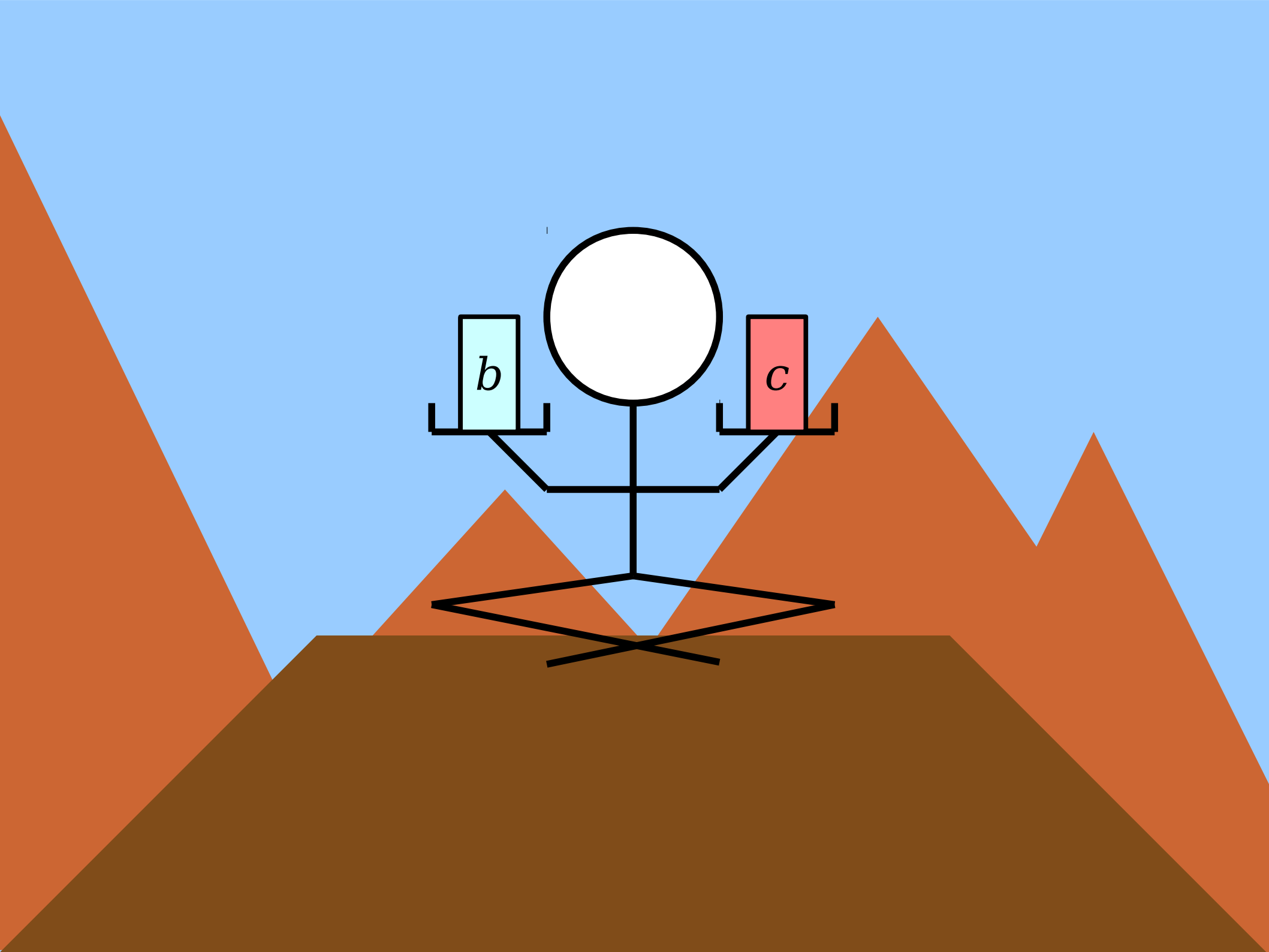


*a*

*b*





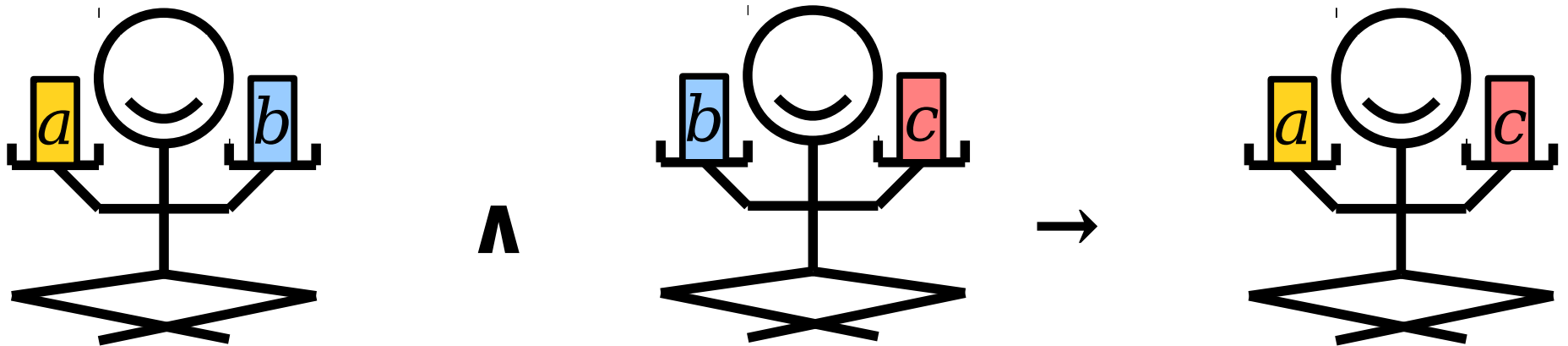
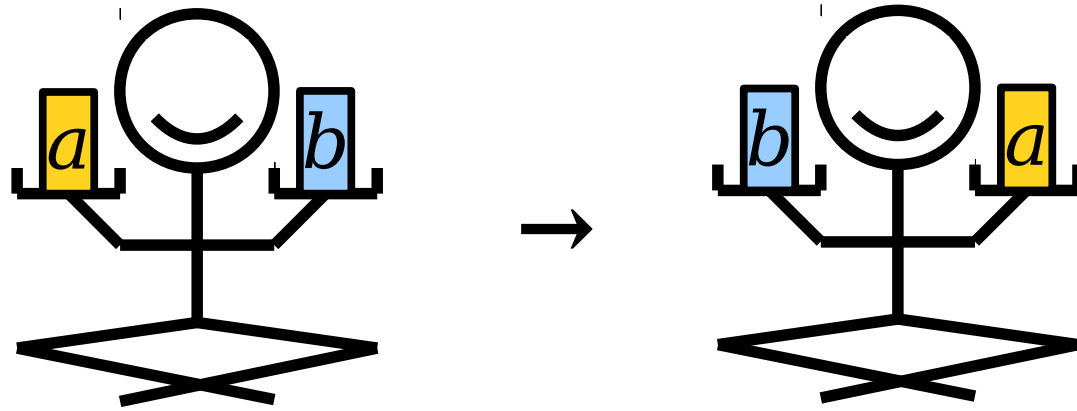
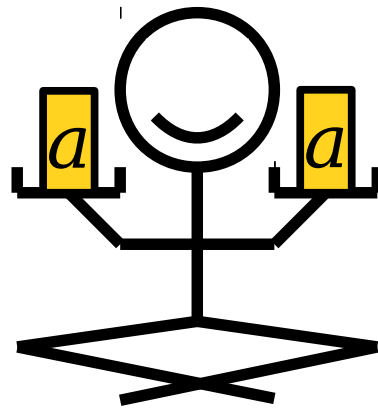


*b*

*c*







$aRa$

---

$aRb \rightarrow bRa$

---

$aRb \wedge bRc \rightarrow aRc$

$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

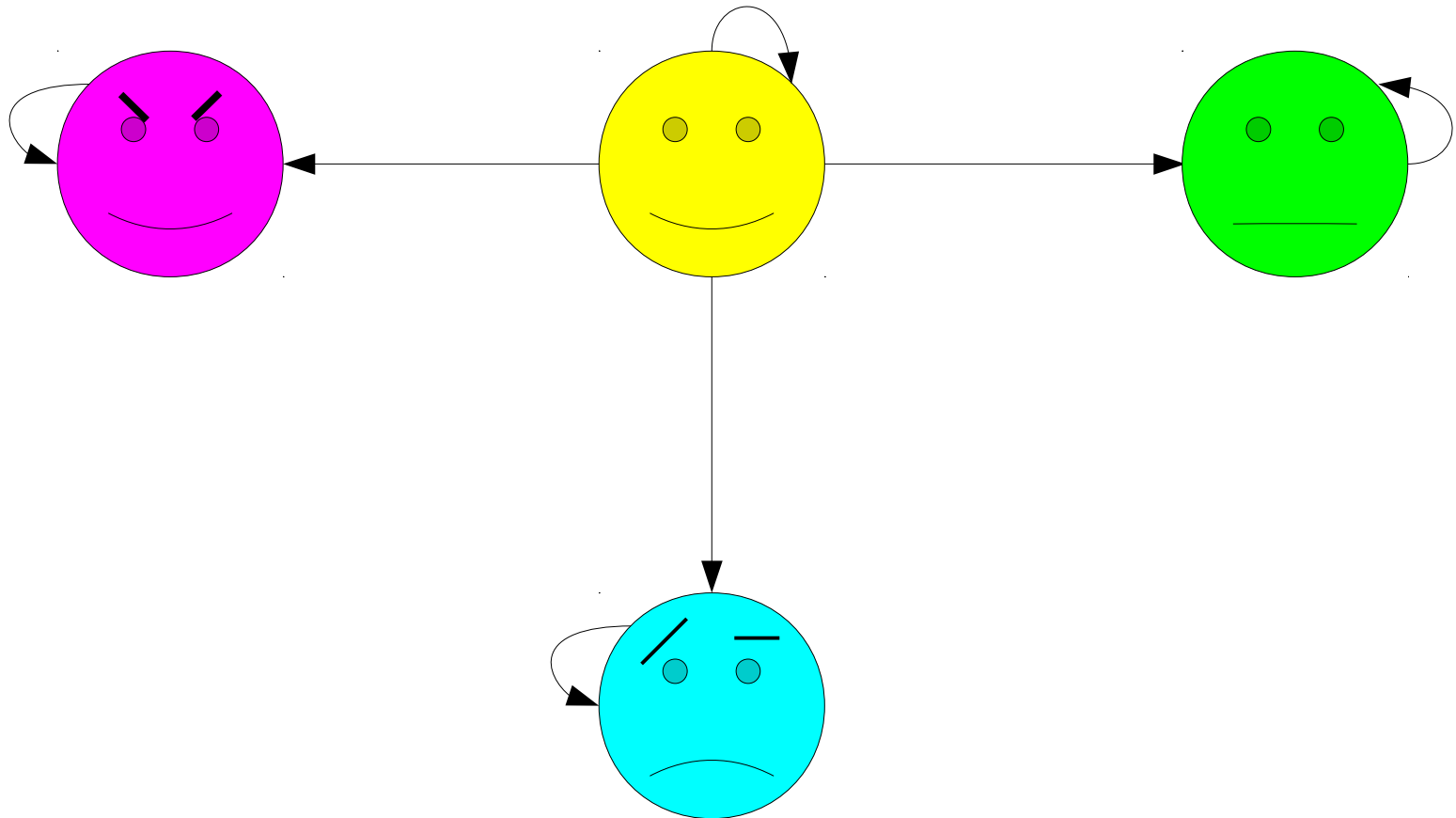
# Reflexivity

- Some relations always hold from any element to itself.
- Examples:
  - $x = x$  for any  $x$ .
  - $A \subseteq A$  for any set  $A$ .
  - $x \equiv_k x$  for any  $x$ .
- Relations of this sort are called ***reflexive***.
- Formally speaking, a binary relation  $R$  over a set  $A$  is reflexive if the following first-order statement is true:

$$\forall a \in A. aRa$$

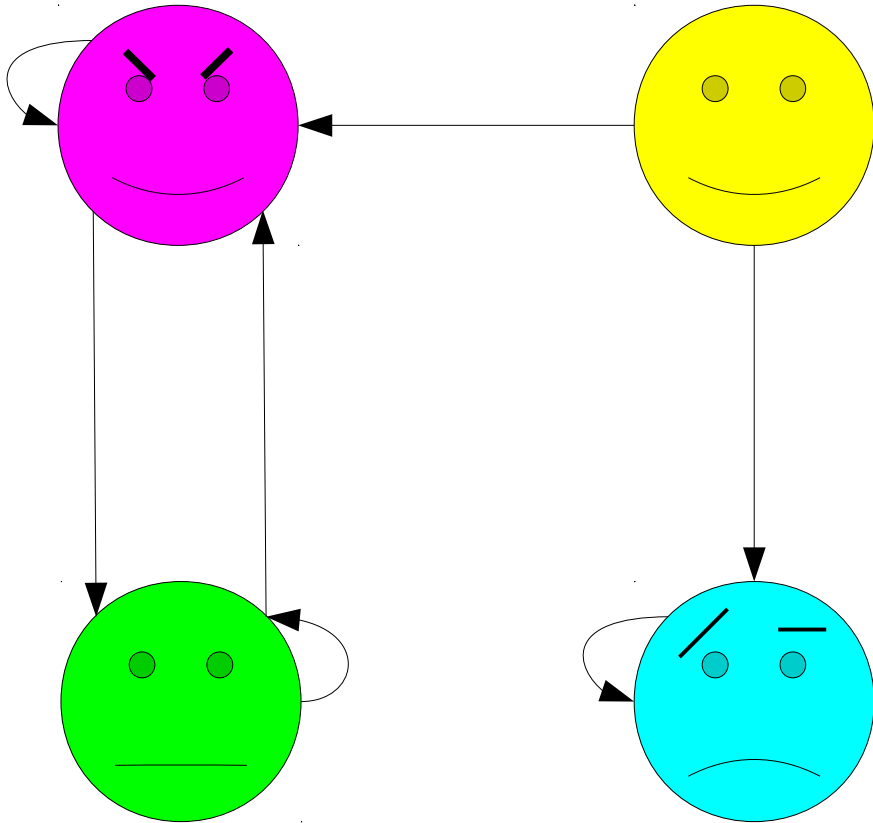
(“*Every element is related to itself.*”)

