61A Extra Lecture 1

Thursday, January 29

Announcements

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A
  - Complete ~6 difficult assignments, which may be released/due at strange times

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A
  - Complete ~6 difficult assignments, which may be released/due at strange times
  - Only for people who really want extra work that's beyond the scope of normal CS 61A

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A
  - Complete ~6 difficult assignments, which may be released/due at strange times
  - Only for people who really want extra work that's beyond the scope of normal CS 61A
- Anyone is welcome to attend the extra lectures, whether or not they enroll

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A
  - Complete ~6 difficult assignments, which may be released/due at strange times
  - Only for people who really want extra work that's beyond the scope of normal CS 61A
- Anyone is welcome to attend the extra lectures, whether or not they enroll
- Lectures will be on Thursdays 5-6:30 PM in 2050 VLSB; A schedule will be posted eventually

Announcements

- If you want 1 unit (pass/no pass) of credit for this CS 98, you need to:
  - Enroll in "Additional Topics on the Structure and Interpretation of Computer Programs"
  - Course control number: 25709
  - Concurrently enroll in CS 61A
  - Complete ~6 difficult assignments, which may be released/due at strange times
  - Only for people who really want extra work that's beyond the scope of normal CS 61A
- Anyone is welcome to attend the extra lectures, whether or not they enroll
- Lectures will be on Thursdays 5-6:30 PM in 2050 VLSB; A schedule will be posted eventually
- John's office hours: 10am-12pm Wednesday & Friday by appt. (denero.org/meet) in 781 Soda

Lambda Expressions

(Demo)

Lambda Expressions

Lambda Expressions

```
>>> x = 10
```

Lambda Expressions

```
>>> x = 10

>>> square = x * x
```

Lambda Expressions

```
>>> x = 10

>>> square = x * x
```

An expression: this one evaluates to a number

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

```
>>> square = lambda x: x * x
```

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

A function

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

A function

with formal parameter x

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

A function

with formal parameter x

that returns the value of "x * x"

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Must be a single expression

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Must be a single expression

```
>>> square(4)
16
```

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Must be a single expression

```
>>> square(4)
16
```

Lambda expressions are not common in Python, but important in general

Lambda Expressions

```
>>> x = 10
```

An expression: this one evaluates to a number

```
>>> square = x * x
```

Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```

Important: No "return" keyword!

A function

with formal parameter x

that returns the value of "x * x"

Must be a single expression

```
>>> square(4)
16
```

Lambda expressions are not common in Python, but important in general

Lambda expressions in Python cannot contain statements at all!

Newton's Method

Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!

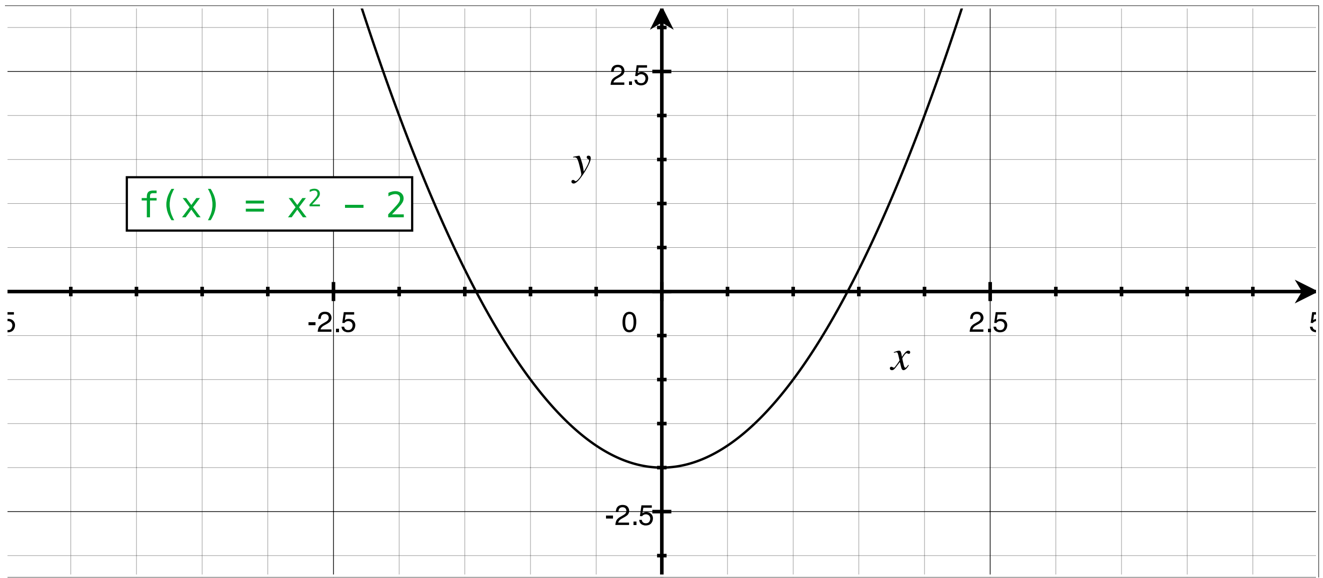
Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!

