Poll: How big is infinity?

(A) There are more real numbers than natural numbers.
(B) There are more rational numbers than natural numbers.
(C) There are more integers than natural numbers.
(D) Pairs of natural numbers.
Poll: How big is infinity?

Mark what’s true.
(A) There are more real numbers than natural numbers.
(B) There are more rational numbers than natural numbers.
(C) There are more integers than natural numbers.
(D) pairs of natural numbers $>>$ natural numbers.
Two sets are the same size?
Two sets are the same size?

(A) Bijection between the sets.
(B) Count the objects and get the same number. same size.
(C) Counting to infinity is hard.
Two sets are the same size?

(A) Bijection between the sets.
(B) Count the objects and get the same number. same size.
(C) Counting to infinity is hard.

(A), (B).
Two sets are the same size?

(A) Bijection between the sets.
(B) Count the objects and get the same number. same size.
(C) Counting to infinity is hard.

(A), (B).
(C)?
Countable.

How to count?
Countable.

How to count?

0,
Countable.

How to count?

0, 1,
Countable.

How to count?
0, 1, 2,
How to count?
0, 1, 2, 3,
Countable.

How to count?
0, 1, 2, 3, ...
How to count?
0, 1, 2, 3, ...
The Counting numbers.
Countable.

How to count?

0, 1, 2, 3, …

The Counting numbers.
The natural numbers! $N$
Countable.

How to count?
0, 1, 2, 3, …
The Counting numbers.
The natural numbers! $N$

Definition: $S$ is **countable** if there is a bijection between $S$ and some subset of $N$. 
How to count?
0, 1, 2, 3, ... 

The Counting numbers.
The natural numbers! $N$

Definition: $S$ is countable if there is a bijection between $S$ and some subset of $N$.

If the subset of $N$ is finite, $S$ has finite cardinality.
Countable.

How to count?
0, 1, 2, 3, …
The Counting numbers.
The natural numbers! \( N \)

Definition: \( S \) is countable if there is a bijection between \( S \) and some subset of \( N \).

If the subset of \( N \) is finite, \( S \) has finite cardinality.
If the subset of \( N \) is infinite, \( S \) is countably infinite.
Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.
Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.
Enumerate $T$ as follows:
Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$, 

$\mathbb{Z}^+$ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
So it is countably infinite.
All countably infinite sets have the same cardinality.
Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$,
output only if $x \in T$. 
Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$,
output only if $x \in T$. 

Implications:
$\mathbb{Z}^+$ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
So it is countably infinite.
All countably infinite sets have the same cardinality.
Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset \( T \) of a countable set \( S \) is countable.

Enumerate \( T \) as follows:
Get next element, \( x \), of \( S \),
output only if \( x \in T \).

Implications:
Countably infinite subsets.

Enumerating a set implies countable. 

Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$,
output only if $x \in T$.

Implications:
$\mathbb{Z}^+$ is countable.
Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$,
output only if $x \in T$.

Implications:
$\mathbb{Z}^+$ is countable.
It is infinite since the list goes on.
Countably infinite subsets.

Enumerating a set implies countable. Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$,
output only if $x \in T$.

Implications:
$\mathbb{Z}^+$ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
Countably infinite subsets.

Enumerating a set implies countable. Corollary: Any subset $T$ of a countable set $S$ is countable.

Enumerate $T$ as follows:
Get next element, $x$, of $S$, output only if $x \in T$.

Implications:
$\mathbb{Z}^+$ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
So it is countably infinite.
Countably infinite subsets.

Enumerating a set implies countable.  
Corollary: Any subset $T$ of a countable set $S$ is countable.  

Enumerate $T$ as follows:  
Get next element, $x$, of $S$,  
output only if $x \in T$.  

Implications:  
$\mathbb{Z}^+$ is countable.  
It is infinite since the list goes on.  
There is a bijection with the natural numbers.  
So it is countably infinite.  

All countably infinite sets have the same cardinality.
Enumeration example.

All binary strings.
Enumeration example.

All binary strings.
\( B = \{0, 1\}^* \).
All binary strings.
\[ B = \{0, 1\}^* \]
Enumeration example.

All binary strings.
\[ B = \{0, 1\}^* \]
\[ B = \{\phi, 0, \]
All binary strings.
\[ B = \{0, 1\}^* . \]
\[ B = \{\phi, 0, 1, \ldots \} . \]
\[ \phi \text{ is empty string.} \]
For any string, it appears at some position in the list.
If \( n \) bits, it will appear before position \( 2^n + 1 \).
Should be careful here.
Enumeration example.

All binary strings.

$B = \{0, 1\}^*.$

$B = \{\phi, 0, 1, 00,$
All binary strings.
$B = \{0, 1\}^*.$
$B = \{\phi, 0, 1, 00, 01, 10, 11, \ldots\}.$
Enumeration example.

All binary strings.

\[ B = \{0, 1\}^*. \]

\[ B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}. \]
Enumeration example.

All binary strings.
$B = \{0, 1\}^*$. 

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.

$\phi$ is empty string.
Enumeration example.

All binary strings.
$B = \{0, 1\}^*$. 

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$. 
$\phi$ is empty string.

For any string, it appears at some position in the list.
All binary strings.
\(B = \{0, 1\}^*\).

\(B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}\).
\(\phi\) is empty string.

For any string, it appears at some position in the list. If \(n\) bits, it will appear before position \(2^{n+1}\).
All binary strings.
$B = \{0, 1\}^*$.  

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.  
$\phi$ is empty string.

For any string, it appears at some position in the list.  
If $n$ bits, it will appear before position $2^{n+1}$.  

Should be careful here.
Enumeration example.

All binary strings.

\[ B = \{0,1\}^*. \]

\[ B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}. \]

\( \phi \) is empty string.

For any string, it appears at some position in the list. If \( n \) bits, it will appear before position \( 2^{n+1} \).

Should be careful here.

\[ B = \{\phi; 0, 00, 000, 0000, \ldots\} \]
Enumeration example.

All binary strings.
\( B = \{0, 1\}^* \).

\( B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots \} \).
\( \phi \) is empty string.

For any string, it appears at some position in the list. If \( n \) bits, it will appear before position \( 2^{n+1} \).

Should be careful here.

\( B = \{\phi; , 0, 00, 000, 0000, \ldots \} \)

Never get to 1.
More fractions?

Enumerate the rational numbers in order...
More fractions?

Enumerate the rational numbers in order...

0, ..., 1/2, ..
Enumerate the rational numbers in order...
0, ..., 1/2, ..
Where is 1/2 in list?
More fractions?

Enumerate the rational numbers in order...

0, ..., 1/2, ...

Where is 1/2 in list?

After 1/3, which is after 1/4, which is after 1/5...
More fractions?

Enumerate the rational numbers in order...
0,...,1/2,..

Where is 1/2 in list?

After 1/3, which is after 1/4, which is after 1/5...

A thing about fractions:
More fractions?

Enumerate the rational numbers in order...
0, ..., 1/2, ...
Where is 1/2 in list?
After 1/3, which is after 1/4, which is after 1/5...
A thing about fractions:
any two fractions has another fraction between it.
Enumerate the rational numbers in order...
0,...,1/2,..

Where is 1/2 in list?

After 1/3, which is after 1/4, which is after 1/5...

A thing about fractions:
any two fractions has another fraction between it.
Can’t even get to “next” fraction!
More fractions?

Enumerate the rational numbers in order...
0, ..., 1/2, ..

Where is 1/2 in list?

After 1/3, which is after 1/4, which is after 1/5...

A thing about fractions:
any two fractions has another fraction between it.
Can’t even get to “next” fraction!
Can’t list in “order”.
Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
Pairs of natural numbers.

Consider pairs of natural numbers: \( N \times N \)
E.g.: (1,2), (100,30), etc.
Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$

E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets $S_1$ and $S_2$, 
Consider pairs of natural numbers: $N \times N$
E.g.: $(1, 2)$, $(100, 30)$, etc.
For finite sets $S_1$ and $S_2$, then $S_1 \times S_2$
Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: (1, 2), (100, 30), etc.

For finite sets $S_1$ and $S_2$, then $S_1 \times S_2$
has size $|S_1| \times |S_2|$. 
Consider pairs of natural numbers: $N \times N$

E.g.: $(1, 2)$, $(100, 30)$, etc.

For finite sets $S_1$ and $S_2$, then $S_1 \times S_2$
has size $|S_1| \times |S_2|$. 
Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets $S_1$ and $S_2$,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite
Consider pairs of natural numbers: $N \times N$
E.g.: $(1, 2)$, $(100, 30)$, etc.

For finite sets $S_1$ and $S_2$, then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite squared
Consider pairs of natural numbers: $N \times N$

E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets $S_1$ and $S_2$,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite squared ???
Pairs of natural numbers.

