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?

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.

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?

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....

Is it actually useful?

Write me a program checker!

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.

$HALT(P, I)$
 P - 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.

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.

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:

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

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!!!!

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!

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.)

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.)

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.

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.

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.

$HALT(P, I)$
  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.

Implementing HALT.

$HALT(P, I)$

Implementing HALT.

$HALT(P, I)$

P - program

Implementing HALT.

$HALT(P, I)$

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.

$HALT(P, I)$
  P - program
  I - input.

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.

$HALT(P, I)$

Halt does not exist.

$HALT(P, I)$

P - program

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

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

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.

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!

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.

$HALT(P, I)$
  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!

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!

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.

$HALT(P, I)$

  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!

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.

$HALT(P, I)$
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.

$HALT(P, I)$
  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.

$HALT(P, I)$
  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! .. $\square$

Yes! No!...

What is he talking about?

Yes! No!...

What is he talking about?

(A) He is confused.

Yes! No!...

What is he talking about?

(A) He is confused.
(B) Diagonalization.

Yes! No!...

What is he talking about?

(A) He is confused.
(B) Diagonalization.
(C) Welch-Berlekamp

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)

Yes! No!...

What is he talking about?

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

(B) and (D)

Yes! No!...

What is he talking about?

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

(B) and (D) maybe?

Yes! No!...

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).

Yes! No!...

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 love Welch-Berlekamp though!

Halt and Turing.

Proof:

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.

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.

Halt and Turing.

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

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

Assumption: there is a program HALT.

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.

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 $HALT(\cdot, \cdot)$.

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!

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) ="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) ="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

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) ="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

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

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 $HALT(\cdot, \cdot)$.

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.

Contradiction.

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.

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) $=$ halts
$\implies$ Turing(Turing) loops forever.

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

Contradiction. Program HALT does not exist! $\square$

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.

Contradiction. Program HALT does not exist! $\square$
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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

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! $\square$

Programs?

What are programs?

Programs?

What are programs?

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

Programs?

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.

Proof play by play.

Assumed HALT(P, I) existed.

What is P?

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I?

Proof play by play.

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 HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have ____ that is the program TURING.

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

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

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

1. If HALT(P,P) =“halts”, then go into an infinite loop.

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

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

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

1. If 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 HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

1. If 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 HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)
1. If 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 HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

1. If 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!!!!
$\implies$ HALT is not a program.

Proof play by play.

Assumed HALT(P, I) existed.

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

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
from fancystuff import halt

Turing(P)

1. If 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!!!!
$\Longrightarrow$ 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.

Turing machine.

A Turing machine.
– an (infinite) tape with characters

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character

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.

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

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

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.

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!

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!

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!

Turing: AI,

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!

Turing: AI, self modifying code,

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!

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.

Reductions from HALT.

Does a program, P, print “Hello World” on input x?

Reductions from HALT.

Does a program, P, print “Hello World” on input x?

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
  Does not exist.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
  Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
    Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
  Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
  Input: P.
  Make P' from P

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
  Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

 Does P, x halt?
  Input: P.
  Make P' from P
    How: rewrite text of P to form P'.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
  Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

  Does P, x halt?
  Input: P.
  Make P' from P
    How: rewrite text of P to form P'.
      programatically.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:
P' run on x prints "Hello, World"

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:
P' run on x prints "Hello, World"
if and only if P run on x halts.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
 Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
 Input: P.
 Make P' from P
  How: rewrite text of P to form P'.
   programatically.

So that:
 P' run on x prints "Hello, World"
 if and only if P run on x halts.

Thus, HALT is HelloWorld(P')

Reductions from HALT.

Does a program, P, print “Hello World” on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of “HALT” to “HELLO WORLD?”.

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:
P' run on x prints “Hello, World”
if and only if P run on x halts.

Thus, HALT is HelloWorld(P’) and $P' = \textit{Reduce}(P)$.

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:
P' run on x prints "Hello, World"
if and only if P run on x halts.