# Reflexivity Visualized



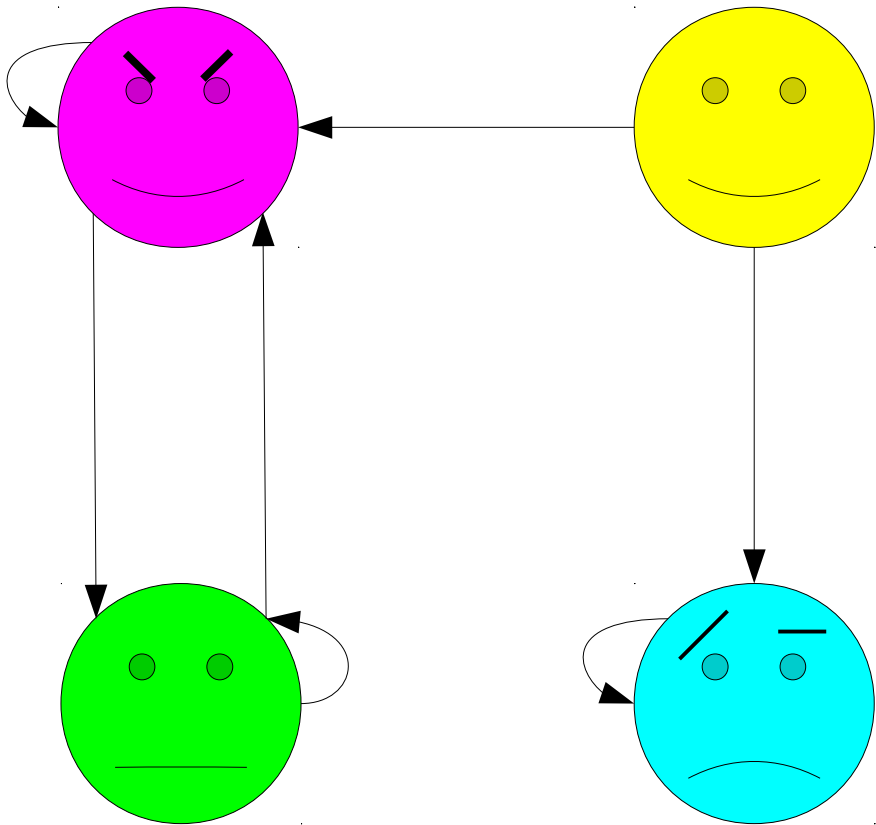
**$\forall a \in A. aRa$**

*(“Every element is related to itself.”)*



Let  $R$  be the relation drawn to the left. Is  $R$  reflexive?

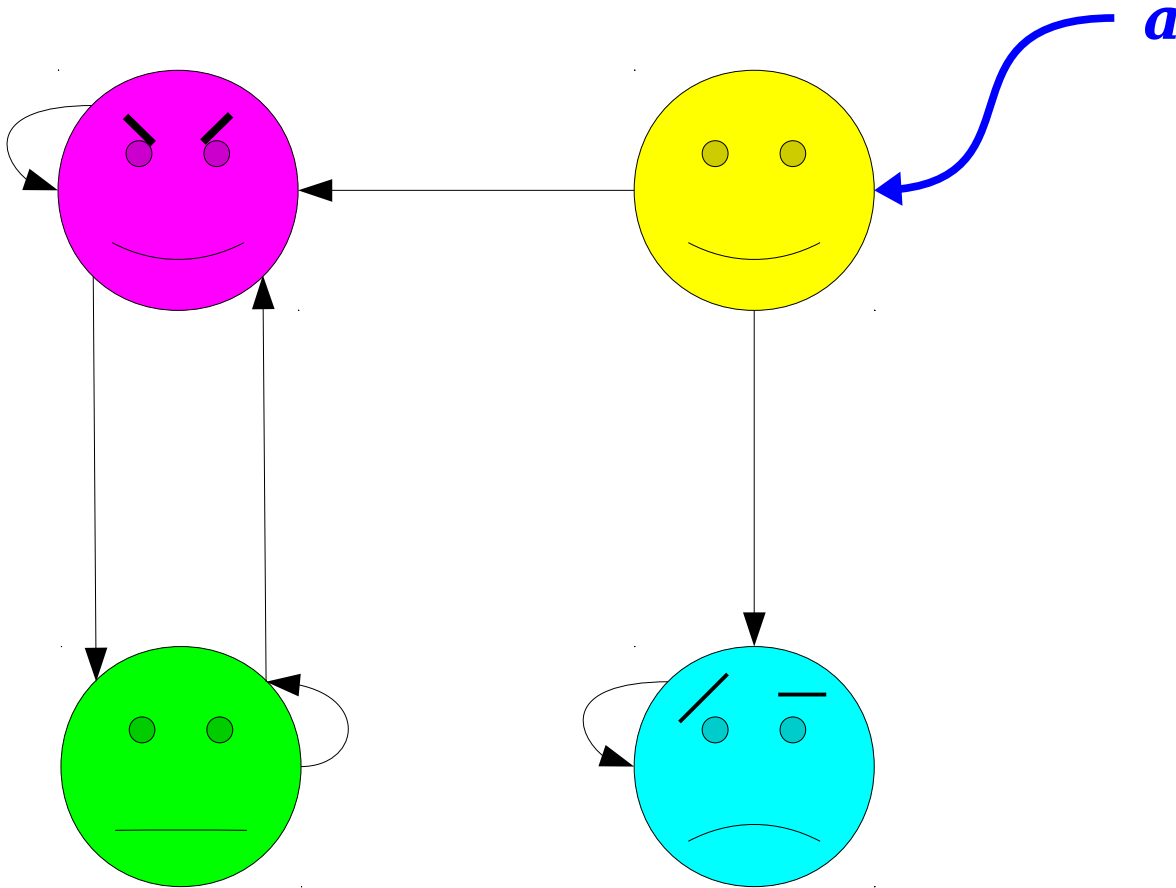
**$\forall a \in A. aRa$**   
(“Every element is related to itself.”)