Enumerate in list:

- \((0, 0)\)
- \((1, 0)\)
- \((0, 1)\)
- \((2, 0)\)
- \((1, 1)\)
- \((0, 2)\)
- \((3, 0)\)
- \((2, 1)\)
- \((1, 2)\)
- \((0, 3)\)
- \((4, 0)\)
- \((3, 1)\)
- \((2, 2)\)
- \((1, 3)\)
- \((0, 4)\)
- \((5, 0)\)
- \((4, 1)\)
- \((3, 2)\)
- \((2, 3)\)
- \((1, 4)\)
- \((0, 5)\)

The pair \((a, b)\), is in first ≈ \((a + b + 1)(a + b) / 2\) elements of list! (i.e., "triangle").

Countably infinite. Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0, 0),
(1, 0),
(0, 1),
(2, 0),
(1, 1),
(0, 2),
......
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0),
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0), (0, 1),
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0), (0, 1), (2, 0),
Pairs of natural numbers.

Enumerate in list:

$(0, 0), (1, 0), (0, 1), (2, 0), (1, 1), \ldots$
Pairs of natural numbers.

Enumerate in list:
\[(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), \ldots \]

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list! (i.e., "triangle").

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list!

(i.e., “triangle”).

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b) / 2\) elements of list!

(i.e., “triangle”).

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:

(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair (a, b), is in first ≈ (a + b + 1)(a + b)/2 elements of list!

(i.e., “triangle”)

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair (a, b), is in first ≈ (a + b + 1)(a + b)/2 elements of list!

(i.e., “triangle”).

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:

(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), …

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list!

(i.e., "triangle").

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), …

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list! (i.e., “triangle”).

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b) / 2\) elements of list!

(i.e., “triangle”)

Countably infinite. Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), …

The pair \((a, b)\), is in first \(\approx \frac{a + b + 1}{2}\) elements of list!

(i.e., "triangle").

Countably infinite.
Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (0, 2), ……

The pair \((a, b)\), is in first ≈ \((a + b + 1)(a + b) / 2\) elements of list!

(i.e., "triangle").

Countably infinite.

Same size as the natural numbers!!
Pairs of natural numbers.

Enumerate in list:

\[(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), \ldots\]

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list!
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (0, 2), ……

The pair \((a, b)\), is in first \( \approx (a + b + 1)(a + b)/2 \) elements of list!
(i.e., “triangle”).
Pairs of natural numbers.

Enumerate in list:
(0,0), (1,0), (0,1), (2,0), (1,1), (0,2), ……

The pair \((a, b)\) is in first \(\approx (a + b + 1)(a + b)/2\) elements of list!
(i.e., “triangle”).

Countably infinite.
Pairs of natural numbers.

Enumerate in list:
(0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (0, 2), …

The pair \((a, b)\), is in first \(\approx (a + b + 1)(a + b)/2\) elements of list!
(i.e., “triangle”).

Countably infinite.

Same size as the natural numbers!!
Poll.

Enumeration to get bijection with naturals?
Enumeration to get bijection with naturals?

(A) Integers: First all negatives, then positives.
(B) Integers: By absolute value, break ties however.
(C) Pairs of naturals: by sum of values, break ties however.
(D) Pairs of naturals: by value of first element.
(E) Pairs of integers: by sum of values, break ties.
(F) Pairs of integers: by sum of absolute values, break ties.
Poll.

Enumeration to get bijection with naturals?

(A) Integers: First all negatives, then positives.
(B) Integers: By absolute value, break ties however.
(C) Pairs of naturals: by sum of values, break ties however.
(D) Pairs of naturals: by value of first element.
(E) Pairs of integers: by sum of values, break ties.
(F) Pairs of integers: by sum of absolute values, break ties.

(B), (C), (F).
Rationals?

Positive rational number.
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
Rationals?

Positive rational number.
Lowest terms: \(a/b\)
\(a, b \in N\)
Rationals?

Positive rational number.
Lowest terms: $a/b$
$a, b \in N$
with $gcd(a, b) = 1$. 
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).

Countably infinite!
Positive rational number. 

Lowest terms: \( a/b \)

\( a, b \in \mathbb{N} \)

with \( \gcd(a, b) = 1 \).

Infinite subset of \( \mathbb{N} \times \mathbb{N} \).

Countably infinite!

All rational numbers?
Rationals?

Positive rational number.
Lowest terms: \( \frac{a}{b} \)
\( a, b \in N \) with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).

Countably infinite!

All rational numbers?

Negative rationals are countable.
Rationals?

Positive rational number.  
Lowest terms: $a/b$  
$a, b \in N$  
with $gcd(a, b) = 1$.  

Infinite subset of $N \times N$.  

Countably infinite!  

All rational numbers?  

Negative rationals are countable. (Same size as positive rationals.)
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in \mathbb{N} \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( \mathbb{N} \times \mathbb{N} \).
Countably infinite!

All rational numbers?
Negative rationals are countable. (Same size as positive rationals.)
Put all rational numbers in a list.
Rationals?

Positive rational number.
Lowest terms: \(a/b\)
\(a, b \in N\)
with \(gcd(a, b) = 1\).

Infinite subset of \(N \times N\).

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.
First negative, then nonegative
Positive rational number.
Lowest terms: \( \frac{a}{b} \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).
Countably infinite!

All rational numbers?
Negative rationals are countable. (Same size as positive rationals.)
Put all rational numbers in a list.
First negative, then nonegative ???
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).
Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)
Put all rational numbers in a list.
First negative, then nonegative ??? No!
Repeatedly and alternatively take one from each list.
Rationals?

Positive rational number.
Lowest terms: \( \frac{a}{b} \)
\( a, b \in N \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( N \times N \).

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Repeatedly and alternatively take one from each list.

Interleave Streams in 61A
Rationals?

Positive rational number.
Lowest terms: \( a/b \)
\( a, b \in \mathbb{N} \)
with \( \gcd(a, b) = 1 \).

Infinite subset of \( \mathbb{N} \times \mathbb{N} \).

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Repeatedly and alternatively take one from each list.
Interleave Streams in 61A

The rationals are countably infinite.
Real numbers are same size as integers?
The reals.

Are the set of reals countable?
The reals.

Are the set of reals countable?
Lets consider the reals $[0, 1]$. 
The reals.

Are the set of reals countable?
Lets consider the reals [0, 1].
Each real has a decimal representation.
Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

$.500000000...$
The reals.

Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

$0.500000000\ldots$ (1/2)
The reals.

Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.
0.500000000... (1/2)
0.785398162...
Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

$.500000000... \ (1/2)$
$.785398162... \ \pi/4$
The reals.

Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

$.500000000... \ (1/2)$

$.785398162... \ \pi/4$

$.367879441...$
Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

.500000000... (1/2)
.785398162... $\pi/4$
.367879441... $1/e$
Are the set of reals countable?

Let's consider the reals \([0, 1]\).

Each real has a decimal representation.

- \(0.500000000\ldots\) \((1/2)\)
- \(0.785398162\ldots\) \(\pi/4\)
- \(0.367879441\ldots\) \(1/e\)
- \(0.632120558\ldots\)
The reals.

Are the set of reals countable?

Let's consider the reals \([0, 1]\).

Each real has a decimal representation.

- \(0.500000000\ldots\) \((1/2)\)
- \(0.785398162\ldots\) \(\pi/4\)
- \(0.367879441\ldots\) \(1/e\)
- \(0.632120558\ldots\) \(1 - 1/e\)
Are the set of reals countable?

Let's consider the reals [0, 1].

Each real has a decimal representation.

.500000000... (1/2)
.785398162... \( \pi/4 \)
.367879441... 1/e
.632120558... 1 − 1/e
.345212312...
Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

- $0.500000000\ldots$ (1/2)
- $0.785398162\ldots$ $\pi/4$
- $0.367879441\ldots$ $1/e$
- $0.632120558\ldots$ $1 – 1/e$
- $0.345212312\ldots$ Some real number
Are the set of reals countable?

Let's consider the reals $[0, 1]$.

Each real has a decimal representation.

- $0.500000000... \ (1/2)$
- $0.785398162... \ \pi/4$
- $0.367879441... \ 1/e$
- $0.632120558... \ 1 – 1/e$
- $0.345212312... \ Some \ real \ number$
Diagonalization.

If countable, there a listing, \( L \) contains all reals.
Diagonalization.

If countable, there a listing, \( L \) contains all reals. For example
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...

Construct "diagonal" number: 

7
7
6
7...

Diagonal Number:

Digit $i$ is 7 if number $i$'s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0, 1]$ is not countable!!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...

Construct "diagonal" number: .77677...

Diagonal Number:

Digit $i$ is 7 if number $i$'s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0, 1]$ is not countable!!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...

Construct "diagonal" number:

.7767...

Diagonal Number:

Digit $i$ is 7 if number $i$'s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0,1]$ is not countable!!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
Diagonalization.

If countable, there a listing, \( L \) contains all reals. For example

0: \( .500000000... \)
1: \( .785398162... \)
2: \( .367879441... \)
3: \( .632120558... \)
4: \( .345212312... \)

\vdots

Construct "diagonal" number:

\[ .7767777... \]

Diagonal Number:

Digit \( i \) is 7 if number \( i \)'s \( i \)th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset \( [0, 1] \) is not countable!!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
...

Construct “diagonal” number:
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Construct “diagonal” number: .7
Diagonalization.