Thus, HALT is HelloWorld(P') and $P' = \mathit{Reduce}(P)$.

Reductions from HALT.

Does a program, P, print “Hello World” on input x?

HALT: Does a program, P, halt?
 Does not exist.

Reduction of “HALT” to “HELLO WORLD?”.

Does P, x halt?
 Input: P.
 Make P' from P
  How: rewrite text of P to form P'.
   programatically.

So that:
 P' run on x prints “Hello, World”
 if and only if P run on x halts.

Thus, HALT is HelloWorld(P’) and $P' = Reduce(P)$.

But HALT does not exist

Reductions from HALT.

Does a program, P, print "Hello World" on input x?

HALT: Does a program, P, halt?
Does not exist.

Reduction of "HALT" to "HELLO WORLD?".

Does P, x halt?
Input: P.
Make P' from P
How: rewrite text of P to form P'.
programatically.

So that:
P' run on x prints "Hello, World"
if and only if P run on x halts.

Thus, HALT is HelloWorld(P') and $P' = \mathit{Reduce}(P)$.

But HALT does not exist $\implies$ "HELLO, WORLD?" does not exist.

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How?

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P?

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** “Hello World.”

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
Remove all print statements.
Find exit points and add statement: **Print** "Hello World."

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** "Hello World."

Makes: P'

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** "Hello World."

Makes: P'

Claim: P halts if and only if P' prints "Hello World."

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** “Hello World.”

Makes: P'

Claim: P halts if and only if P' prints “Hello World.”

  Only if: Never prints anything unless it exits.

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** "Hello World."

Makes: P'

Claim: P halts if and only if P' prints "Hello World."

  Only if: Never prints anything unless it exits.
    cuz no other print statements.

Reduce HALT to HELLOWORLD?

Does P Halt on x?

Rewrite to P'.

How? What is P? Text!!!!!!

Rewrite:
  Remove all print statements.
  Find exit points and add statement: **Print** “Hello World.”

Makes: P'

Claim: P halts if and only if P' prints “Hello World.”

  Only if: Never prints anything unless it exits.
    cuz no other print statements.

  If: If P halts, P' prints “Hello World.”

Subroutine version of reduction.

Input to HALT: P.

Subroutine version of reduction.

Input to HALT: P.

Program Reduce takes P and writes this program.

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

def P'(x):

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

def P'(x):
  Redirect stdout ¿ null

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

```
def P'(x):
   Redirect stdout ¿ null
   P(x)
```

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

```
def P'(x):
   Redirect stdout ¿ null
   P(x)
   Restore stdout.
```

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

```
def P'(x):
   Redirect stdout ¿ null
   P(x)
   Restore stdout.
   print "Hello World".
```

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

```
def P'(x):
  Redirect stdout ¿ null
  P(x)
  Restore stdout.
  print "Hello World".
```

Subroutine version of reduction.

Input to HALT: *P*.

Program Reduce takes *P* and writes this program.

```
def P'(x):
  Redirect stdout ¿ null
  P(x)
  Restore stdout.
  print "Hello World".
```

$P'(x)$ prints "Hello World" if and only if $P(x)$ halts.

Undecidable problems.

Does a program, *P*, print “Hello World”?

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: **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*? 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? 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: **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
$\Longrightarrow$ no program can take any set of integer equations and always corectly output whether it has an integer 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: “ $x^n + y^n = 1$?”
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 corectly output whether it has an integer solution.

Commonality:

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 corectly output whether it has an integer solution.

Commonality: Computing.

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 corectly output whether it has an integer solution.

Commonality: Computing.
61c out of anything.

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:

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.

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.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: 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, arms, head.... How?
  Fly: blob. Torso becomes striped.

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.

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.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.
  Developed chemical reaction-diffusion networks that break symmetry.
- Imitation Game.

Commonality:

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.

Commonality: Computing.

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.

Commonality: Computing.
61c out of anything.

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.