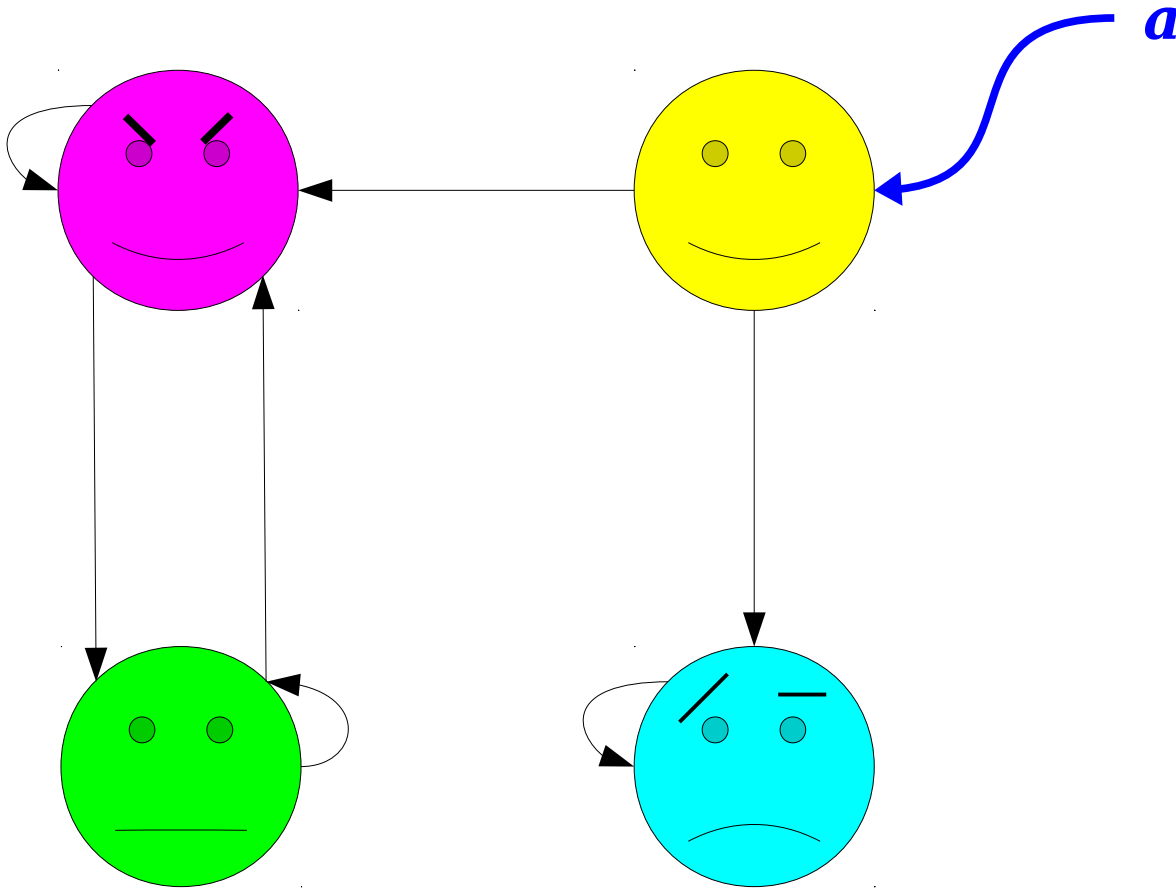
**$\forall a \in A. aRa$**

*(“Every element is related to itself.”)*



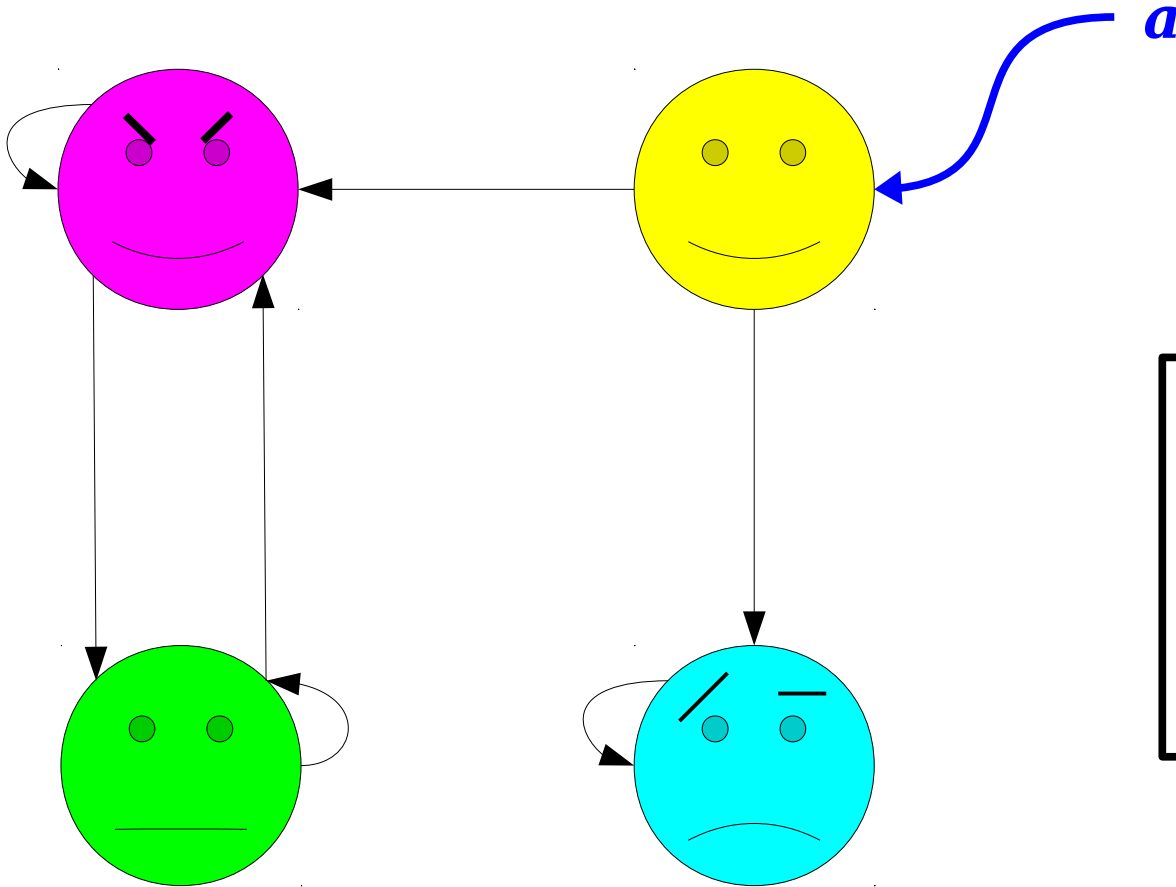


**$\forall a \in A. aRa$**   
(“Every element is related to itself.”)



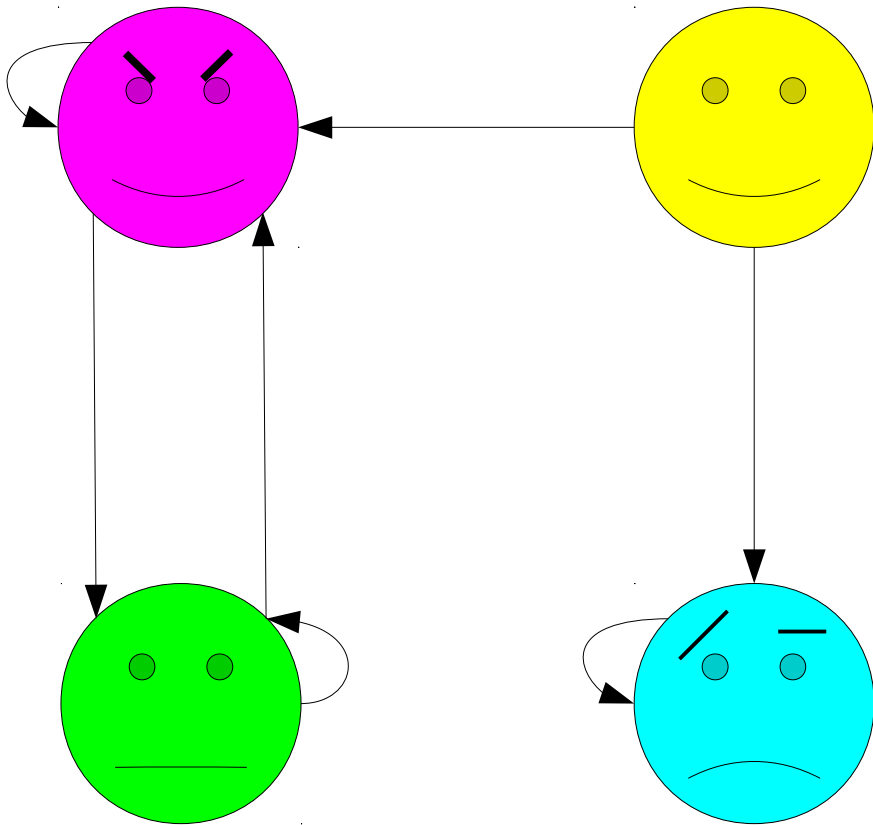
$$\forall a \in A. \mathbf{aRa}$$


*(“Every element is related to itself.”)*



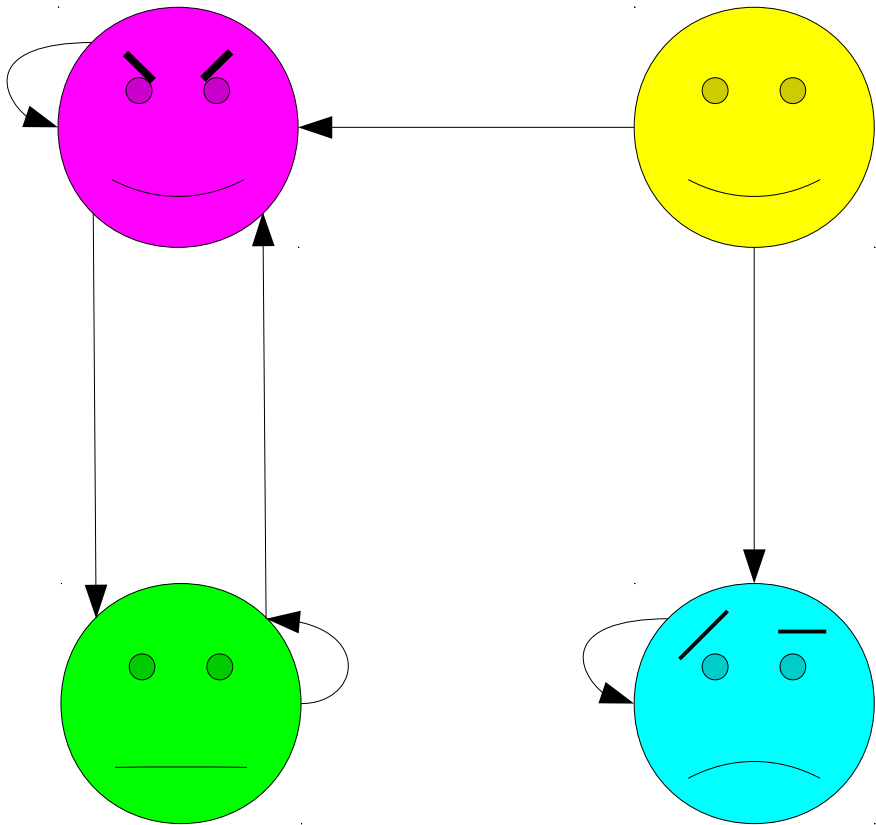
This means that  $R$  is not reflexive, since the first-order logic statement given below is not true.


**$\forall a \in A. aRa$**   
("Every element is related to itself.")



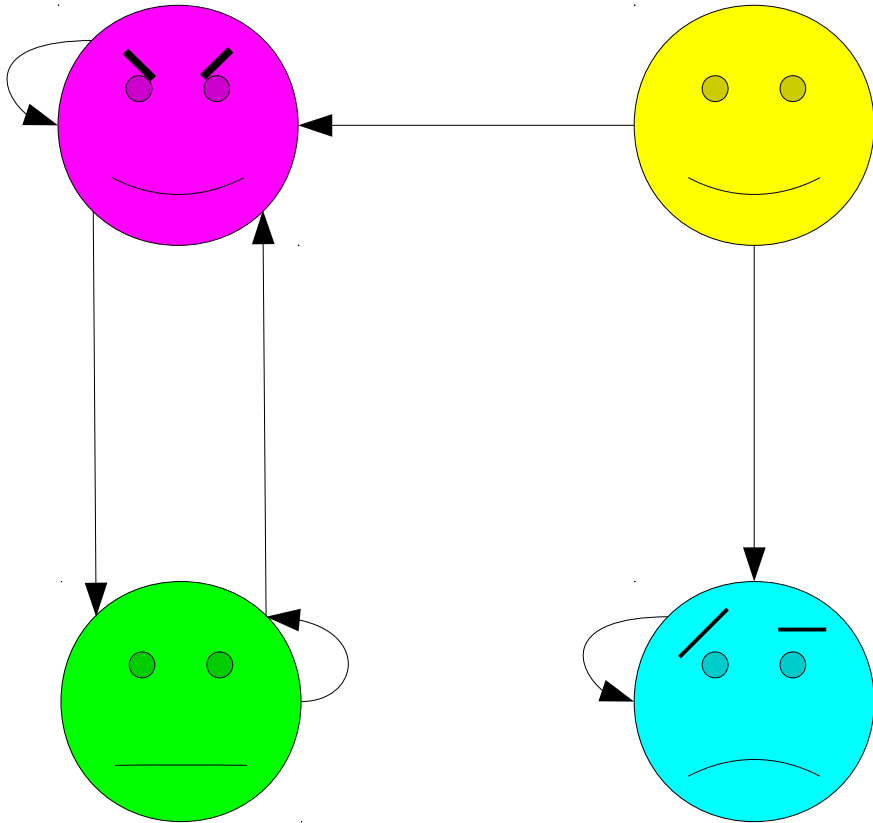
Is  reflexive?

**$\forall a \in A. aRa$**   
*(“Every element is related to itself.”)*



Is  reflexive?

$\forall a \in ?? . a$    $a$



Is  reflexive?

Reflexivity is a property of *relations*, not *individual objects*.

$\forall a \in ?? . a$    $a$

$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$



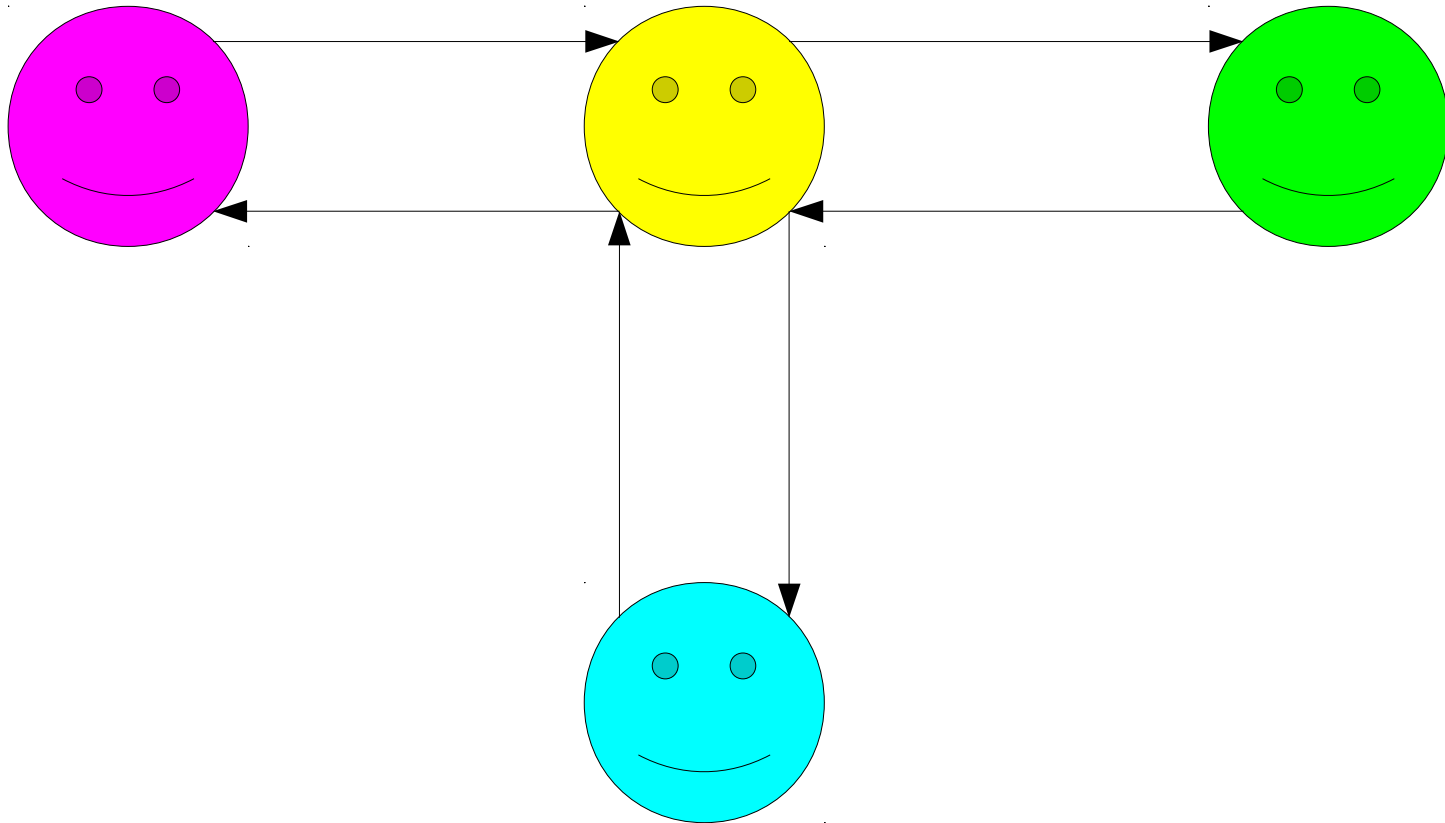
# Symmetry

- In some relations, the relative order of the objects doesn't matter.
- Examples:
  - If  $x = y$ , then  $y = x$ .
  - If  $x \equiv_k y$ , then  $y \equiv_k x$ .
- These relations are called ***symmetric***.
- Formally: a binary relation  $R$  over a set  $A$  is called *symmetric* if the following first-order statement is true about  $R$ :

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)