If countable, there a listing, \( L \) contains all reals. For example

0: .5000000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

\vdots

Construct “diagonal” number: .77
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: \[.500000000\ldots\]
1: \[.785398162\ldots\]
2: \[.367879441\ldots\]
3: \[.632120558\ldots\]
4: \[.345212312\ldots\]

: 

Construct “diagonal” number: \[.776\]
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: 0.5000000000...
1: 0.785398162...
2: 0.367879441...
3: 0.632120558...
4: 0.345212312...

Construct “diagonal” number: 0.7767
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example:

0: .5000000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
...

Construct “diagonal” number: .77677
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Construct “diagonal” number: .77677…
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: \( .500000000 \ldots \)

1: \( .785398162 \ldots \)

2: \( .367879441 \ldots \)

3: \( .632120558 \ldots \)

4: \( .345212312 \ldots \)

...

Construct “diagonal” number: \( .77677 \ldots \)

Diagonal Number:
Diagonalization.

If countable, there a listing, \( L \) contains all reals. For example

0: \(.500000000...\)
1: \(.785398162...\)
2: \(.367879441...\)
3: \(.632120558...\)
4: \(.345212312...\)

Construct “diagonal” number: \(.77677...\)

Diagonal Number: Digit \( i \) is 7 if number \( i \)'s \( i \)th digit is not 7
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

... 

Construct “diagonal” number: .77677...

Diagonal Number: Digit $i$ is 7 if number $i$’s $i$th digit is not 7 and 6 otherwise.
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
...

Construct “diagonal” number: .77677...

Diagonal Number: Digit $i$ is 7 if number $i$’s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Construct “diagonal” number: .77677...

Diagonal Number: Digit $i$ is 7 if number $i$’s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list! Diagonal number not in list.
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: .5000000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Construct “diagonal” number: .77677...

Diagonal Number: Digit $i$ is 7 if number $i$’s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.
Diagonalization.

If countable, there a listing, \( L \) contains all reals. For example

0: \( .500000000... \)
1: \( .785398162... \)
2: \( .367879441... \)
3: \( .632120558... \)
4: \( .345212312... \)

Construct “diagonal” number: \( .77677... \)

Diagonal Number: Digit \( i \) is 7 if number \( i \)'s \( i \)th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!
Diagonalization.

If countable, there a listing, $L$ contains all reals. For example

0: $0.500000000\ldots$
1: $0.785398162\ldots$
2: $0.367879441\ldots$
3: $0.632120558\ldots$
4: $0.345212312\ldots$

Construct “diagonal” number: $0.77677\ldots$

Diagonal Number: Digit $i$ is 7 if number $i$’s $i$th digit is not 7 and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0, 1]$ is not countable!!
All reals?

Subset $[0, 1]$ is not countable!!
All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.
All reals?

Subset $[0, 1]$ is not countable!!
What about all reals?
No.

Any subset of a countable set is countable.
All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

Any subset of a countable set is countable.
If reals are countable then so is $[0, 1]$. 
Diagonalization.

1. Assume that a set $S$ can be enumerated.
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$. 
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$.
3. Use the diagonal from the list to construct a new element $t$.
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$.
3. Use the diagonal from the list to construct a new element $t$.
4. Show that $t$ is different from all elements in the list
5. Show that $t$ is in $S$.
6. Contradiction.
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$.
3. Use the diagonal from the list to construct a new element $t$.
4. Show that $t$ is different from all elements in the list $\implies t$ is not in the list.
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$.
3. Use the diagonal from the list to construct a new element $t$.
4. Show that $t$ is different from all elements in the list $\implies t$ is not in the list.
5. Show that $t$ is in $S$. 

Contradiction.
Diagonalization.

1. Assume that a set $S$ can be enumerated.
2. Consider an arbitrary list of all the elements of $S$.
3. Use the diagonal from the list to construct a new element $t$.
4. Show that $t$ is different from all elements in the list $\implies t$ is not in the list.
5. Show that $t$ is in $S$.
6. Contradiction.
Another diagonalization.

The set of all subsets of $N$. 
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: $\{0\}$,
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: $\{0\}$, $\{0, \ldots, 7\}$,
Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \( \{0\} \), \( \{0, \ldots , 7\} \),
Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \( \{0\} \), \( \{0, \ldots, 7\} \),
evens,
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: $\{0\}$, $\{0, \ldots, 7\}$, evens, odds,
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: \{0\}, \{0, \ldots, 7\}, evens, odds, primes,
Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \{0\}, \{0, \ldots, 7\}, evens, odds, primes,
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: $\{0\}$, $\{0, \ldots, 7\}$, evens, odds, primes,

Assume is countable.
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$. 

\[ \text{Contradiction.} \]

Theorem: The set of all subsets of $N$ is not countable.

(The set of all subsets of $S$, is the powerset of $N$.)
Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \{0\}, \{0,\ldots,7\}, evens, odds, primes,

Assume is countable.

There is a listing, \( L \), that contains all subsets of \( N \).

Define a diagonal set, \( D \):
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.  

Theorem: The set of all subsets of $N$ is not countable.
(The set of all subsets of $S$, is the powerset of $N$.)


Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \( \{0\} \), \( \{0,\ldots,7\} \), evens, odds, primes,

Assume is countable.

There is a listing, \( L \), that contains all subsets of \( N \).

Define a diagonal set, \( D \):
If \( i \)th set in \( L \) does not contain \( i \), \( i \in D \).
otherwise \( i \notin D \).

Contradiction.

Theorem: The set of all subsets of \( N \) is not countable.

(The set of all subsets of \( S \), is the powerset of \( N \).)
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.
otherwise $i \notin D$. 

$D$ is different from $i$th set in $L$ for every $i$.
$= \Rightarrow D$ is not in the listing.
$D$ is a subset of $N$.
$L$ does not contain all subsets of $N$.

Contradiction.

Theorem: The set of all subsets of $N$ is not countable.
(The set of all subsets of $S$ is the powerset of $N$.)

Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:

If $i$th set in $L$ does not contain $i$, $i \in D$.

otherwise $i \notin D$.

$D$ is different from $i$th set in $L$ for every $i$.
Another diagonalization.

The set of all subsets of \( N \).

Example subsets of \( N \): \( \{0\} \), \( \{0, \ldots, 7\} \), evens, odds, primes,

Assume is countable.

There is a listing, \( L \), that contains all subsets of \( N \).

Define a diagonal set, \( D \):
If \( i \)th set in \( L \) does not contain \( i \), \( i \in D \).
otherwise \( i \notin D \).

\( D \) is different from \( i \)th set in \( L \) for every \( i \).
\( \rightarrow D \) is not in the listing.
Another diagonalization.

The set of all subsets of $\mathbb{N}$.

Example subsets of $\mathbb{N}$: $\{0\}$, $\{0, \ldots, 7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $\mathbb{N}$.

Define a diagonal set, $D$:

If $i$th set in $L$ does not contain $i$, $i \in D$.

otherwise $i \notin D$.

$D$ is different from $i$th set in $L$ for every $i$.

$\implies D$ is not in the listing.

$D$ is a subset of $\mathbb{N}$.
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: \{0\}, \{0, \ldots, 7\}, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.
otherwise $i \notin D$.

$D$ is different from $i$th set in $L$ for every $i$.
$\implies$ $D$ is not in the listing.

$D$ is a subset of $N$.

$L$ does not contain all subsets of $N$. 

(Theorem: The set of all subsets of $S$, is the powerset of $N$.)

Contradiction.
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.
otherwise $i \notin D$.

$D$ is different from $i$th set in $L$ for every $i$.
$\implies D$ is not in the listing.

$D$ is a subset of $N$.

$L$ does not contain all subsets of $N$.

Contradiction.
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: \{0\}, \{0,\ldots,7\}, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.
otherwise $i \not\in D$.

$D$ is different from $i$th set in $L$ for every $i$.
\[\implies D \text{ is not in the listing}.\]

$D$ is a subset of $N$.

$L$ does not contain all subsets of $N$.

Contradiction.

**Theorem:** The set of all subsets of $N$ is not countable.
Another diagonalization.

The set of all subsets of $N$.

Example subsets of $N$: $\{0\}$, $\{0,\ldots,7\}$, evens, odds, primes,

Assume is countable.

There is a listing, $L$, that contains all subsets of $N$.

Define a diagonal set, $D$:
If $i$th set in $L$ does not contain $i$, $i \in D$.
otherwise $i \notin D$.

$D$ is different from $i$th set in $L$ for every $i$.

$\implies D$ is not in the listing.

$D$ is a subset of $N$.

$L$ does not contain all subsets of $N$.

Contradiction.

**Theorem:** The set of all subsets of $N$ is not countable.
(The set of all subsets of $S$, is the powerset of $N$.)
Poll: diagonalization Proof.

Mark parts of proof.
Poll: diagonalization Proof.

Mark parts of proof.

(A) Integers are larger than naturals cuz obviously.
(B) Integers are countable cuz, interleaving bijection.
(C) Reals are uncountable cuz obviously!
(D) Reals can’t be in a list: diagonal number not on list.
(E) Powerset in list: diagonal set not in list.
Mark parts of proof.