Commonality: Computing.
61c out of anything.

Computing as a lense:

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.

Commonality: Computing.
61c out of anything.

Computing as a lense: Science, Quantum Computing, DNA, ...

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;

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.

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.

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....)

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?

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?
- British Government apologized (2009) and pardoned (2013).

Back to technical..

This statement is a lie.

Back to technical..

This statement is a lie. Neither true nor false!

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Back to technical..

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?

Back to technical..

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):
```

Back to technical..

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
```

Back to technical..

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...

Back to technical..

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 Progam $\Longrightarrow$ Turing Program.

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")?

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops!

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$

Back to technical..

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 Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program.

Back to technical..

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 Progam $\Longrightarrow$ Turing Program. ($P \Longrightarrow Q$)

Turing("Turing")? Neither halts nor loops! $\Longrightarrow$ No Turing program.

No Turing Program $\Longrightarrow$ No halt program. ($\neg Q \Longrightarrow \neg P$)

Back to technical..

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 Progam $\implies$ 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,

Back to technical..

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 Progam $\implies$ 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.

Summary: decidability.

Computer Programs are an interesting thing.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Summary: decidability.

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.

Kolmogorov Complexity, Google, and CS70

Of strings, s.

Minimum sized program that prints string s.

Kolmogorov Complexity, Google, and CS70

Of strings, s.

Minimum sized program that prints string s.

What Kolmogorov complexity of a string of 1,000,000, one's?

Kolmogorov Complexity, Google, and CS70

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?

Kolmogorov Complexity, Google, and CS70

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'.

Kolmogorov Complexity, Google, and CS70

What is the minimum I need to know (remember) to know stuff.

Kolmogorov Complexity, Google, and CS70

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?

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?

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?

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?

Google.

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?

Google. Plus reference.

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?

Google. Plus reference.

Syntax of pandas?

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?

Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.

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?

Google. Plus reference.

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?

Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

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?

Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?

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?

Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:

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?

Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus “how to program” and remembering a bit.

What is π?

Kolmorogorov Complexity View:

perimeter of a circle/diameter.

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate:

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus “how to program” and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1+r/n)^n \rightarrow e^r$.
Solution to: $dy/dx = y$,

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1+r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1+\frac{1}{n})^n)^x \to e^x$.

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1 + r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1 + \frac{1}{n})^n)^x \to e^x$. Population growth.

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus “how to program” and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1+r/n)^n \rightarrow e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1+\frac{1}{n})^n)^x \rightarrow e^x$. Population growth. Covid.

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1 + r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1 + \frac{1}{n})^n)^x \to e^x$. Population growth. Covid.

Calculus:

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1 + r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1 + \frac{1}{n})^n)^x \to e^x$. Population growth. Covid.

Calculus: what is minimum you need to know?

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1 + r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1 + \frac{1}{n})^n)^x \to e^x$. Population growth. Covid.

Calculus: what is minimum you need to know?
Depends on your skills!

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1+r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1+\frac{1}{n})^n)^x \to e^x$. Population growth. Covid.

Calculus: what is minimum you need to know?
Depends on your skills! Conceptualization.

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?
Google. Plus reference.

Syntax of pandas? Google + Stackoverflow.
Plus "how to program" and remembering a bit.

What is π?
Kolmorogorov Complexity View:
perimeter of a circle/diameter.

What is e?
Kolmorogorov Complexity View(s):
Continuous Interest Rate: $(1+r/n)^n \to e^r$.
Solution to: $dy/dx = y$,
$y \approx ((1+\frac{1}{n})^n)^x \to e^x$. Population growth. Covid.

Calculus: what is minimum you need to know?
Depends on your skills! Conceptualization.
Reason and understand an argument and you can generate a lot.

Calculus

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.

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.

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.

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?

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?

$f(x) = ax$, $g(x) = bx$.

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?

$f(x) = ax$, $g(x) = bx$. $f(g(x)) = ab\,x$.

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.

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!

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!

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!