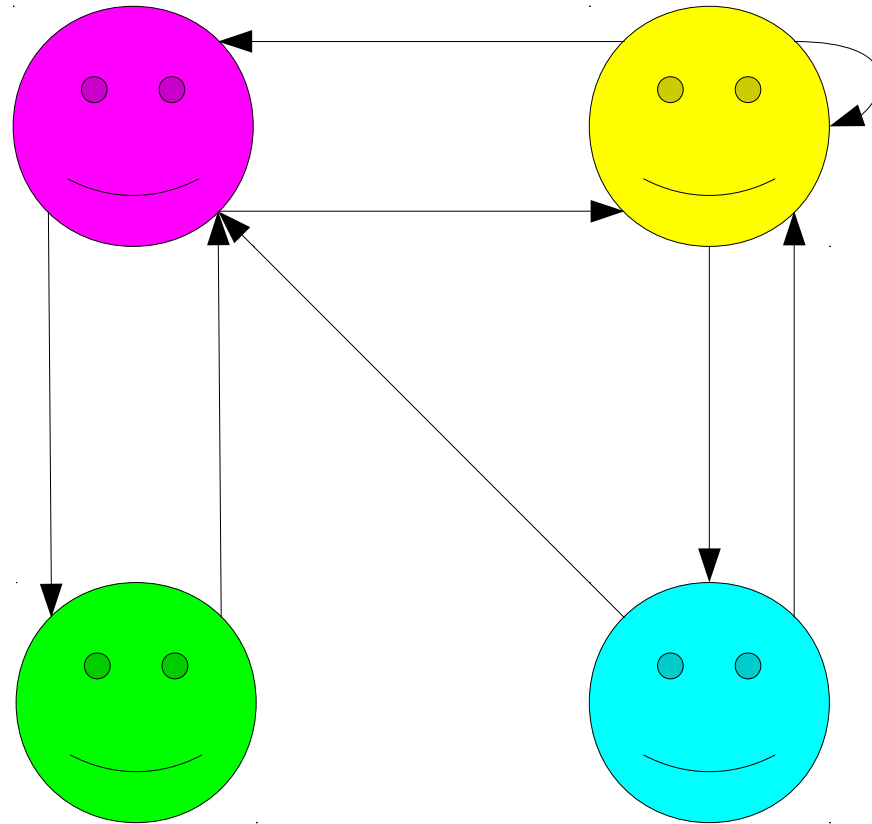
# Symmetry Visualized



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

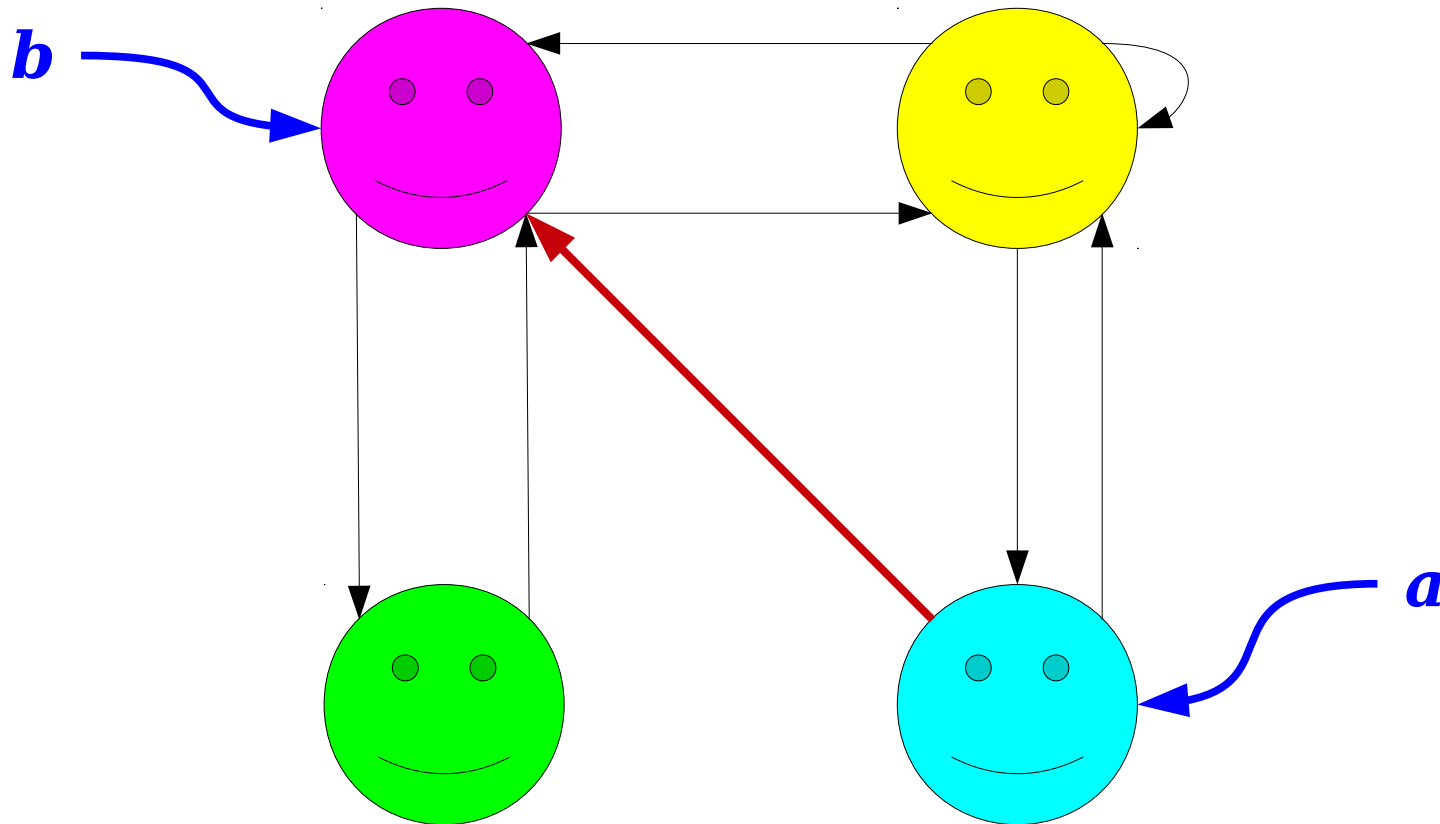
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

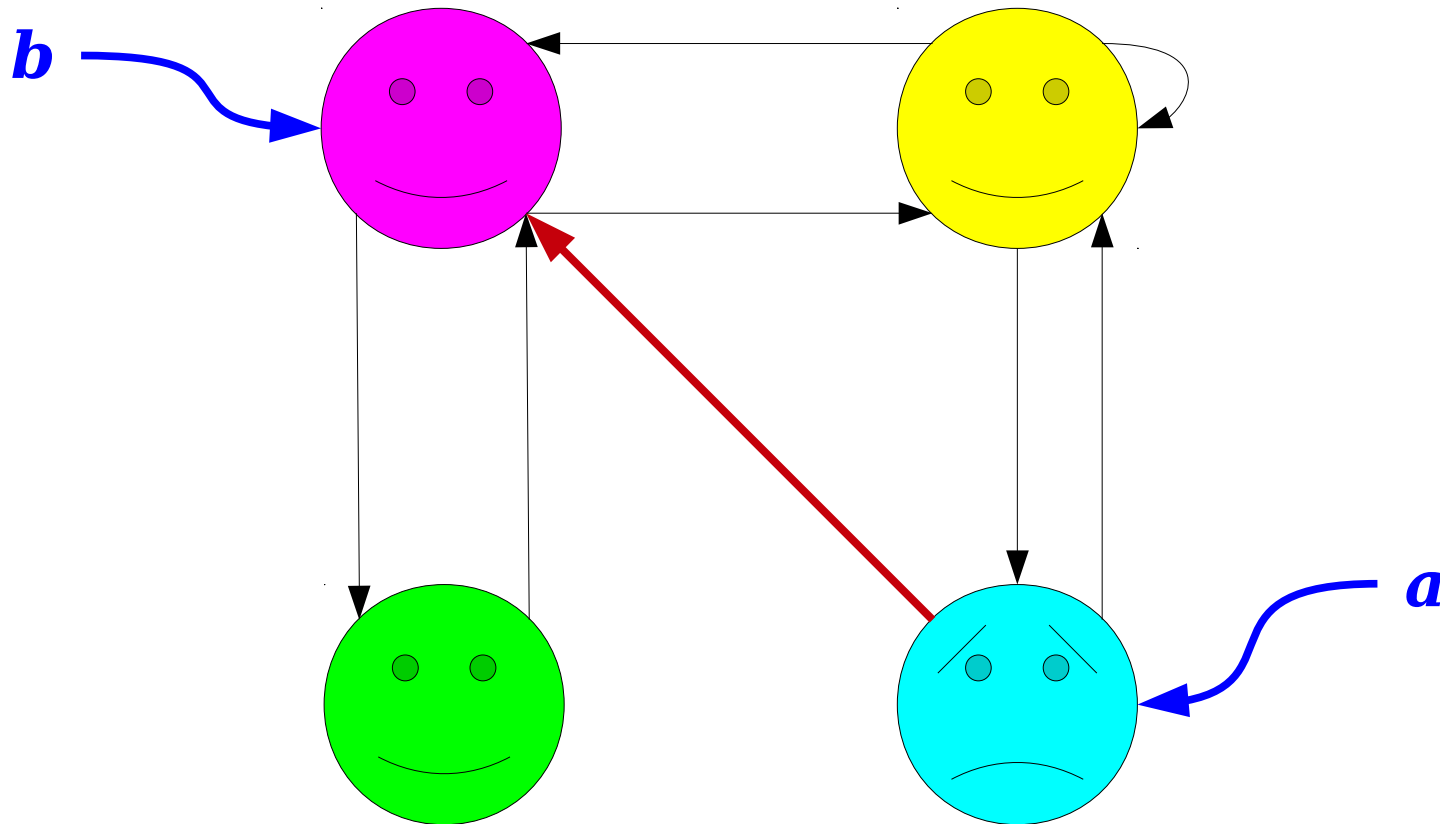
# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

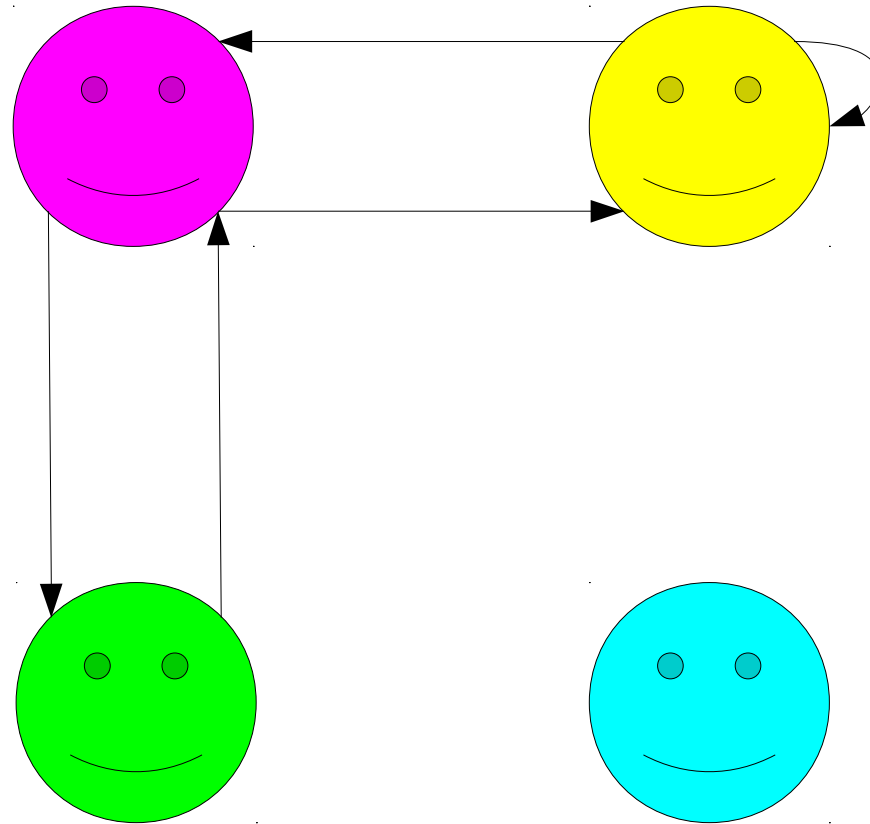
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

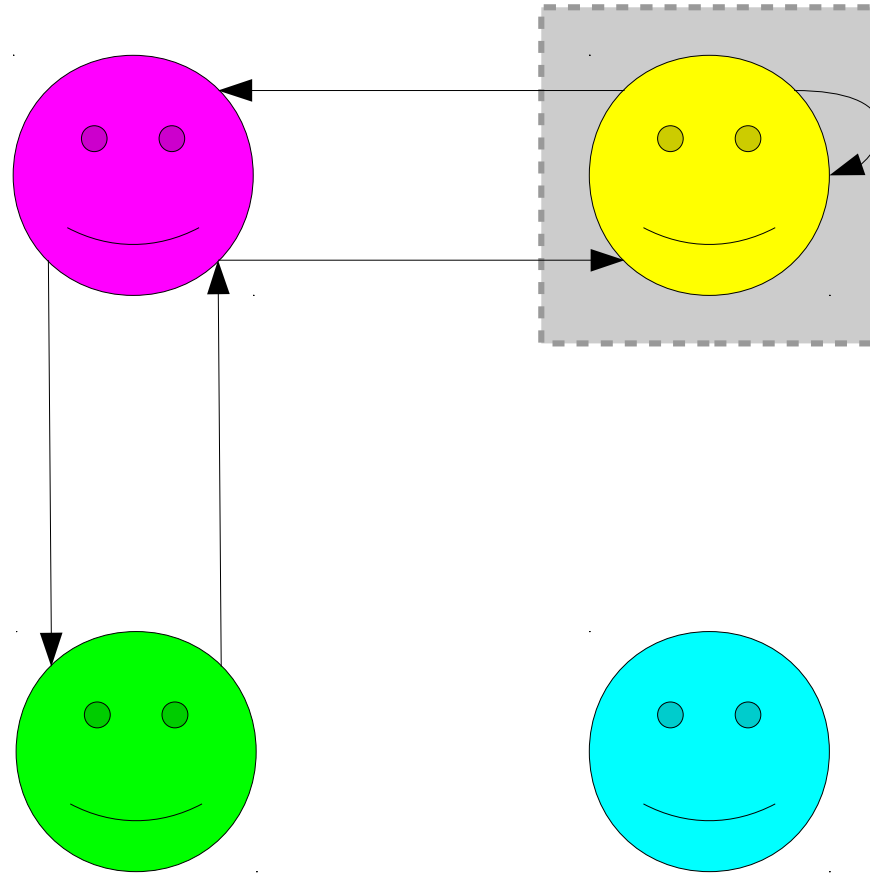
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