(A) Integers are larger than naturals cuz obviously.
(B) Integers are countable cuz, interleaving bijection.
(C) Reals are uncountable cuz obviously!
(D) Reals can’t be in a list: diagonal number not on list.
(E) Powerset in list: diagonal set not in list.

(B), (C)?, (D), (E)
The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals.
The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals. First of Hilbert’s problems!
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$. 
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \to [0, 1]$. 

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2
\end{cases}$$
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$.

\[
f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases}
\]

One to one.
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \to [0, 1].$

\[f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases}\]

One to one. $x \neq y$
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \to [0, 1]$.

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases}$$

One to one. $x \neq y$

If both in $[0, 1/2]$, 
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \to [0, 1]$.

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2
\end{cases}$$

One to one. $x \neq y$

If both in $[0, 1/2]$, a shift
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \to [0, 1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$

If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$. 
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \rightarrow [0, 1]$.

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & \text{if } 0 \leq x \leq \frac{1}{2} \\
  \frac{1}{4x} & \text{if } x > \frac{1}{2}
\end{cases}$$

One to one. $x \neq y$

If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.

If neither in $[0, 1/2]$
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \to [0, 1]$. 

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2
\end{cases}$$

One to one. $x \neq y$
If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0, 1/2]$ a division
Cardinalities of uncountable sets?

Cardinality of \([0, 1]\) smaller than all the reals?

\[ f : R^+ \to [0, 1]. \]

\[ f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases} \]

One to one. \( x \neq y \)
If both in \([0, 1/2]\), a shift \(\implies f(x) \neq f(y)\).
If neither in \([0, 1/2]\) a division \(\implies f(x) \neq f(y)\).
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$.

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq \frac{1}{2} \\
  \frac{1}{4x} & x > \frac{1}{2}
\end{cases}$$

One to one. $x \neq y$
If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0, 1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0, 1/2]$ and one isn’t,
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : \mathbb{R}^+ \rightarrow [0, 1]$.  

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2
\end{cases}$$

One to one.  $x \neq y$
If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0, 1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0, 1/2]$ and one isn’t, different ranges
Cardinalities of uncountable sets?

Cardinality of \([0, 1]\) smaller than all the reals?

\(f : R^+ \rightarrow [0, 1].\)

\[
f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases}
\]

One to one. \(x \neq y\)

- If both in \([0, 1/2]\), a shift \(\implies f(x) \neq f(y).\)
- If neither in \([0, 1/2]\) a division \(\implies f(x) \neq f(y).\)
- If one is in \([0, 1/2]\) and one isn’t, different ranges \(\implies f(x) \neq f(y).\)
Cardinalities of uncountable sets?

Cardinality of \([0, 1]\) smaller than all the reals?

\(f : \mathbb{R}^+ \to [0, 1].\)

\[
f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2 
\end{cases}
\]

One to one. \(x \neq y\)

If both in \([0, 1/2]\), a shift \(\implies f(x) \neq f(y).\)

If neither in \([0, 1/2]\) a division \(\implies f(x) \neq f(y).\)

If one is in \([0, 1/2]\) and one isn’t, different ranges \(\implies f(x) \neq f(y).\)

Bijection!
Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$.

$$f(x) = \begin{cases} 
  x + \frac{1}{2} & 0 \leq x \leq 1/2 \\
  \frac{1}{4x} & x > 1/2
\end{cases}$$

One to one. $x \neq y$

If both in $[0, 1/2]$, a shift $\implies f(x) \neq f(y)$.

If neither in $[0, 1/2]$ a division $\implies f(x) \neq f(y)$.

If one is in $[0, 1/2]$ and one isn’t, different ranges $\implies f(x) \neq f(y)$.

Bijection!

$[0, 1]$ is same cardinality as nonnegative reals!
Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.
Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

The powerset of a set is the set of all subsets.
Resolution of hypothesis?

Gödel. 1940.

Can't use math!
If math doesn't contain a contradiction.
This statement is a lie.
Is the statement above true?
The barber shaves every person who does not shave themselves.
Who shaves the barber?
Self reference.
Can a program refer to a program?
Can a program refer to itself?
Uh oh....
Gödel. 1940.
Can’t use math!
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
Gödel. 1940.  
Can’t use math!  
If math doesn’t contain a contradiction.  
This statement is a lie.
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.
Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.
Who shaves the barber?

Self reference.
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.
Is the statement above true?
The barber shaves every person who does not shave themselves.
Who shaves the barber?
Self reference.
Can a program refer to a program?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.
Is the statement above true?
The barber shaves every person who does not shave themselves.
Who shaves the barber?
Self reference.
Can a program refer to a program?
Can a program refer to itself?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.
Is the statement above true?
The barber shaves every person who does not shave themselves.
Who shaves the barber?
Self reference.
Can a program refer to a program?
Can a program refer to itself?
Uh oh....
The barber shaves every person who does not shave themselves.
The barber shaves every person who does not shave themselves.

(A) Barber not Mark. Barber shaves Mark.
(B) Mark shaves the Barber.
(C) Barber doesn’t shave themself.
(D) Barber shaves themself.
The barber shaves every person who does not shave themselves.

(A) Barber not Mark. Barber shaves Mark.
(B) Mark shaves the Barber.
(C) Barber doesn’t shave themself.
(D) Barber shaves themself.

It's all true.
The Barber!

The barber shaves every person who does not shave themselves.

(A) Barber not Mark. Barber shaves Mark.
(B) Mark shaves the Barber.
(C) Barber doesn’t shave themself.
(D) Barber shaves themself.

Its all true. It’s all a problem.
Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.
Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

The powerset of a set is the set of all subsets.
Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

The powerset of a set is the set of all subsets.

Recall: powerset of the naturals is not countable.
Resolution of hypothesis?

Gödel. 1940.

Can’t use math!

If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?

Can a program refer to itself?

Uh oh....
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.
Is the statement above true?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.
Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.
This statement is a lie.
Is the statement above true?
The barber shaves every person who does not shave themselves.
Who shaves the barber?
Self reference.
Can a program refer to a program?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?
Can a program refer to itself?
Resolution of hypothesis?

Gödel. 1940.
Can’t use math!
If math doesn’t contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?
Can a program refer to itself?

Uh oh....
Changing Axioms?

Goedel:
Any set of axioms is either

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine...
..but for ...
two schizophrenic children..

Dangerous work?
See Logicomix by Doxiaidis, Papadimitriou (was professor here),
Papadatos, Di Donna.
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:

BTW:
Cantor..bipolar disorder..
Goedel..starved himself out of fear of being poisoned..
Russell.. was fine...
..but for ...
two schizophrenic children..

Dangerous work?
See Logicomix by Doxiaidis, Papadimitriou (was professor here), Papadatos, Di Donna.
Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatility between reals and naturals.”
Continuum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.” Continuum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable. (Cohen 1963: only Fields medal in logic)

BTW:
Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Continuum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Changing Axioms?

Goedel:
Any set of axioms is either 
inconsistent (can prove false statements) or 
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel..
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinality between reals and naturals.”
Continuum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.” Continuum hypothesis not disprovable in ZFC (Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell
Changing Axioms?

Goedel:  
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:  
Continuum hypothesis: “no cardinality between reals and naturals.”  
Continuum hypothesis not disprovable in ZFC  
(Goedel 1940.)

Continuum hypothesis not provable.  
(Cohen 1963: only Fields medal in logic)

BTW:  
Cantor ..bipolar disorder..  
Goedel ..starved himself out of fear of being poisoned..  
Russell .. was fine...
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinativity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatility between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatility between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?
Changing Axioms?

Goedel:
Any set of axioms is either inconsistent (can prove false statements) or incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.” Continentum hypothesis not disprovable in ZFC (Goedel 1940.)
Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?

See Logicomix by Doxiaidis, Papadimitriou (was professor here), Papadatos, Di Donna.
Is it actually useful?

Write me a program checker!

Check that the compiler works!

Check that the compiler terminates on a certain input.

\[
\text{HALT}(P, I)
\]

\(P\) - program
\(I\) - input.

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

Notice:
Need a computer...
with the notion of a stored program!!!!

(not an adding machine!
not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.
Is it actually useful?

Write me a program checker!
Check that the compiler works!
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

\( \text{HALT}(P, I) \)

\[ P - \text{program} \]
Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ HALT(P, I) \]
\[
P - \text{program}
\]
\[
I - \text{input.}
\]
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ HALT(P, I) \]
- \( P \) - program
- \( I \) - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice: Need a computer...with the notion of a stored program!!!! (not an adding machine! not a person and an adding machine.)
Program is a text string. Text string can be an input to a program. Program can be an input to a program.
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P,I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P, I) \]

\(P\) - program

\(I\) - input.

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

Notice:

Need a computer
Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

\textit{HALT}(P, I)

- \(P\) - program
- \(I\) - input.

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

Notice:

Need a computer

...with the notion of a stored program!!!!
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\textit{HALT}(\textit{P}, \textit{I})
\begin{align*}
\textit{P} & - \text{program} \\
\textit{I} & - \text{input}.
\end{align*}

Determines if \textit{P(I)} (\textit{P} run on \textit{I}) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine!)
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[
HALT(P, I)
\]

\[
P - \text{program}
\]

\[
I - \text{input.}
\]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input}. \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P, I) \]
\[ P \] - program
\[ I \] - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Is it actually useful?