$(f(g(x))' = f'(\cdot)g'(\cdot)$

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...

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...

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?

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? $f(g(x))$

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? $f(g(x))$

slope of f at $g(x)$ times slope of g at x.

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? $f(g(x))$

slope of f at $g(x)$ times slope of g at x.

$(f(g(x))' = f'(g(x))g'(x)$.

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? $f(g(x))$

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!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu.$

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu$.

used foil.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu$.

used foil.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu$.

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2$.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \rightarrow udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

Product Rule.

Idea: use rise in function value!
$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu.$
used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \rightarrow udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \rightarrow udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$. Which is small.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$. Which is small.

Product Rule.

Idea: use rise in function value!

$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \rightarrow udv + vdu$.

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2$.

$f'(x) = ax \times b + bx \times a = 2abx$.

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$. Which is small.

Any concept:

Product Rule.

Idea: use rise in function value!

$d(uv) = (u + du)(v + dv) - uv = udv + vdu + dudv \to udv + vdu.$

used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2.$

$f'(x) = ax \times b + bx \times a = 2abx.$

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$. Which is small.

Any concept:
A quick argument from basic concept of slope of a tangent line.

Product Rule.

Idea: use rise in function value!
$d(uv) = (u+du)(v+dv) - uv = udv + vdu + dudv \to udv + vdu$.
used foil.

$f(x) = u(x)v(x) = ax \times bx = abx^2$.

$f'(x) = ax \times b + bx \times a = 2abx$.

If $u(x)$ is constant, change is $u(x)dv$.
If $v(x)$ is constant, change is $v(x)du$.
Also have $dv \times du$. Which is small.

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 θ and do derivative of $\sin\theta$.

Derivative of sine?

$\sin(x)$.

What is x? An angle in radians.

Let's call it θ and do derivative of $\sin\theta$.

θ - Length of arc of unit circle

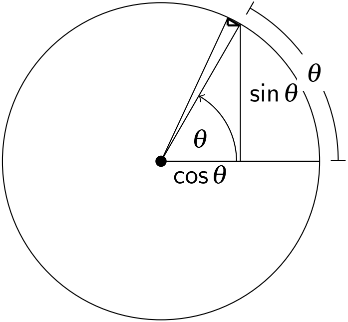
Derivative of sine?

$\sin(x)$.

What is x? An angle in radians.

Let's call it θ and do derivative of $\sin\theta$.

θ - Length of arc of unit circle



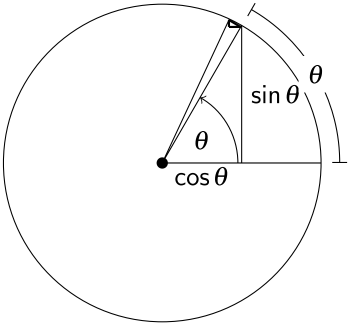
Derivative of sine?

$\sin(x)$.

What is x? An angle in radians.

Let's call it θ and do derivative of $\sin\theta$.

θ - Length of arc of unit circle



Rise.

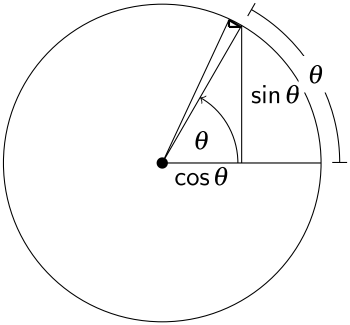
Derivative of sine?

$\sin(x)$.

What is x? An angle in radians.

Let's call it θ and do derivative of $\sin\theta$.

θ - Length of arc of unit circle



Rise.
Similar triangle!!!

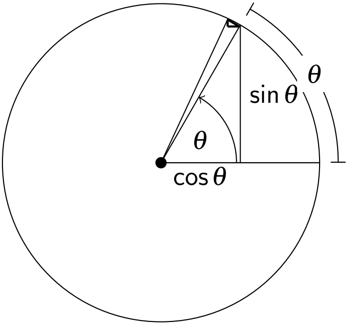
Derivative of sine?