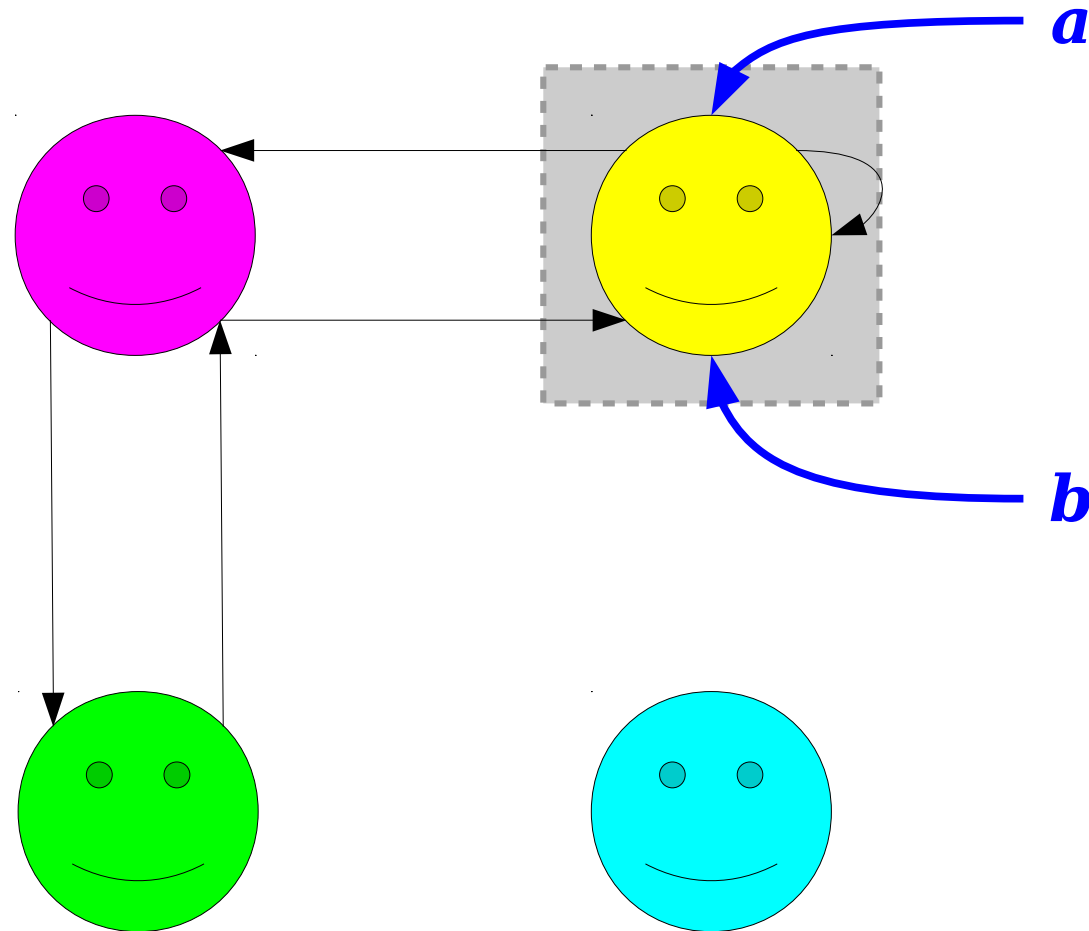
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If a is related to b, then b is related to a.”)*

# Is This Relation Symmetric?

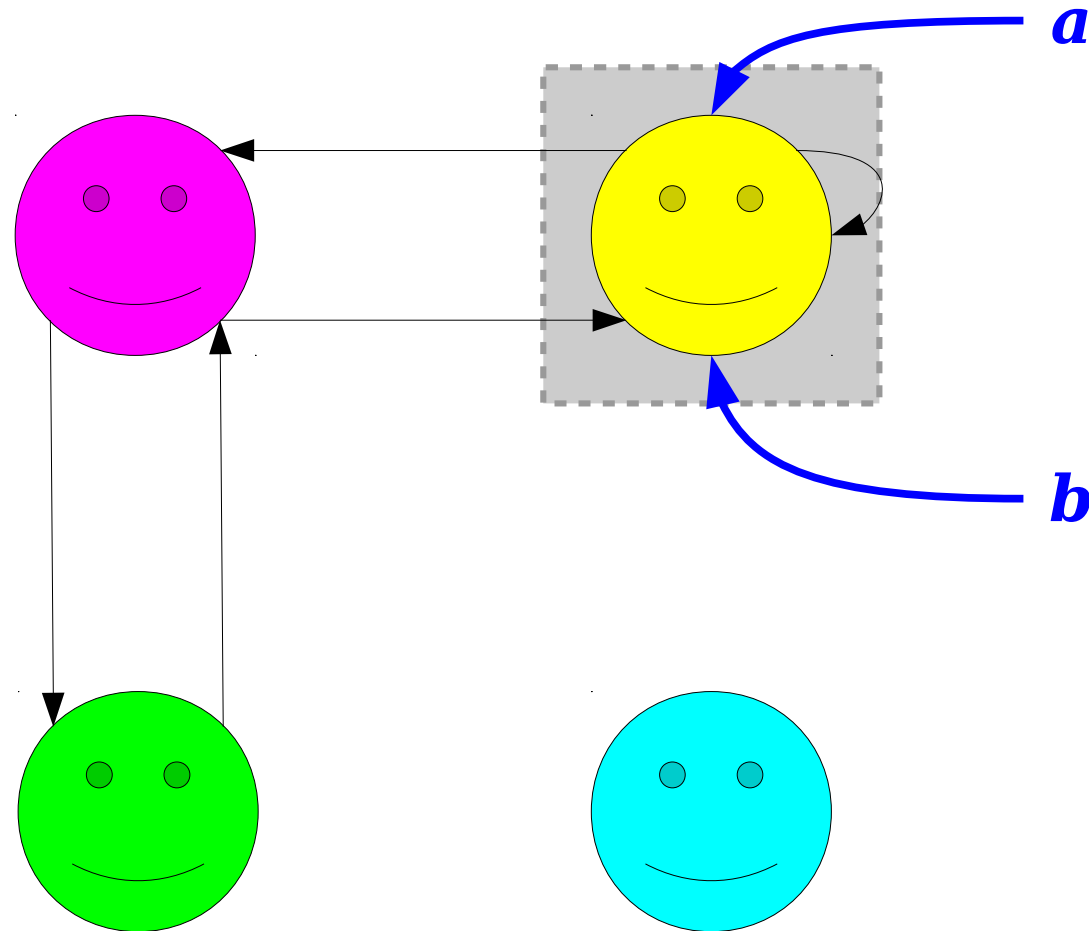


**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If a is related to b, then b is related to a.”)*



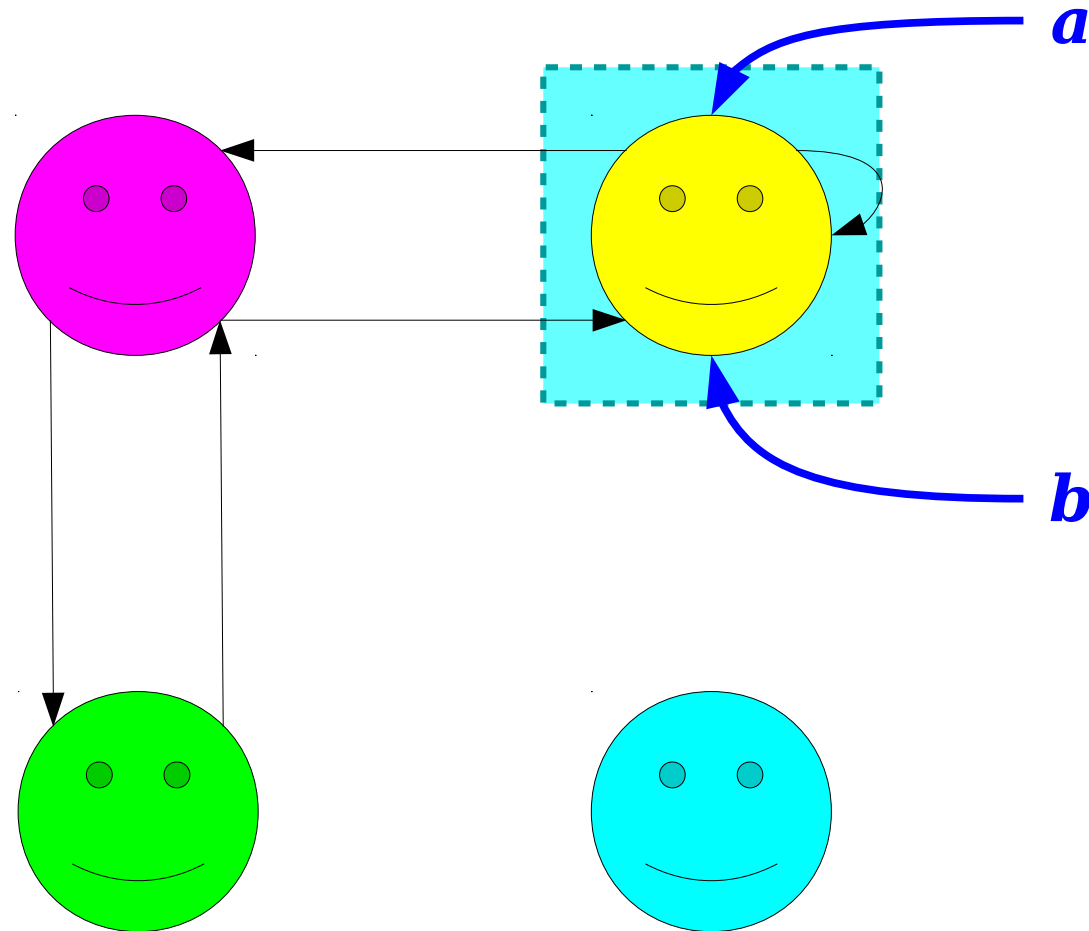
# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

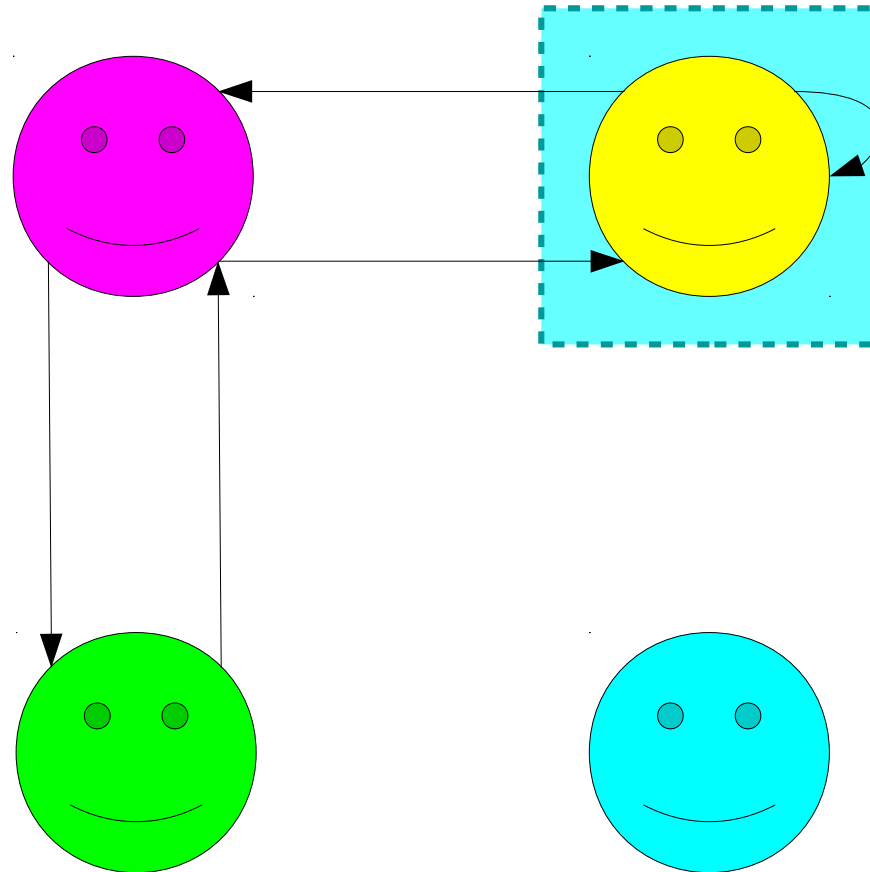
# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

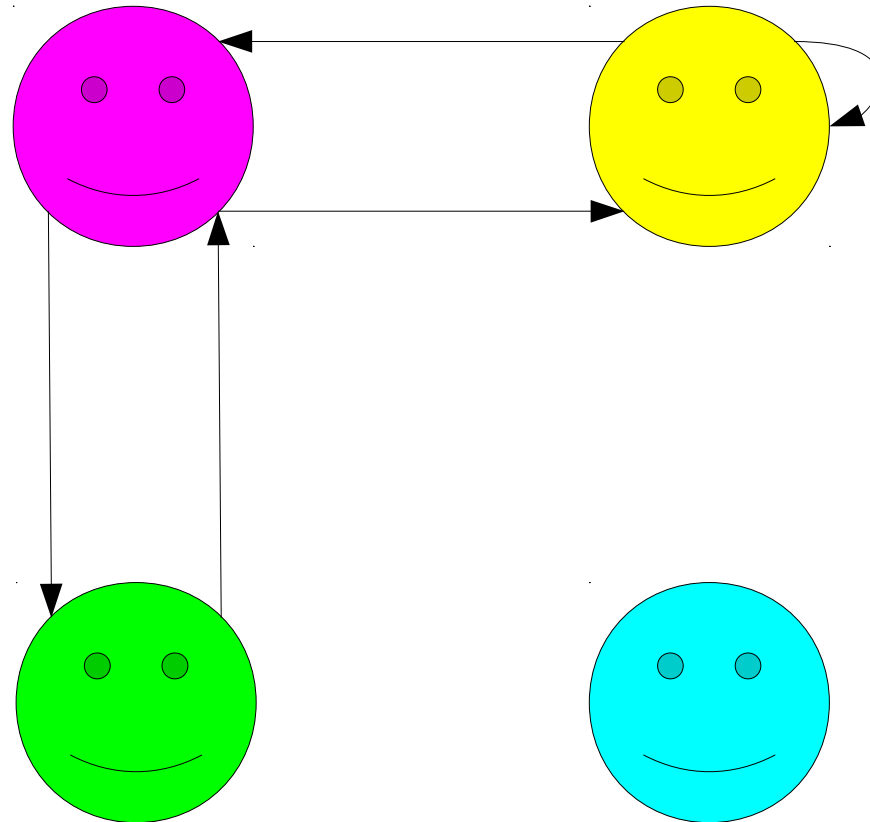
# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If a is related to b, then b is related to a.”)*