Write me a program checker!
Check that the compiler works!
How about.. Check that the compiler terminates on a certain input.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.
Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$

$P$ - program

$I$ - input.

Determines if $P(I)$ ($P$ run on $I$) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.

**Text string can be an input to a program.**

**Program can be an input to a program.**
Implementing HALT.

HALT($P, I$) - program $P$ run on input $I$, determines if it halts or loops forever. Run $P$ on $I$ and check! How long do you wait? Something about infinity here, maybe?
Implementing HALT.

$HALT(P, I)$
Implementing HALT.

$HALT(P, I)$

$P$ - program
Implementing HALT.

\[ \text{HALT}(P, I) \]
- \( P \) - program
- \( I \) - input.
Implementing HALT.

$HALT(P, I)$

$P$ - program
$I$ - input.

Determines if $P(I)$ ($P$ run on $I$) halts or loops forever.
Implementing HALT.

HALT\((P, I)\)
\begin{align*}
P & - \text{program} \\
I & - \text{input.}
\end{align*}

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

Run \(P\) on \(I\) and check!
Implementing HALT.

**HALT**\((P, I)\)
- \(P\) - program
- \(I\) - input.

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

Run \(P\) on \(I\) and check!

How long do you wait?
Implementing HALT.

$HALT(P, I)$

$P$ - program

$I$ - input.

Determines if $P(I)$ ($P$ run on $I$) halts or loops forever.

Run $P$ on $I$ and check!

How long do you wait?

Something about infinity here, maybe?
Halt does not exist.
Halt does not exist.

\[ \text{HALT}(P, I) \]
Halt does not exist.

\[ \text{HALT}(P, I) \]

\( P \) - program
Halt does not exist.

\[ HALT(P, I) \]
- \( P \) - program
- \( I \) - input.
Halt does not exist.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.
Halt does not exist.

\[ \text{HALT}(P, I) \]

*P* - program

*I* - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.
Halt does not exist.

\[ \text{HALT}(P, I) \]
\[ P \text{ - program} \]
\[ I \text{ - input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes!
Halt does not exist.

**HALT**\((P, I)\)
- \(P\) - program
- \(I\) - input.

Determines if \(P(I)\) (\(P\) run on \(I\)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No!
Halt does not exist.

$HALT(P, I)$

$P$ - program
$I$ - input.

Determines if $P(I)$ ($P$ run on $I$) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes!
Halt does not exist.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes! No!
Halt does not exist.

\[ \text{HALT}(P, I) \]
\[ P \text{ - program} \]
\[ I \text{ - input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes! No! No!
Halt does not exist.

\[ HALT(P, I) \]
- \( P \) - program
- \( I \) - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes! No! No! Yes!
Halt does not exist.

\[ \text{HALT}(P, I) \]

\( P \) - program
\( I \) - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program \text{HALT}.

**Proof:** Yes! No! Yes! No! No! Yes! No!
Halt does not exist.

\[ \text{HALT}(P, I) \]
- \( P \) - program
- \( I \) - input.

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes! No! No! Yes! No! Yes!
Halt does not exist.

\[ \textit{HALT}(P, I) \]

\begin{itemize}
  \item \textit{P} - program
  \item \textit{I} - input.
\end{itemize}

Determines if \( P(I) \) (\textit{P} run on \textit{I}) halts or loops forever.

\textbf{Theorem:} There is no program \textit{HALT}.

\textbf{Proof:} Yes! No! Yes! No! No! Yes! No! Yes! ..
Halt does not exist.

$HALT(P, I)$

$P$ - program
$I$ - input.

Determines if $P(I)$ ($P$ run on $I$) halts or loops forever.

**Theorem:** There is no program $HALT$.

**Proof:** Yes! No! Yes! No! No! Yes! No! Yes! ..
Halt does not exist.

\[ \text{HALT}(P, I) \]
\[ P - \text{program} \]
\[ I - \text{input.} \]

Determines if \( P(I) \) (\( P \) run on \( I \)) halts or loops forever.

**Theorem:** There is no program HALT.

**Proof:** Yes! No! Yes! No! No! Yes! No! Yes! ..
Yes! No!...

What is he talking about?
Yes! No!...

What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.
Yes! No!...

What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.

(B)
What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.

(B) and (D)
What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.

(B) and (D) maybe?
What is he talking about?
(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.

(B) and (D) maybe? and maybe (A).
What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp
(D) Professor is just strange.

(B) and (D) maybe? and maybe (A).

Professor does me some love Welch-Berlekamp though!
Halt and Turing.

Proof:

Assume there is a program \( \text{HALT}(\cdot, \cdot) \).

1. If \( \text{HALT}(\text{Turing}, \text{Turing}) \) = "halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program \( \text{HALT} \).

There is text that "is" the program \( \text{Turing} \).

Can run \( \text{Turing} \) on \( \text{Turing} \)!

Does \( \text{Turing}(\text{Turing}) \) halt?

\[ \text{Turing}(\text{Turing}) \text{ halts} = \Rightarrow \text{HALTS}(\text{Turing}, \text{Turing}) = \text{halts} = \Rightarrow \text{Turing}(\text{Turing}) \text{ loops forever} = \Rightarrow \text{HALTS}(\text{Turing}, \text{Turing}) \neq \text{halts} = \Rightarrow \text{Turing}(\text{Turing}) \text{ halts}. \]

Contradiction.

Program \( \text{HALT} \) does not exist!
Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$. 
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$. 

$Turing(P)$
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

Turing($P$)
1. If $HALT(P,P) =$“halts”, then go into an infinite loop.
Proof: Assume there is a program \( HALT(P,P) \).

\textbf{Turing(P)}
1. If \( HALT(P,P) = \text{"halts"} \), then go into an infinite loop.
2. Otherwise, halt immediately.
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

$\text{Turing}(P)$
1. If $HALT(P,P) =$“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$.

Contradiction.

Program $HALT$ does not exist!
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

**Turing(P)**
1. If $HALT(P, P) = \text{“halts”}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$.
There is text that “is” the program $HALT$. 
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

**Turing**($P$)
1. If $HALT(P,P) =$ “halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$. 
There is text that “is” the program $HALT$. 
There is text that is the program Turing.
**Halt and Turing.**

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

$\text{Turing}(P)$
1. If $HALT(P, P) = \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$.
There is text that “is” the program $HALT$.
There is text that is the program $Turing$.
Can run $Turing$ on $Turing$!
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

**Turing**(P)
1. If $HALT(P,P) =$ “halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does **Turing**(Turing) halt?
Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing($P$)
1. If $HALT(P,P) = \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$. There is text that “is” the program $HALT$. There is text that is the program $Turing$. Can run $Turing$ on $Turing$!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

$Turing(P)$

1. If $HALT(P, P) = \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$. There is text that “is” the program $HALT$. There is text that is the program $Turing$. Can run $Turing$ on $Turing$!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts

$\implies$ then $HALTS(Turing, Turing) = \text{halts}$
**Halt and Turing.**

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

$Turing(P)$
1. If $HALT(P,P) =$“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT. There is text that “is” the program HALT. There is text that is the program Turing. Can run Turing on Turing!

Does **Turing(Turing)** halt?

$Turing(Turing)$ halts

$\implies$ then $HALTS(Turing, Turing) = halts$  
$\implies$ Turing(Turing) loops forever.
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

**Turing($P$)**
1. If $HALT(P, P) = \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$.  
There is text that “is” the program $HALT$.  
There is text that is the program $Turing$.  
Can run $Turing$ on $Turing$!

Does $Turing(Turing)$ halt?

- $Turing(Turing)$ halts  
  $\implies$ then $HALTS(Turing, Turing) = \text{halts}$  
  $\implies$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

$Turing(P)$
1. If $HALT(P,P) \Rightarrow \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$.
There is text that “is” the program $HALT$.
There is text that is the program $Turing$.
Can run $Turing$ on $Turing$!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts

$\Rightarrow$ then $HALTS(Turing, Turing) \Rightarrow \text{halts}$

$\Rightarrow$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever

$\Rightarrow$ then $HALTS(Turing, Turing) \neq \text{halts}$
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

$Turing(P)$
1. If $HALT(P, P) =$ "halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT. There is text that “is” the program HALT. There is text that is the program Turing. Can run Turing on Turing!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts
$\implies$ then $HALTS(Turing, Turing) =$ halts
$\implies$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever
$\implies$ then $HALTS(Turing, Turing) \neq$ halts
$\implies$ $Turing(Turing)$ halts.
Halt and Turing.

**Proof:** Assume there is a program $HALT(·, ·)$.

$Turing(P)$
1. If $HALT(P, P) =$ “halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT. There is text that “is” the program HALT. There is text that is the program Turing. Can run Turing on Turing!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts
$\implies$ then $HALTS(Turing, Turing) = halts$
$\implies$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever
$\implies$ then $HALTS(Turing, Turing) \neq halts$
$\implies$ $Turing(Turing)$ halts.