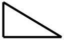
$\sin(x)$.

What is x? An angle in radians.

Let's call it θ and do derivative of $\sin\theta$.

θ - Length of arc of unit circle





Rise.
Similar triangle!!!
Rise proportional to cosine!

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

Useful?

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

Useful?
  Speed times Time is Distance.

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

Useful?
  Speed times Time is Distance.

Conceptual: Area is proportional to height.

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

Useful?
 Speed times Time is Distance.

Conceptual: Area is proportional to height.

If you change width, change in area is proportional to height.

Fundamental Theorem of Calculus.

Conceptual: Height times Width = Area.

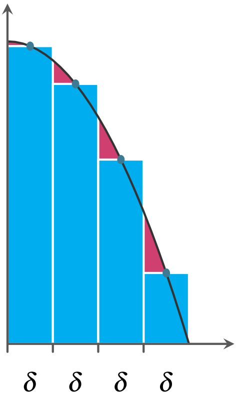
Useful?
  Speed times Time is Distance.

Conceptual: Area is proportional to height.

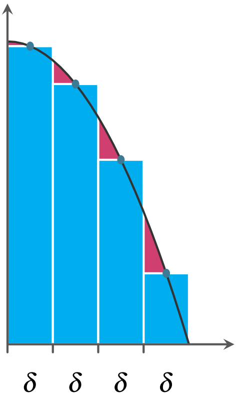
If you change width, change in area is proportional to height.

Derivative (rate of change) of Area (Integral) under curve, is height of curve.

Calculus

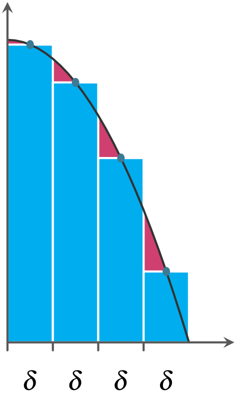


Calculus



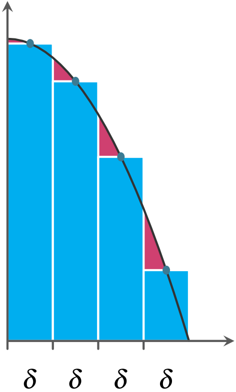
Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$

Calculus



Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$
"Area is defined as rectangles and add up some thin ones."

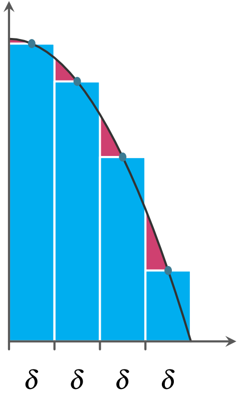
Calculus



Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$
"Area is defined as rectangles and add up some thin ones."

Derivative (Rate of change):
$$F'(x) = \lim_{h \to 0} \frac{F(x+h) - F(x)}{h}.$$

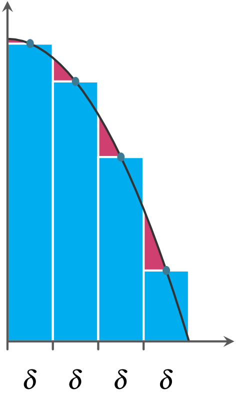
Calculus



Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$
"Area is defined as rectangles and add up some thin ones."

Derivative (Rate of change):
$F'(x) = \lim_{h \to 0} \frac{F(x+h) - F(x)}{h}$.
"Rise over run of close together points."

Calculus



Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$

"Area is defined as rectangles and add up some thin ones."

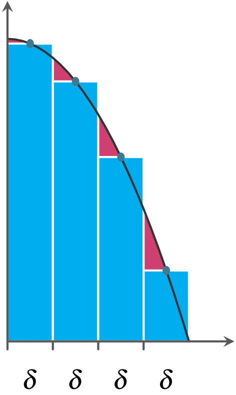
Derivative (Rate of change):

$F'(x) = \lim_{h \to 0} \frac{F(x+h)-F(x)}{h}$.