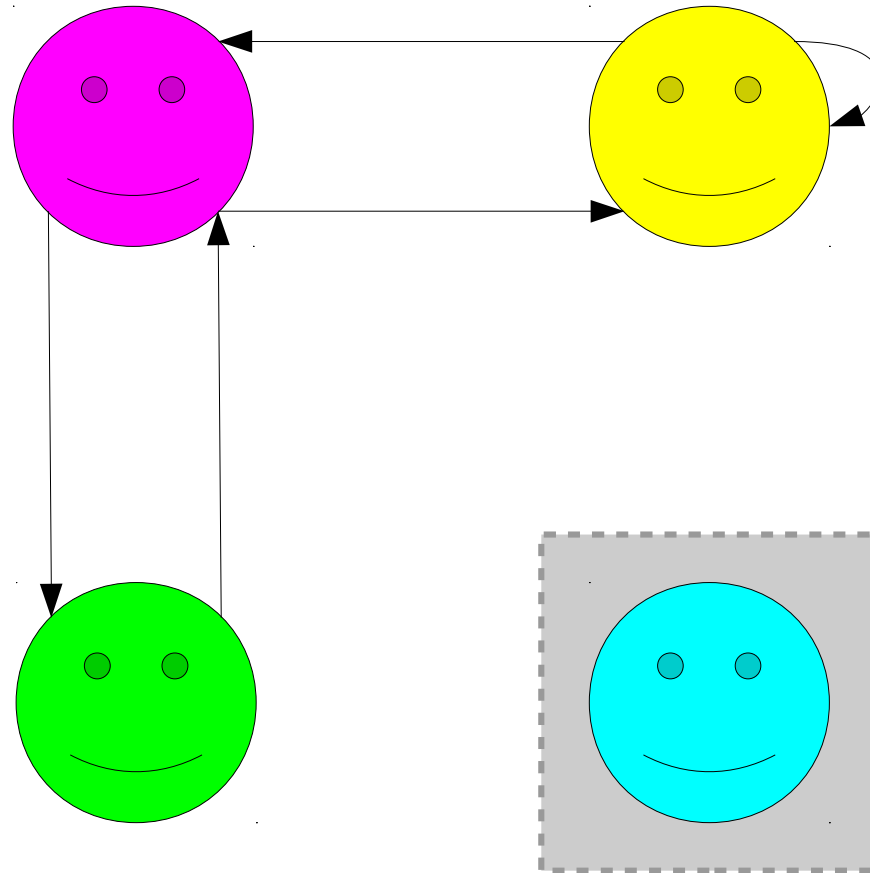
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

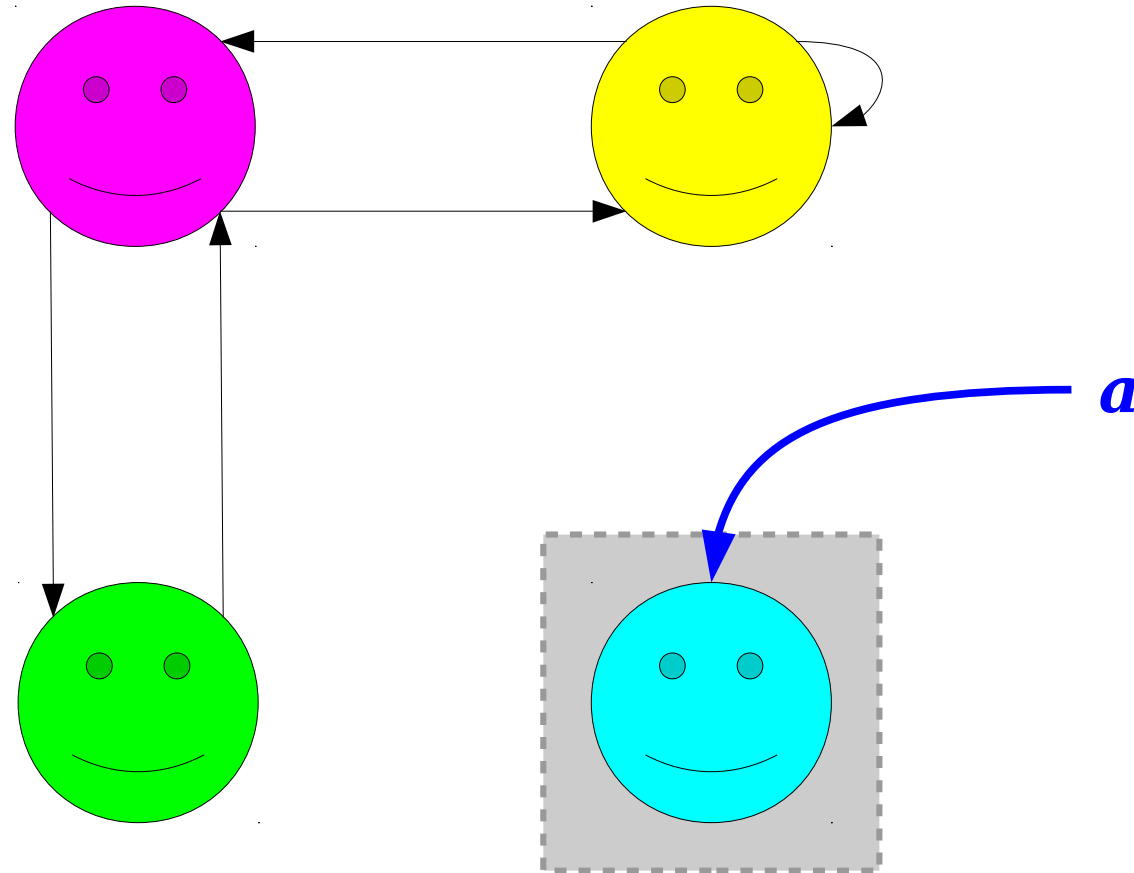
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

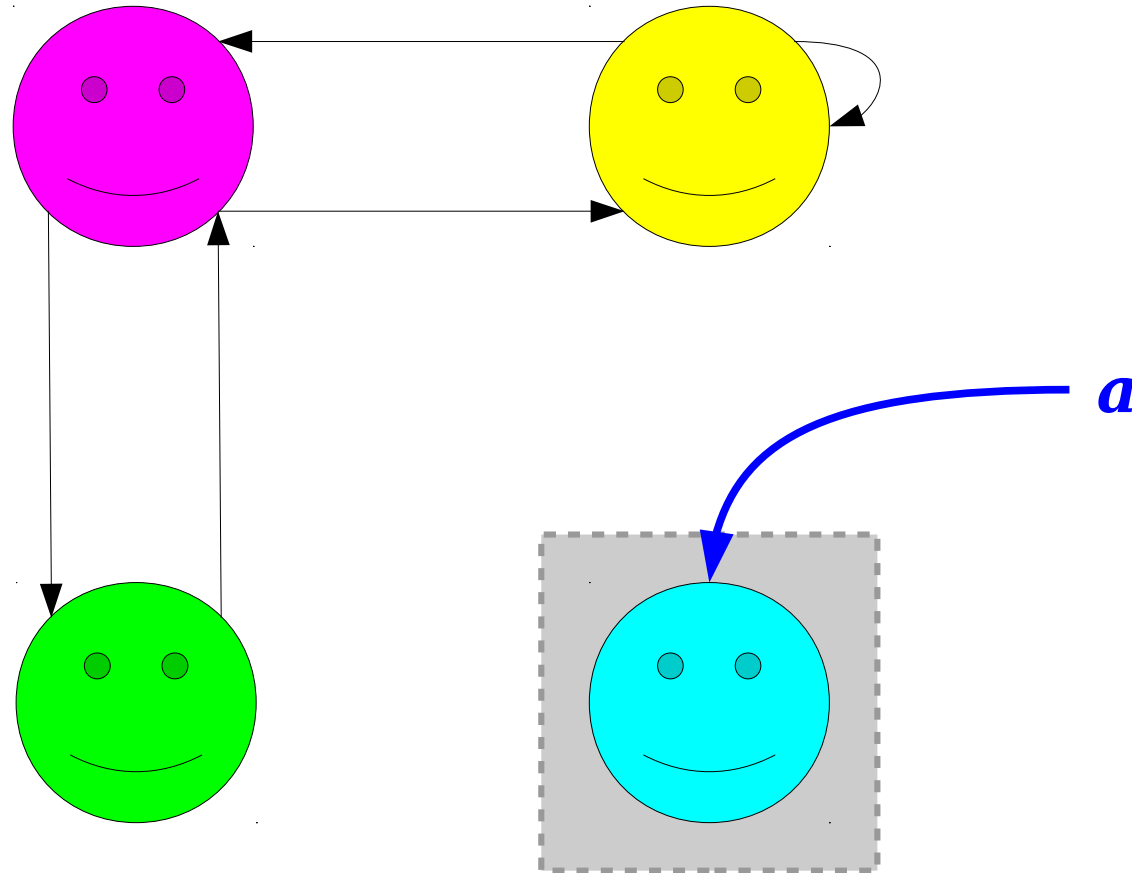
# Is This Relation Symmetric?



**$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$**

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

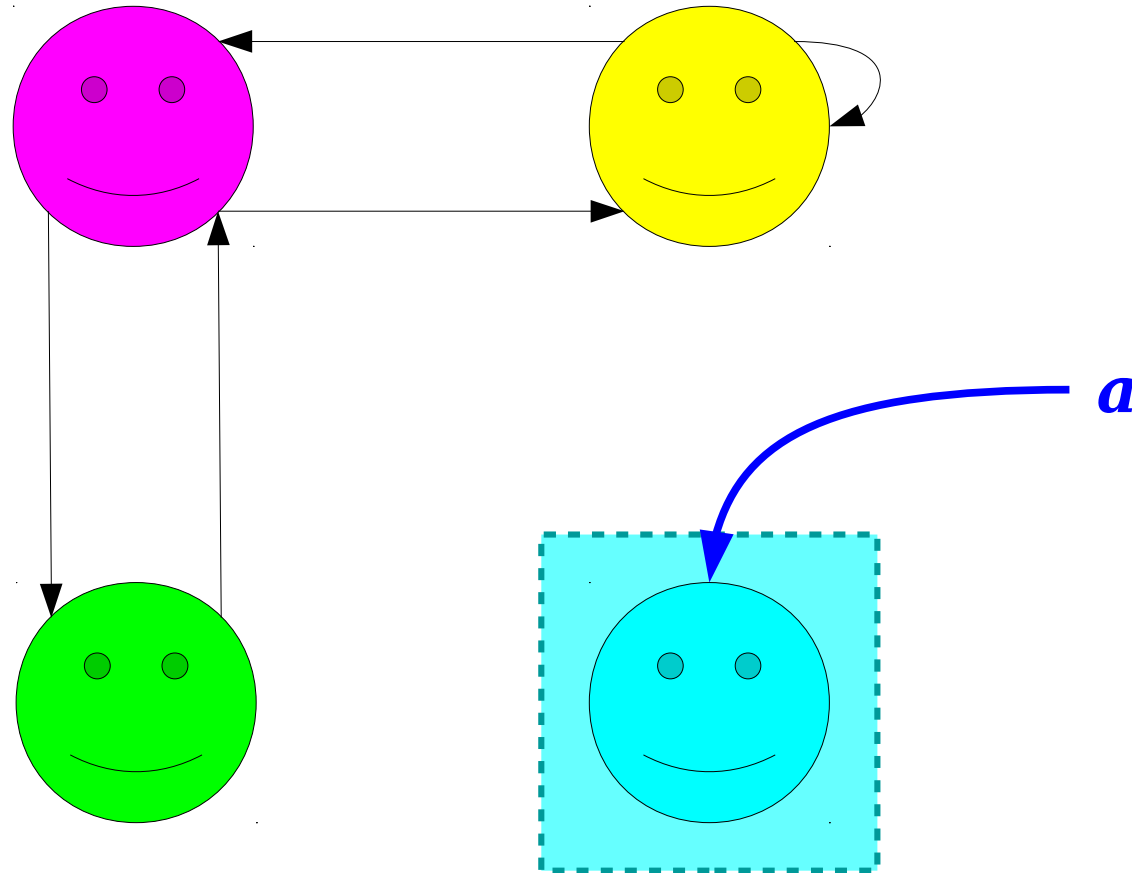
# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*