Contradiction.
Halt and Turing.

**Proof:** Assume there is a program \( HALT(\cdot, \cdot) \).

**Turing(P)**
1. If \( HALT(P, P) = \text{“halts”} \), then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT. 
There is text that “is” the program HALT. 
There is text that is the program Turing. 
Can run Turing on Turing!

Does **Turing(Turing)** halt?

**Turing(Turing) halts**
\[ \implies \text{ then } HALTS(\text{Turing}, \text{Turing}) = \text{halts} \]
\[ \implies \text{Turing(Turing) loops forever.} \]

**Turing(Turing) loops forever**
\[ \implies \text{ then } HALTS(\text{Turing}, \text{Turing}) \neq \text{halts} \]
\[ \implies \text{Turing(Turing) halts.} \]

Contradiction. Program HALT does not exist!
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot, \cdot)$.

$Turing(P)$
1. If $HALT(P, P) =$"halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts
   $\implies$ then $HALTS(Turing, Turing) = \text{halts}$
   $\implies$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever
   $\implies$ then $HALTS(Turing, Turing) \neq \text{halts}$
   $\implies$ $Turing(Turing)$ halts.

Contradiction. Program HALT does not exist!
Halt and Turing.

**Proof:** Assume there is a program $HALT(\cdot,\cdot)$.

$Turing(P)$
1. If $HALT(P,P) = \text{"halts"}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program $HALT$. There is text that “is” the program $HALT$. There is text that is the program $Turing$. Can run $Turing$ on $Turing$!

Does $Turing(Turing)$ halt?

$Turing(Turing)$ halts
$\implies$ then $HALTS(Turing, Turing) = \text{halts}$
$\implies$ $Turing(Turing)$ loops forever.

$Turing(Turing)$ loops forever
$\implies$ then $HALTS(Turing, Turing) \neq \text{halts}$
$\implies$ $Turing(Turing)$ halts.

Contradiction. Program $HALT$ does not exist!

Questions?
Another view of proof: diagonalization.

Any program is a fixed length string.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal. Turing - is not Halt.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal.
Turing - is not Halt.
and is different from every $P_i$ on the diagonal.
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal. Turing - is not Halt. and is different from every $P_i$ on the diagonal. Turing is not on list.
Another view of proof: diagonalization.

Any program is a fixed length string.  
Fixed length strings are enumerable.  
Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal.  
Turing - is **not** Halt.  
and is different from every $P_i$ on the diagonal.  
Turing is not on list.  Turing is not a program.
Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal.
Turing - is not Halt.
and is different from every $P_i$ on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Another view of proof: diagonalization.

Any program is a fixed length string.  
Fixed length strings are enumerable.  
Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
</tbody>
</table>
| ... | ... | ... | ... | ...

Halt - diagonal.  
Turing - is not Halt.  
and is different from every $P_i$ on the diagonal.  
Turing is not on list. Turing is not a program.  
Turing can be constructed from Halt.  
Halt does not exist!
Another view of proof: diagonalization.

Any program is a fixed length string. Fixed length strings are enumerable. Program halts or not on any input, which is a string.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>...</td>
</tr>
<tr>
<td>$P_2$</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>$P_3$</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Halt - diagonal. Turing - is not Halt. and is different from every $P_i$ on the diagonal. Turing is not on list. Turing is not a program. Turing can be constructed from Halt. Halt does not exist!
What are programs?
What are programs?

(A) Instructions.
(B) Text.
(C) Binary String.
(D) They run on computers.
What are programs?

(A) Instructions.
(B) Text.
(C) Binary String.
(D) They run on computers.

All are correct.
Proof play by play.

Assumed HALT($P, I$) existed.

You have that is the program HALT($P, I$).

Have that is the program TURING. Here it is!!

Turing($P$)
1. If HALT($P, P$) = "halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Turing "diagonalizes" on list of programs.
It is not a program!!

$\Rightarrow$ HALT is not a program.

Questions?
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$?
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.

What is $I$?
Assumed HALT$(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.
Proof play by play.

Assumed HALT($P, I$) existed.

What is $P$? Text.
What is $I$? Text.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.
Proof play by play.

Assumed $\text{HALT}(P,I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P,I)$.
You have *Text* that is the program $\text{HALT}(P,I)$.
Proof play by play.

Assumed \( \text{HALT}(P, I) \) existed.

What is \( P \)? Text.
What is \( I \)? Text.

What does it mean to have a program \( \text{HALT}(P, I) \).
   You have *Text* that is the program \( \text{HALT}(P, I) \).
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have Text that is the program $\text{HALT}(P, I)$.

Have ____ that is the program TURING.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have *Text* that is the program $\text{HALT}(P, I)$.

Have *Text* that is the program TURING.
Proof play by play.

Assumed \( \text{HALT}(P, I) \) existed.

What is \( P \)? Text.
What is \( I \)? Text.

What does it mean to have a program \( \text{HALT}(P, I) \).

You have Text that is the program \( \text{HALT}(P, I) \).

Have Text that is the program TURING.
Here it is!!
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have Text that is the program $\text{HALT}(P, I)$.

Have Text that is the program Turing. Here it is!!

$\text{Turing}(P)$
Proof play by play.

Assumed \( \text{HALT}(P, I) \) existed.

What is \( P \)? Text.
What is \( I \)? Text.

What does it mean to have a program \( \text{HALT}(P, I) \).
You have Text that is the program \( \text{HALT}(P, I) \).

Have Text that is the program TURING.
Here it is!!

\( \text{Turing}(P) \)
  1. If \( \text{HALT}(P,P) = \text{"halts"} \), then go into an infinite loop.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have Text that is the program $\text{HALT}(P, I)$.

Have Text that is the program TURING.
Here it is!!

$\text{Turing}(P)$

1. If $\text{HALT}(P,P) =$“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have Text that is the program $\text{HALT}(P, I)$.

Have Text that is the program TURING.
Here it is!!

$\text{Turing}(P)$
1. If $\text{HALT}(P,P) = \text{“halts”}$, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
Proof play by play.

Assumed $\text{HALT}(P,I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P,I)$.

You have Text that is the program $\text{HALT}(P,I)$.

Have Text that is the program TURING.
Here it is!!

Turing($P$)
1. If $\text{HALT}(P,P) =$"halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have $Text$ that is the program $\text{HALT}(P, I)$.

Have $Text$ that is the program TURING. Here it is!!

$\text{Turing}(P)$

1. If $\text{HALT}(P, P) =$“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
It is not a program!!!!
Proof play by play.

Assumed \( \text{HALT}(P, I) \) existed.

What is \( P \)? Text.
What is \( I \)? Text.

What does it mean to have a program \( \text{HALT}(P, I) \).

You have Text that is the program \( \text{HALT}(P, I) \).

Have Text that is the program \( \text{TURING} \).
Here it is!!

\text{Turing}(P)

1. If \( \text{HALT}(P, P) = \text{"halts"} \), then go into an infinite loop.
2. Otherwise, halt immediately.

\( \text{Turing} \) “diagonalizes” on list of program.
It is not a program!!!!
\( \implies \) \( \text{HALT} \) is not a program.
Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is $P$? Text.
What is $I$? Text.

What does it mean to have a program $\text{HALT}(P, I)$.

You have Text that is the program $\text{HALT}(P, I)$.

Have Text that is the program TURING.
Here it is!!

\textbf{Turing}(P)

1. If $\text{HALT}(P, P) =$“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
It is not a program!!!!

$\implies$ $\text{HALT}$ is not a program.

Questions?
We are so smart!

Wow, that was easy!
We are so smart!

Wow, that was easy!
We should be famous!
No computers for Turing!

In Turing’s time.
No computers for Turing!

In Turing’s time.
No computers.
No computers for Turing!

In Turing's time.
No computers.
Adding machines.
No computers for Turing!

In Turing’s time.
No computers.
Adding machines.
e.g., Babbage (from table of logarithms) 1812.
No computers for Turing!

In Turing’s time.
No computers.
Adding machines.
e.g., Babbage (from table of logarithms) 1812.
Concept of program as data wasn’t really there.
Turing machine.

A Turing machine – an (infinite) tape with characters – be in a state, and read a character – move left, right, and/or write a character.

Universal Turing machine – an interpreter program for a Turing machine – where the tape could be a description of a ...

Now that's a computer!

Turing: AI, self modifying code, learning...
A Turing machine.
– an (infinite) tape with characters
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
A Turing machine.
- an (infinite) tape with characters
- be in a state, and read a character
- move left, right, and/or write a character.

Universal Turing machine
- an interpreter program for a Turing machine
A Turing machine.

– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a...
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that’s a computer!
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI,
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that’s a computer!

Turing: AI, self modifying code,
A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI, self modifying code, learning...
Turing and computing.

Just a mathematician?
Turing and computing.

Just a mathematician?

“Wrote” a chess program.
Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.
Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won!
Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.
Turing and computing.

Just a mathematician?
“Wrote” a chess program.
Simulated the program by hand to play chess.
It won! Once anyway.
Involved with computing labs through the 40s.
Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Involved with computing labs through the 40s.