"Rise over run of close together points."

Fundamental Theorem: $F(b) - F(a) = \int_a^b F'(x)dx$.

Calculus

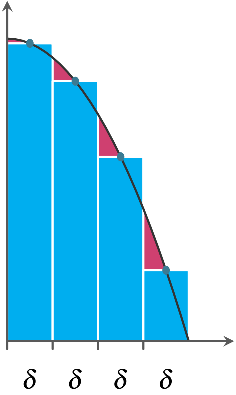


Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$
"Area is defined as rectangles and add up some thin ones."

Derivative (Rate of change):
$F'(x) = \lim_{h \to 0} \frac{F(x+h)-F(x)}{h}$.
"Rise over run of close together points."

Fundamental Theorem: $F(b) - F(a) = \int_a^b F'(x)dx$.
"Area ($F(\cdot)$) under $f(x)$ grows at x, $F'(x)$, by $f(x)$"

Calculus



Riemann Sum/Integral: $\int_a^b f(x)dx = lim_{\delta \to 0} \sum_i \delta f(a_i)$
"Area is defined as rectangles and add up some thin ones."

Derivative (Rate of change):
$F'(x) = \lim_{h \to 0} \frac{F(x+h) - F(x)}{h}$.
"Rise over run of close together points."

Fundamental Theorem: $F(b) - F(a) = \int_a^b F'(x)dx$.
"Area ($F(\cdot)$) under $f(x)$ grows at x, $F'(x)$, by $f(x)$"
Thus $F'(x) = f(x)$.

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, $\ldots$..

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. “out of”

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. "out of"

...plus reasoning.

CS 70 : ideas.

Induction

CS 70 : ideas.

Induction $\equiv$ every integer has a next one.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.
Twice number of edges is sum of degrees.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.
 Twice number of edges is sum of degrees.
  Cuz edge counted twice in sum.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.
  Twice number of edges is sum of degrees.
    Cuz edge counted twice in sum.
  $\Delta + 1$ coloring. Neighbors only take up Δ.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.

Twice number of edges is sum of degrees.

Cuz edge counted twice in sum.

$\Delta + 1$ coloring. Neighbors only take up Δ.

Connectivity plus connected components.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.
 Twice number of edges is sum of degrees.
  Cuz edge counted twice in sum.
 $\Delta + 1$ coloring. Neighbors only take up Δ.
 Connectivity plus connected components.
 Eulerian paths: if you enter you can leave.

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.

Twice number of edges is sum of degrees.

Cuz edge counted twice in sum.

$\Delta + 1$ coloring. Neighbors only take up Δ.

Connectivity plus connected components.

Eulerian paths: if you enter you can leave.

Euler's formula: tree has $v - 1$ edges and 1 face plus

CS 70 : ideas.

Induction $\equiv$ every integer has a next one. Graph theory.

- Twice number of edges is sum of degrees.
  - Cuz edge counted twice in sum.
- $\Delta + 1$ coloring. Neighbors only take up Δ.
- 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.
  - $v - 1 + (f - 1) = e$

CS 70 : ideas.

Number theory.

A divisor of x and y divides $x - y$.

The remainder is always smaller than the divisor.

$\Longrightarrow$ Euclid's GCD algorithm.

Multiplicative Inverse.

Fermat's theorem: function with inverse is a bijection.

Multiplication is commutative.

Gives RSA.

CS 70 : ideas.

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: function with inverse is a bijection.
Multiplication is commutative.
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 an equation is wrong, then multiply them by zero, i.e., Error polynomial.

Probability

Next up?

Probability

Next up? Probability.

Probability

Next up? Probability.

A bag contains:

Probability

Next up? Probability.

A bag contains:

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue.

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total.

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now:

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now: Counting!

Probability

Next up? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now: Counting!

Later: Probability.