# Is This Relation Symmetric?



$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$

*(“If  $a$  is related to  $b$ , then  $b$  is related to  $a$ .”)*



$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

$$\forall a \in A. aRa$$

---

$$\forall a \in A. \forall b \in A. (aRb \rightarrow bRa)$$

---

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

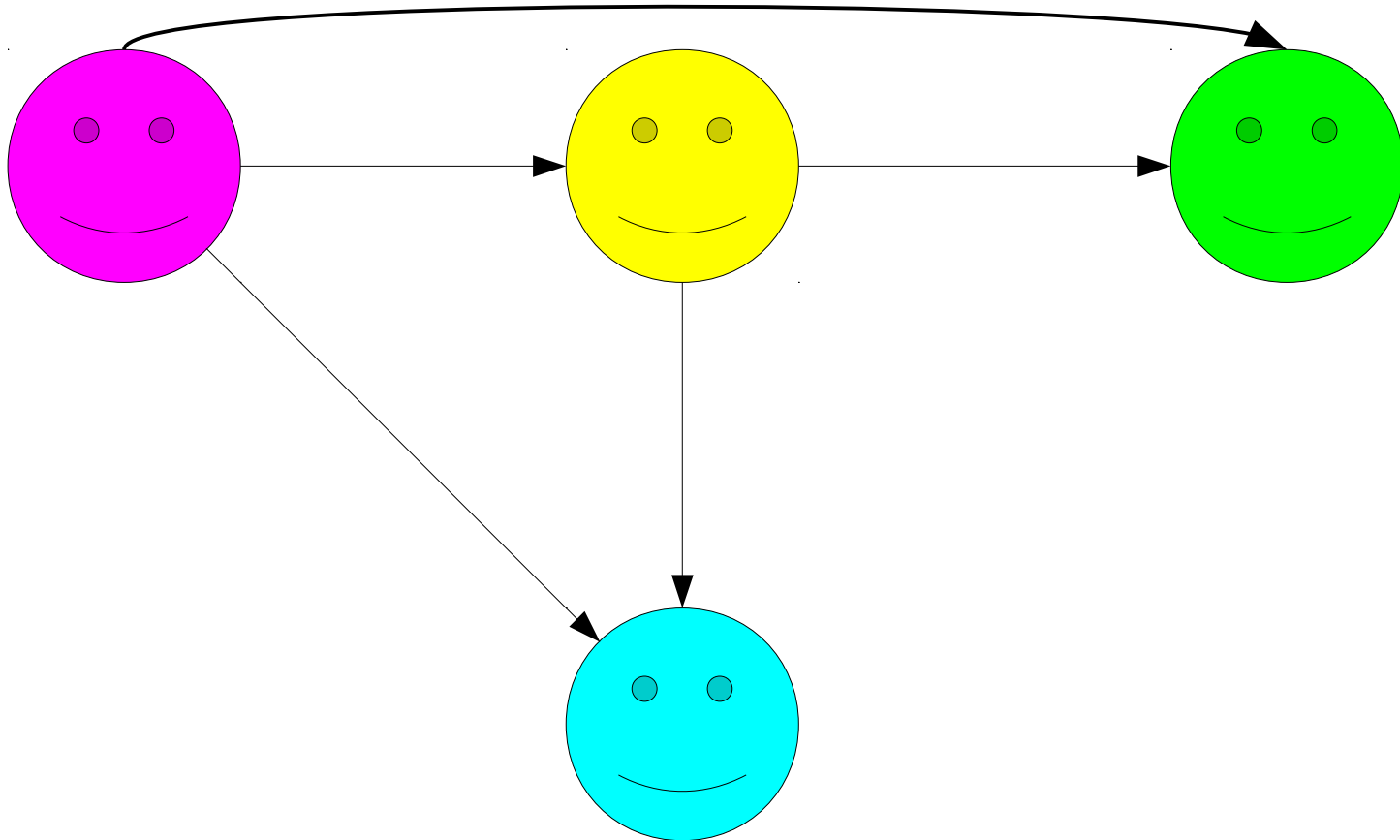
# Transitivity

- Many relations can be chained together.
- Examples:
  - If  $x = y$  and  $y = z$ , then  $x = z$ .
  - If  $R \subseteq S$  and  $S \subseteq T$ , then  $R \subseteq T$ .
  - If  $x \equiv_k y$  and  $y \equiv_k z$ , then  $x \equiv_k z$ .
- These relations are called ***transitive***.
- A binary relation  $R$  over a set  $A$  is called *transitive* if the following first-order statement is true about  $R$ :

$$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$$

*(“Whenever  $a$  is related to  $b$  and  $b$  is related to  $c$ , we know  $a$  is related to  $c$ .)*

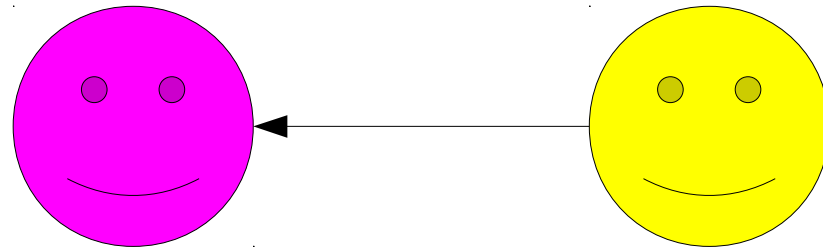
# Transitivity Visualized



**$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$**

*("Whenever  $a$  is related to  $b$  and  $b$  is related to  $c$ , we know  $a$  is related to  $c$ .)*

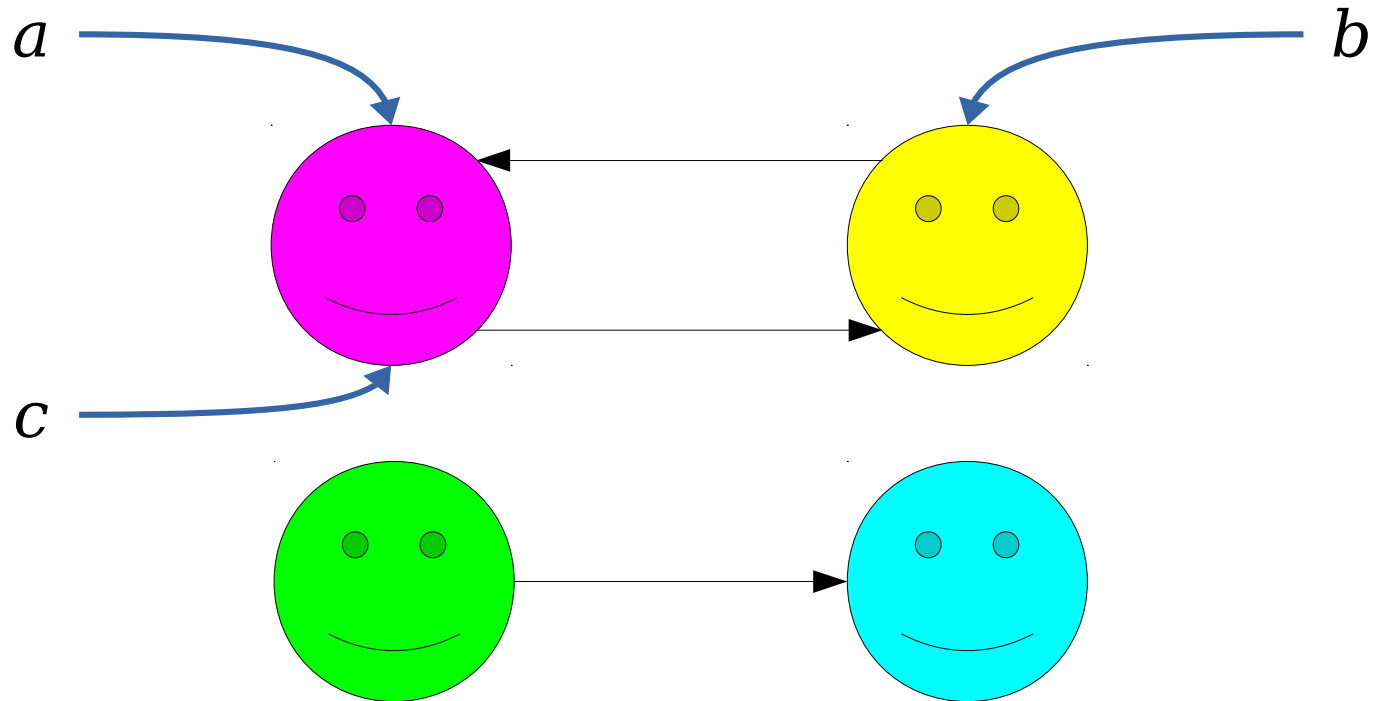
# Is This Relation Transitive?



**$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$**

*("Whenever a is related to b and b is related to c, we know a is related to c.")*

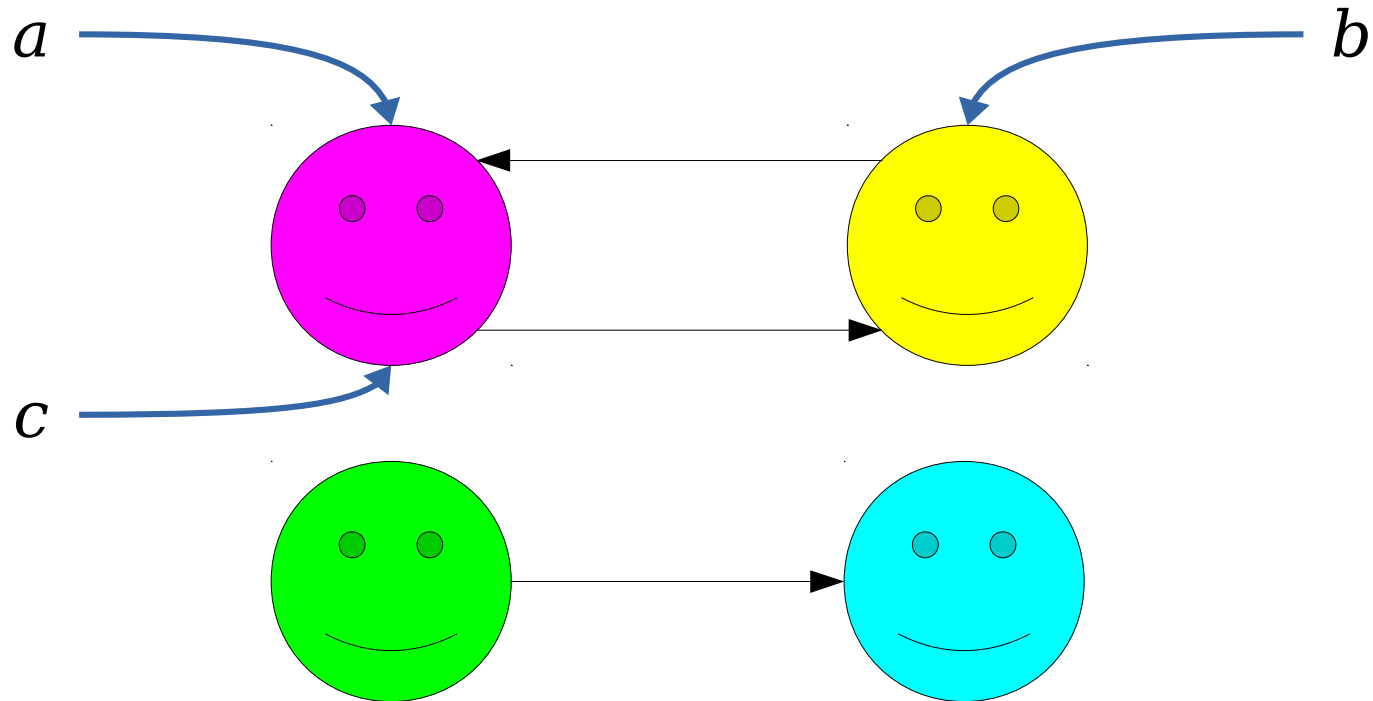
# Is This Relation Transitive?



**$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$**

*("Whenever  $a$  is related to  $b$  and  $b$  is related to  $c$ , we know  $a$  is related to  $c$ .)*

# Is This Relation Transitive?



$\forall a \in A. \forall b \in A. \forall c \in A. (aRb \wedge bRc \rightarrow aRc)$

*(“Whenever  $a$  is related to  $b$  and  $b$  is related to  $c$ , we know  $a$  is related to  $c$ .)*





# Equivalence Relations

- An ***equivalence relation*** is a relation that is reflexive, symmetric and transitive.
- Some examples:
  - $x = y$
  - $x \equiv_k y$
  - $x$  has the same color as  $y$
  - $x$  has the same shape as  $y$ .

Binary relations give us a ***common language*** to describe ***common structures***.

# Equivalence Relations

- Most modern programming languages include some sort of hash table data structure.
  - Java: `HashMap`
  - C++: `std::unordered_map`
  - Python: `dict`
- If you insert a key/value pair and then try to look up a key, the implementation has to be able to tell whether two keys are equal.
- Although each language has a different mechanism for specifying this, many languages describe them in similar ways...

# Equivalence Relations

“The equals method implements an equivalence relation on non-null object references:

- It is *reflexive*: for any non-null reference value `x`, `x.equals(x)` should return true.
- It is *symmetric*: for any non-null reference values `x` and `y`, `x.equals(y)` should return true if and only if `y.equals(x)` returns true.
- It is *transitive*: for any non-null reference values `x`, `y`, and `z`, if `x.equals(y)` returns true and `y.equals(z)` returns true, then `x.equals(z)` should return true.”

Java 8 Documentation

# Equivalence Relations

“The equals method implements an equivalence relation on non-null object references:

- It is *reflexive*: for any non-null reference value  $x$ ,  $x.equals(x)$  should return true.
- It is *symmetric*: for any non-null reference values  $x$  and  $y$ ,  $x.equals(y)$  should return true if and only if  $y.equals(x)$  returns true.
- It is *transitive*: for any non-null reference values  $x$ ,  $y$ , and  $z$ , if  $x.equals(y)$  returns true and  $y.equals(z)$  returns true, then  $x.equals(z)$  should return true.”

Java 8 Documentation

# Equivalence Relations

“Each unordered associative container is parameterized by `Key`, by a function object type `Hash` that meets the `Hash` requirements (17.6.3.4) and acts as a hash function for argument values of type `Key`, and by a binary predicate `Pred` that induces an equivalence relation on values of type `Key`. Additionally, `unordered_map` and `unordered_multimap` associate an arbitrary mapped type `T` with the `Key`.”

C++14 ISO Spec, §23.2.5/3

# Equivalence Relations

“Each unordered associative container is parameterized by `Key`, by a function object type `Hash` that meets the `Hash` requirements (17.6.3.4) and acts as a hash function for argument values of type `Key`, **and by a binary predicate `Pred` that induces an equivalence relation on values of type `Key`**. Additionally, `unordered_map` and `unordered_multimap` associate an arbitrary mapped type `T` with the `Key`.”