Helped Break the enigma code.
Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Involved with computing labs through the 40s.

Helped Break the enigma code.

The polish machine...the *bomba*. 
Computing on top of computing...

Computer, assembly code, programming language, browser, html, javascript..
Computing on top of computing...

Computer, assembly code, programming language, browser, html, javascript..
We can’t get enough of building more Turing machines.
Undecidable problems.

Does a program, $P$, print “Hello World”? 

Does a set of notched tiles tile the infinite plane? 

Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?

Example: $x^n + y^n = 1$?

Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?

(Diophantine equation.)

The answer is yes or no.

This “problem” is not undecidable.

Undecidability for Diophantine set of equations $\Rightarrow$ no program can take any set of integer equations and always correctly output whether it has an integer solution.
Undecidable problems.

Does a program, $P$, print “Hello World”? How?
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$?
Undecidable problems.

Does a program, \( P \), print “Hello World”? How? What is \( P \)? Text!!!!!!
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!
Undecidable problems.

Does a program, \( P \), print “Hello World”? How? What is \( P \)? Text!!!!!!

Find exit points and add statement: Print “Hello World.”
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: \textbf{Print} “Hello World.”
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: Print “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: Print “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?”
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: Print “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.

Be careful!
Undecidable problems.

Does a program, \( P \), print “Hello World”? How? What is \( P \)?

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “\( x^n + y^n = 1 \)?” Problem is undecidable.

Be careful!
Undecidable problems.

Does a program, $P$, print “Hello World”?  
How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?  
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?  
Example: “$x^n + y^n = 1$?”  
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$? (Diophantine equation.)
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$? (Diophantine equation.)

The answer is yes or no.
Undecidable problems.

Does a program, $P$, print "Hello World"?
How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: "$x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$? (Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations
Undecidable problems.

Does a program, \( P \), print “Hello World”? How? What is \( P \)? Text!!!!!!

Find exit points and add statement: \textbf{Print} “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “\( x^n + y^n = 1? \)” Problem is undecidable.

Be careful!

Is there an integer solution to \( x^n + y^n = 1? \) (Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations
\[ \implies \] no program can take any set of integer equations and
Undecidable problems.

Does a program, $P$, print “Hello World”? How? What is $P$? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane? Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution? Example: “$x^n + y^n = 1$?” Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$? (Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations $\implies$ no program can take any set of integer equations and always correctly output whether it has an integer solution.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis:
- Seminal paper in mathematical biology.
- Person: embryo is blob. Legs, arms, head.... How?
- Imitation Game.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work.
  Almost dependent matrices.

- Seminal paper in mathematical biology. Person: embryo is blob. Legs, arms, head....
  How?
  Fly: blob. Torso becomes striped. Developed chemical reaction-diffusion networks that break symmetry.

- Imitation Game.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person:
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person: embryo is blob.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person: embryo is blob. Legs,
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person: embryo is blob. Legs, arms,
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).

- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.

- Seminal paper in mathematical biology. Person: embryo is blob. Legs, arms, head....
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person: embryo is blob. Legs, arms, head.... How?
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
- Seminal paper in mathematical biology. Person: embryo is blob. Legs, arms, head.... How? Fly:
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.
More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).

- Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.

- Seminal paper in mathematical biology.
  - Person: embryo is blob. Legs, arms, head.... How?
More about Alan Turing.

► Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).

► Seminal paper in numerical analysis: Condition number. Math 54 doesn’t really work. Almost dependent matrices.


► Imitation Game.
Turing: personal.

Tragic ending...
Turing: personal.

Tragic ending...

▶ Arrested as a homosexual, (not particularly closeted)
Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- Given choice of prison or (quackish) injections to eliminate sex drive;
- Lost security clearance...
- Suffered from depression;
- (Possibly) suicided with cyanide at age 42 in 1954.

(A bite from the apple....) Accident?

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- Given choice of prison or (quackish) injections to eliminate sex drive;
- Took injections.

(A bite from the apple....) accident?

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954.
Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954. (A bite from the apple....)
Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954. (A bite from the apple....) accident?
Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954. (A bite from the apple....) accident?
This statement is a lie.
This statement is a lie. *Neither true nor false!*

[...Technical content...]
Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?
This statement is a lie. **Neither true nor false!**

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):

This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return
This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

*Who shaves the barber?*

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program.
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

```python
def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. ($P \implies Q$)
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

*Who shaves the barber?*

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. $(P \implies Q)$
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. $(P \implies Q)$

Turing(“Turing”)?
This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. ($P \implies Q$)

Turing(“Turing”)? Neither halts nor loops!
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

\[ Halt \text{ Program} \implies Turing \text{ Program. } (P \implies Q) \]

Turing(“Turing”)? Neither halts nor loops! \[\implies\] No Turing program.
This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. ($P \implies Q$)

Turing(“Turing”)? Neither halts nor loops! $\implies$ No Turing program.
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

*Who shaves the barber?*

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. $(P \implies Q)$

Turing(“Turing”)? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

*Who shaves the barber?*

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\Rightarrow$ Turing Program. $(P \Rightarrow Q)$

Turing(“Turing”)? Neither halts nor loops! $\Rightarrow$ No Turing program.

No Turing Program $\Rightarrow$
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

```python
def Turing(P):
    if Halts(P, P): while(true): pass
    else:
        return
...Text of Halt...
```

*Halt Program $\Rightarrow$ Turing Program. $(P \Rightarrow Q)$*

*Turing(“Turing”)? Neither halts nor loops! $\Rightarrow$ No Turing program.*

*No Turing Program $\Rightarrow$ No halt program.*
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

**Who shaves the barber?**

```python
def Turing(P):
    if Halts(P, P):
        while (true):
            pass
    else:
        return

...Text of Halt...
```

Halt Program $\implies$ Turing Program. ($P \implies Q$)

Turing(“Turing”)? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)
This statement is a lie. *Neither true nor false!*

Every person who doesn’t shave themselves is shaved by the barber.

*Who shaves the barber?*

```python
def Turing(P):
    if Halts(P, P): while(true): pass
    else:
        return