C++14 ISO Spec, §23.2.5/3

**Time-Out for Announcements!**



# Girls Teaching Girls to Code Exec Team Applications are open now!

Want to inspire the next generation of women in tech?  
Passionate about CS education and outreach?

Join the Executive Team of Girls Teaching Girls to Code!



Girls Teaching Girls to Code is a program for Stanford students to teach high school girls from around the Bay Area to code. Students learn programming basics, build exciting projects, and develop strong relationships with mentors in the field.

We are currently accepting applicants from all majors who are interested in building our corporate relations, coordinating logistics for our events, creating and reviewing curricula, or supporting outreach efforts in the community.

This year, we brought together over 200 high school girls and 60 Stanford mentors for a day-long programming experience. Aside from our main event, we also hosted smaller events throughout the year. We received overwhelmingly positive feedback, and we're looking for team leads help us grow GTGTC even more!

Learn more about us at <http://www.girlsteachinggirlstocode.org/>, and apply at <https://bit.ly/2y3CkTb> by Friday, October 12, 2018 at 11:59 pm!

# Problem Set One Solutions

- We've just released solutions to Problem Set One, both in hardcopy and online.
- ***You need to read over these solutions as soon as possible.***
- Why?
  - Each question is there for a reason. We've described what it is that we hoped you would have learned when solving those problems.
  - There are lots of different ways of solving these problems. Comparing what you did against our solutions, which are just one possible set of solutions, can help introduce new techniques.

# Problem Set Two

- The Problem Set Two checkpoint was due today at 2:30PM.
- The remaining problems are due Friday at 2:30PM.
- Have questions?
  - Stop by office hours!
  - Ask on Piazza!

Your Questions

“I’m considering an independent contractor job. The contract is confusing- I don’t want to get “screwed,” per se. Where can I get legal advice as an undergrad?”

You can get free legal advice from the ASSU (check [this link](#) for more details.)

I’d like to hear more about why you’re considering this job. If you need the money and they’re offering to pay well, or if it’s for a project you’re genuinely excited about, great!

If you’re trying to improve your coding skills, make sure that you’re actually signing up for something where that will happen. You learn best when you’re surrounded by other skilled people who know what they’re doing. Is that the case here?

If you’re doing this to improve your resume – are you sure you need to do that? You aren’t expected to have past job experience when applying for internships and the like.

# “When’s a good time and what’s the best way to tell my PSet partner I like them?”

Think of this as akin to asking a coworker out on a date.

1. Keep the Categorical Imperative in mind and remember that many people legitimately do not like being hit on in the workplace. As an undergrad there is basically no separation between “home” and “the workplace,” but working on problem sets together is way more of “the workplace” than “home.”
2. Think of the position you are putting the other person in. Your ask comes with a real risk of disrupting their work schedule. General advice: give people face-saving, minimally-disruptive ways to say no.
3. The more important question is “do you have a reason to believe the other person enjoys your company?” Because if the answer is no, then the answer is “never,” since otherwise you’re weird and awkward.

Back to CS103!

# Equivalence Relation Proofs

- Let's suppose you've found a binary relation  $R$  over a set  $A$  and want to prove that it's an equivalence relation.
- How exactly would you go about doing this?



# An Example Relation

- Consider the binary relation  $\sim$  defined over the set  $\mathbb{Z}$ :

$$a \sim b \quad \text{if} \quad a+b \text{ is even}$$

- Some examples:

$$0 \sim 4 \quad 1 \sim 9 \quad 2 \sim 6 \quad 5 \sim 5$$

- Turns out, this is an equivalence relation! Let's see how to prove it.

We can binary relations by giving a rule, like this:

$$a \sim b \quad \text{if} \quad \text{some property of } a \text{ and } b \text{ holds}$$

***This is the general template for defining a relation.***

Although we're using “if” rather than “iff” here, the two above statements are definitionally equivalent. For a variety of reasons, definitions are often introduced with “if” rather than “iff.” Check the “Mathematical Vocabulary” handout for details.

What properties must  $\sim$  have to be an equivalence relation?

***Reflexivity***

***Symmetry***

***Transitivity***

Let's prove each property independently.

$a \sim b$  if  $a+b$  is even

***Lemma 1:*** The binary relation  $\sim$  is reflexive.

$a \sim b$  if  $a+b$  is even

***Lemma 1:*** The binary relation  $\sim$  is reflexive.

***Proof:***

$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:**

What is the formal definition of reflexivity?

$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:**

What is the formal definition of reflexivity?

$$\forall a \in \mathbb{Z}. a \sim a$$

$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:**

What is the formal definition of reflexivity?

$$\forall a \in \mathbb{Z}. a \sim a$$

Therefore, we'll choose an arbitrary integer  $a$ , then go prove that  $a \sim a$ .

$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:**

What is the formal definition of reflexivity?

$$\forall a \in \mathbb{Z}. a \sim a$$

Therefore, we'll choose an arbitrary integer  $a$ , then go prove that  $a \sim a$ .



$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:**

What is the formal definition of reflexivity?

$$\forall a \in \mathbb{Z}. a \sim a$$

Therefore, we'll choose an arbitrary integer  $a$ , then go prove that  $a \sim a$ .

**$a \sim b$  if  $a+b$  is even**

***Lemma 1:*** The binary relation  $\sim$  is reflexive.

***Proof:*** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ .

**$a \sim b$  if  $a+b$  is even**

***Lemma 1:*** The binary relation  $\sim$  is reflexive.

***Proof:*** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ . From the definition of the  $\sim$  relation, this means that we need to prove that  $a+a$  is even.

**$a \sim b$  if  $a+b$  is even**

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ . From the definition of the  $\sim$  relation, this means that we need to prove that  $a+a$  is even.

To see this, notice that  $a+a = 2a$ , so the sum  $a+a$  can be written as  $2k$  for some integer  $k$  (namely,  $a$ ), so  $a+a$  is even.

**$a \sim b$  if  $a+b$  is even**

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ . From the definition of the  $\sim$  relation, this means that we need to prove that  $a+a$  is even.

To see this, notice that  $a+a = 2a$ , so the sum  $a+a$  can be written as  $2k$  for some integer  $k$  (namely,  $a$ ), so  $a+a$  is even. Therefore,  $a \sim a$  holds, as required.

**$a \sim b$  if  $a+b$  is even**

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ . From the definition of the  $\sim$  relation, this means that we need to prove that  $a+a$  is even.

To see this, notice that  $a+a = 2a$ , so the sum  $a+a$  can be written as  $2k$  for some integer  $k$  (namely,  $a$ ), so  $a+a$  is even. Therefore,  $a \sim a$  holds, as required. ■

**$a \sim b$  if  $a+b$  is even**

***Lemma 2:*** The binary relation  $\sim$  is symmetric.

$a \sim b$  if  $a+b$  is even

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**



$a \sim b$  if  $a+b$  is even

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**

What is the formal definition of symmetry?

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**

What is the formal definition of symmetry?

$$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. (a \sim b \rightarrow b \sim a)$$

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**

What is the formal definition of symmetry?

$$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. (a \sim b \rightarrow b \sim a)$$

Therefore, we'll choose arbitrary integers  **$a$**  and  **$b$**  where  **$a \sim b$** , then prove that  **$b \sim a$** .

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**

What is the formal definition of symmetry?

$$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. (a \sim b \rightarrow b \sim a)$$

Therefore, we'll choose arbitrary integers  **$a$**  and  **$b$**  where  **$a \sim b$** , then prove that  **$b \sim a$** .

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:**

What is the formal definition of symmetry?

$$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. (a \sim b \rightarrow b \sim a)$$

Therefore, we'll choose arbitrary integers  $a$  and  $b$  where  $a \sim b$ , then prove that  $b \sim a$ .

**$a \sim b$  if  $a+b$  is even**

***Lemma 2:*** The binary relation  $\sim$  is symmetric.

***Proof:*** Consider any integers  $a$  and  $b$  where  $a \sim b$ .  
We need to show that  $b \sim a$ .

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:** Consider any integers  $a$  and  $b$  where  $a \sim b$ .  
We need to show that  $b \sim a$ .

Since  $a \sim b$ , we know that  $a+b$  is even.

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:** Consider any integers  $a$  and  $b$  where  $a \sim b$ .  
We need to show that  $b \sim a$ .

Since  $a \sim b$ , we know that  $a+b$  is even. Because  $a+b = b+a$ , this means that  $b+a$  is even.



**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:** Consider any integers  $a$  and  $b$  where  $a \sim b$ . We need to show that  $b \sim a$ .

Since  $a \sim b$ , we know that  $a+b$  is even. Because  $a+b = b+a$ , this means that  $b+a$  is even. Since  $b+a$  is even, we know that  $b \sim a$ , as required.

**$a \sim b$  if  $a+b$  is even**

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:** Consider any integers  $a$  and  $b$  where  $a \sim b$ . We need to show that  $b \sim a$ .

Since  $a \sim b$ , we know that  $a+b$  is even. Because  $a+b = b+a$ , this means that  $b+a$  is even. Since  $b+a$  is even, we know that  $b \sim a$ , as required. ■

**$a \sim b$  if  $a+b$  is even**

***Lemma 3:*** The binary relation  $\sim$  is transitive.

**$a \sim b$  if  $a+b$  is even**

***Lemma 3:*** The binary relation  $\sim$  is transitive.

***Proof:***

$a \sim b$  if  $a+b$  is even

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:**

What is the formal definition of transitivity?

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:**

What is the formal definition of transitivity?

**$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \forall c \in \mathbb{Z}. (a \sim b \wedge b \sim c \rightarrow a \sim c)$**

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:**

What is the formal definition of transitivity?

**$\forall a \in \mathbb{Z}. \forall b \in \mathbb{Z}. \forall c \in \mathbb{Z}. (a \sim b \wedge b \sim c \rightarrow a \sim c)$**

Therefore, we'll choose arbitrary integers  **$a$** ,  **$b$** , and  **$c$**   
where  **$a \sim b$**  and  **$b \sim c$** , then prove that  **$a \sim c$** .

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ .



**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even.

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ .

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

Rearranging, we see that

$$a+c + 2b = 2k + 2m,$$

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

Rearranging, we see that

$$a+c + 2b = 2k + 2m,$$

so

$$a+c = 2k + 2m - 2b = 2(k+m-b).$$

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

Rearranging, we see that

$$a+c + 2b = 2k + 2m,$$

so

$$a+c = 2k + 2m - 2b = 2(k+m-b).$$

So there is an integer  $r$ , namely  $k+m-b$ , such that  $a+c = 2r$ . Thus  $a+c$  is even, so  $a \sim c$ , as required.

**$a \sim b$  if  $a+b$  is even**

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

Rearranging, we see that

$$a+c + 2b = 2k + 2m,$$

so

$$a+c = 2k + 2m - 2b = 2(k+m-b).$$

So there is an integer  $r$ , namely  $k+m-b$ , such that  $a+c = 2r$ . Thus  $a+c$  is even, so  $a \sim c$ , as required. ■



# An Observation

$a \sim b$  if  $a+b$  is even

**Lemma 1:** The binary relation  $\sim$  is reflexive.

**Proof:** Consider an arbitrary  $a \in \mathbb{Z}$ . We need to prove that  $a \sim a$ . From the definition of the  $\sim$  relation, this means that we need to prove that  $a+a$  is even.

To see this, notice that  $a+a = 2a$ , so the sum  $a+a$  can be written as  $2k$  for some integer  $k$  (namely,  $a$ ), so  $a+a$  is even. Therefore,  $a \sim a$  holds, as required. ■

The formal definition of reflexivity is given in first-order logic, but this proof does not contain any first-order logic symbols!

$a \sim b$  if  $a+b$  is even

**Lemma 2:** The binary relation  $\sim$  is symmetric.

**Proof:** Consider any integers  $a$  and  $b$  where  $a \sim b$ . We need to show that  $b \sim a$ .

Since  $a \sim b$ , we know that  $a+b$  is even. Because  $a+b = b+a$ , this means that  $b+a$  is even. Since  $b+a$  is even, we know that  $b \sim a$ , as required. ■

The formal definition of symmetry is given in first-order logic, but this proof does not contain any first-order logic symbols!

## $a \sim b$ if $a+b$ is even

**Lemma 3:** The binary relation  $\sim$  is transitive.

**Proof:** Consider arbitrary integers  $a$ ,  $b$  and  $c$  where  $a \sim b$  and  $b \sim c$ . We need to prove that  $a \sim c$ , meaning that we need to show that  $a+c$  is even.

Since  $a \sim b$  and  $b \sim c$ , we know that  $a+b$  and  $b+c$  are even. This means there are integers  $k$  and  $m$  where  $a+b = 2k$  and  $b+c = 2m$ . Notice that

$$(a+b) + (b+c) = 2k + 2m.$$

Rearranging, we see that

$$a+c + 2b = 2k + 2m,$$

so

$$a+c = 2k +$$

So there is an integer  $r$ ,  
 $a+c = 2r$ . Thus  $a+c$  is even.

The formal definition of transitivity is given in first-order logic, but this proof does not contain any first-order logic symbols!

# First-Order Logic and Proofs

- First-order logic is an excellent tool for giving formal definitions to key terms.
- While first-order logic *guides* the structure of proofs, it is *exceedingly rare* to see first-order logic in written proofs.
- Follow the example of these proofs:
  - Use the FOL definitions to determine what to assume and what to prove.
  - Write the proof in plain English using the conventions we set up in the first week of the class.
- ***Please, please, please, please, please internalize the contents of this slide!***

# Next Time

- ***A Fundamental Theorem***
  - Why are equivalence relations so useful?
- ***Proofs on Relations***
  - Proving properties of abstract objects.
- ***Strict Orders***
  - Representing rankings and requirements.
- ***Hasse Diagrams***
  - How to draw strict orders.