...Text of Halt...
```

Halt Program $\implies$ Turing Program. ($P \implies Q$)

Turing(“Turing”)? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)

Program is text, so we can pass it to itself,
This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.  
Who shaves the barber?

def Turing(P):
    if Halts(P,P): while(true): pass
    else:
        return

...Text of Halt...

Halt Program $\implies$ Turing Program. ($P \implies Q$)

Turing(“Turing”)? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)

Program is text, so we can pass it to itself, or refer to self.
Computer Programs are an interesting thing.
Computer Programs are an interesting thing. Like Math.
Summary: decidability.

Computer Programs are an interesting thing.
Like Math.
Formal Systems.
Computer Programs are an interesting thing.
Like Math.
Formal Systems.
Summary: decidability.

Computer Programs are an interesting thing.  
    Like Math.  
    Formal Systems.  

Computer Programs cannot completely “understand” computer programs.
Summary: decidability.

Computer Programs are an interesting thing. Like Math. Formal Systems.

Computer Programs cannot completely “understand” computer programs.
Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Computation is a lens for other action in the world.
Kolmogorov Complexity, Google, and CS70

Of strings, $s$. 

What Kolmogorov complexity of a string of 1,000,000 one's?

What is Kolmogorov complexity of a string of $n$ one's?

for $i = 1$ to $n$: print '1'. 

Kolmogorov Complexity, Google, and CS70

Of strings, \( s \).

Minimum sized program that prints string \( s \).
Of strings, $s$.

Minimum sized program that prints string $s$.

What Kolmogorov complexity of a string of 1,000,000, one's?
Of strings, $s$.

Minimum sized program that prints string $s$.

What Kolmogorov complexity of a string of 1,000,000, one’s?

What is Kolmogorov complexity of a string of $n$ one’s?
Of strings, \( s \).

Minimum sized program that prints string \( s \).

What Kolmogorov complexity of a string of 1,000,000, one’s?

What is Kolmogorov complexity of a string of \( n \) one’s?

for \( i = 1 \) to \( n \): print ’1’. 
What is the minimum I need to know (remember) to know stuff.
What is the minimum I need to know (remember) to know stuff.

Radius of the earth?
Kolmogorov Complexity, Google, and CS70

What is the minimum I need to know (remember) to know stuff. Radius of the earth? Distance to the sun?
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US?
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?
What is the minimum I need to know (remember) to know stuff.

What is the minimum I need to know (remember) to know stuff.

What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?


Syntax of pandas?
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow.
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow. Plus “how to program”
Kolmogorov Complexity, Google, and CS70

What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow. Plus “how to program” and remembering a bit.
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US?
Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow.
Plus “how to program” and remembering a bit.

Calculus:
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow. Plus “how to program” and remembering a bit.

Calculus: what is minimum you need to know?
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US?
Accelerate due to gravity on earth?

Syntax of pandas? Google + Stackoverflow.
Plus “how to program” and remembering a bit.

Calculus: what is minimum you need to know?
Depends on your skills!
What is the minimum I need to know (remember) to know stuff.

Radius of the earth? Distance to the sun? Population of the US? Acceleration due to gravity on earth?

Syntax of pandas? Google + Stackoverflow. Plus “how to program” and remembering a bit.

Calculus: what is minimum you need to know?
Depends on your skills!
Reason and understand an argument and you can generate a lot.
What is the first half of calculus about?

Calculus
What is the first half of calculus about?

The slope of a tangent line to a function at a point.
What is the first half of calculus about?
The slope of a tangent line to a function at a point.
Slope is rise/run.
Calculus

What is the first half of calculus about?
The slope of a tangent line to a function at a point.
Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).
Calculus

What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).

Chain rule?
Calculus

What is the first half of calculus about?
The slope of a tangent line to a function at a point.
Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).
Chain rule? Derivative of a function composition.
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition. Intuition: composition of two linear functions?
Calculus

What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition. Intuition: composition of two linear functions?

\[ f(x) = ax, \quad g(x) = bx. \]
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?
\( f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \)
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\( f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \) Slope is \( ab \).

Multiply slopes!
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \; g(x) = bx. \; f(g(x)) = ab \; x. \] Slope is \( ab \).

Multiply slopes!
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \] Slope is \( ab \).

Multiply slopes!

\[ (f(g(x)))' = f'(\cdot)g'(\cdot) \]
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition. Intuition: composition of two linear functions?

\[ f(x) = ax, \quad g(x) = bx. \quad f(g(x)) = ab \, x. \quad \text{Slope is } ab. \]

Multiply slopes!

\[ (f(g(x)))' = f'(\cdot)g'(\cdot) \]

But...but...
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \quad g(x) = bx. \quad f(g(x)) = ab \cdot x. \quad \text{Slope is } ab. \]

Multiply slopes!

\[ (f(g(x)))' = f'(\cdot)g'(\cdot) \]

But...but...
Calculus

What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\( f(x) = ax, g(x) = bx \). \( f(g(x)) = ab \, x \). Slope is \( ab \).

Multiply slopes!

\((f(g(x)))' = f'(\cdot)g'(\cdot)\)

But...but...

For function slopes of tangent differ at different places.
Calculus

What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition. Intuition: composition of two linear functions?

\( f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \) Slope is \( ab \). Multiply slopes!

\( (f(g(x)))' = f'(\cdot)g'(\cdot) \)

But...but...

For function slopes of tangent differ at different places.
Calculus

What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \[ \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \].

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \] Slope is \( ab \).

Multiply slopes!

\[ (f(g(x)))' = f'(\cdot)g'(\cdot) \]

But...but...

For function slopes of tangent differ at different places.

So, where?
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\( f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \cdot x. \) Slope is \( ab. \)

Multiply slopes!

\( (f(g(x)))' = f'()g'() \)

But...but...

For function slopes of tangent differ at different places.

So, where? \( f(g(x)) \)
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?
\[ f(x) = ax, \quad g(x) = bx. \quad f(g(x)) = ab \cdot x. \] Slope is \( ab \).

Multiply slopes!
\[ (f(g(x)))' = f'(\cdot)g'(\cdot) \]

But...but...

For function slopes of tangent differ at different places.

So, where? \( f(g(x)) \)

slope of \( f \) at \( g(x) \) times slope of \( g \) at \( x \).
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \quad g(x) = bx. \quad f(g(x)) = abx. \quad \text{Slope is } ab. \]

Multiply slopes!

\( (f(g(x)))' = f'(\cdot)g' (\cdot) \)

But...but...

For function slopes of tangent differ at different places.

So, where? \( f(g(x)) \)

slope of \( f \) at \( g(x) \) times slope of \( g \) at \( x \).

\( (f(g(x)))' = f'(g(x))g'(x) \).
What is the first half of calculus about?

The slope of a tangent line to a function at a point.

Slope is rise/run. Oh, yes: \( \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \).

Chain rule? Derivative of a function composition.

Intuition: composition of two linear functions?

\[ f(x) = ax, \ g(x) = bx. \ f(g(x)) = ab \ x. \text{ Slope is } ab. \]

Multiply slopes!

\[
(f(g(x)))' = f'(\cdot)g'(\cdot)
\]

But...but...

For function slopes of tangent differ at different places.

So, where? \( f(g(x)) \)

slope of \( f \) at \( g(x) \) times slope of \( g \) at \( x \).

\[
(f(g(x)))' = f'(g(x))g'(x).
\]
Product Rule.

Idea: use rise in function value!
Product Rule.

Idea: use rise in function value!
\[ d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu. \]
Product Rule.

Idea: use rise in function value!
\[ d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu. \]

Any concept:
Idea: use rise in function value!
\[ d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu. \]

Any concept:
A quick argument from basic concept of slope of a tangent line.
Product Rule.

Idea: use rise in function value!
\[ d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu. \]

Any concept:
A quick argument from basic concept of slope of a tangent line.

Perhaps.
Derivative of sine?

\[ \sin(x). \]
Derivative of sine?

\[ \sin(x) \].

What is \( x \)? An angle in radians.
Derivative of sine?

\[ \sin(x) \].

What is \( x \)? An angle in radians.
Let’s call it \( \theta \) and do derivative of \( \sin \theta \).
Derivative of sine?

\( \sin(x) \).

What is \( x \)? An angle in radians.
Let’s call it \( \theta \) and do derivative of \( \sin \theta \).

\( \theta \) - Length of arc of unit circle
Derivative of sine?

\[ \sin(x) \].

What is \( x \)? An angle in radians.

Let’s call it \( \theta \) and do derivative of \( \sin \theta \).

\( \theta \) - Length of arc of unit circle
Derivative of sine?

\[ \sin(x) \].

What is \( x \)? An angle in radians.

Let’s call it \( \theta \) and do derivative of \( \sin \theta \).

\( \theta \) - Length of arc of unit circle
Derivative of sine?

\[ \sin(x) \].

What is \( x \)? An angle in radians.
Let’s call it \( \theta \) and do derivative of \( \sin \theta \).

\( \theta \) - Length of arc of unit circle

Rise. Similar triangle!!!
Arguments, reasoning.

What you know: slope, limit.
Arguments, reasoning.

What you know: slope, limit.
    Plus: definition.
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, ….
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, ….

Knowing how to program
Arguments, reasoning.

What you know: slope, limit.
  Plus: definition.
yields calculus.
  Minimization, optimization, ….

Knowing how to program plus some syntax (google) gives the ability to program.
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, . . . .

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, ….

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, ….

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition gives calculus.
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, ….

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition gives calculus.

Discrete Math: basics are counting, how many, when are two sets the same size?
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, . . . .

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition gives calculus.

Discrete Math: basics are counting, how many, when are two sets the same size?

Probability:
Arguments, reasoning.

What you know: slope, limit.
   Plus: definition.
yields calculus.
   Minimization, optimization, .....

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition gives calculus.

Discrete Math: basics are counting, how many, when are two sets the same size?

Probability: division.
Arguments, reasoning.

What you know: slope, limit.
  Plus: definition.
yields calculus.
  Minimization, optimization, ….

Knowing how to program plus some syntax (google) gives the ability to program.

Knowing how to reason plus some definition gives calculus.

Discrete Math: basics are counting, how many, when are two sets the same size?

Probability: division.

...plus reasoning.
CS 70: ideas.

Induction

Induction
CS 70: ideas.

Induction $\equiv$ every integer has a next one.
Induction $\equiv$ every integer has a next one. Graph theory.
Number of edges is sum of degrees.
$\Delta + 1$ coloring. Neighbors only take up $\Delta$.
Connectivity plus connected components.
Eulerian paths: if you enter you can leave.
Euler’s formula: tree has $v - 1$ edges and 1 face plus each extra edge makes additional face.
$v - 1 + (f - 1) = e$
Number theory.
A divisor of $x$ and $y$ divides $x - y$.
The remainder is always smaller than the divisor.
$\implies$ Euclid’s GCD algorithm.
Multiplicative Inverse.
Fermat’s theorem from function with inverse is a bijection.
Gives RSA.
Number theory.
A divisor of $x$ and $y$ divides $x - y$.
The remainder is always smaller than the divisor.
⇒ Euclid’s GCD algorithm.
Multiplicative Inverse.
Fermat’s theorem from function with inverse is a bijection.
Gives RSA.

Error Correction.
(Any) Two points determine a line.
(well, and $d$ points determine a degree $d + 1$-polynomials.
Cuz, factoring.
Find line by linear equations.
If a couple are wrong, then multiply them by zero, i.e., Error polynomial.
CS70 and your future?

What’s going on?
CS70 and your future?

What’s going on?
Define. Understand properties. And build from there.
CS70 and your future?

What’s going on?
Define. Understand properties. And build from there.
Tools: reasoning, proofs, care.
What’s going on?
Define. Understand properties. And build from there.
Tools: reasoning, proofs, care.
CS70 and your future?

What’s going on?
Define. Understand properties. And build from there.
Tools: reasoning, proofs, care.
Gives power to your creativity and in your pursuits.
What’s going on?
Define. Understand properties. And build from there.
Tools: reasoning, proofs, care.
Gives power to your creativity and in your pursuits.
....and you will pursue probability in this course.