$$\boxed{f(x) = x^2 - 2}$$

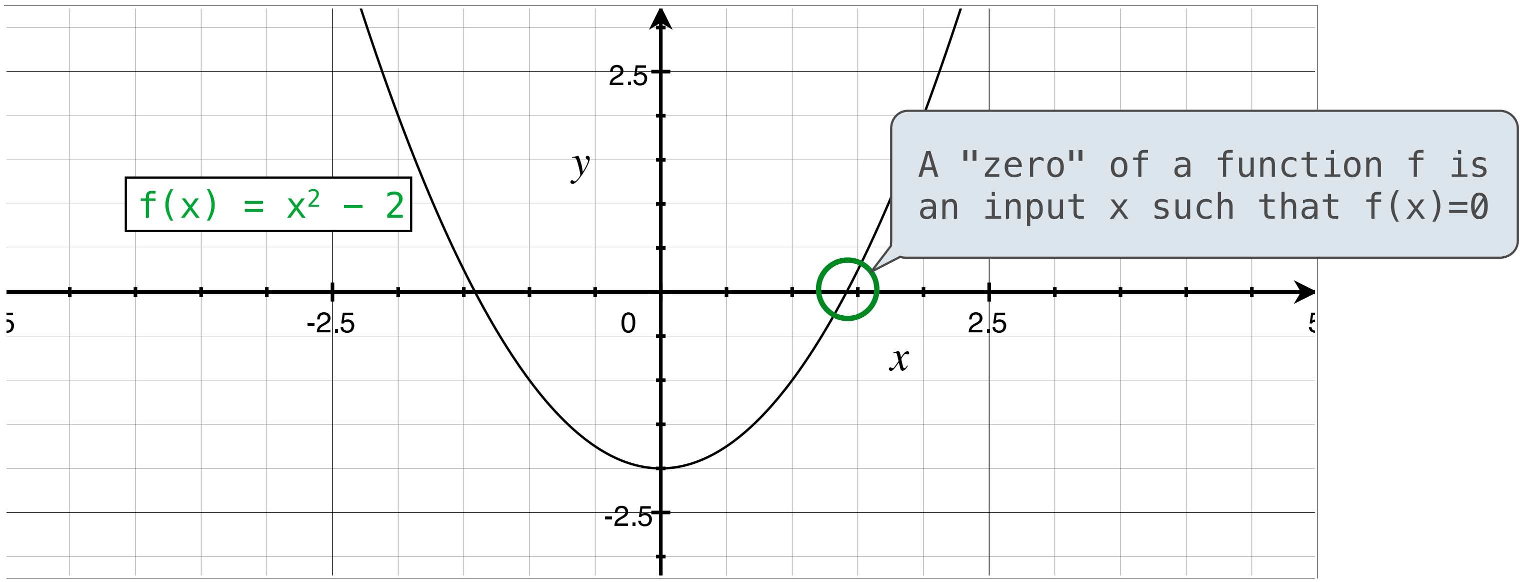
Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!



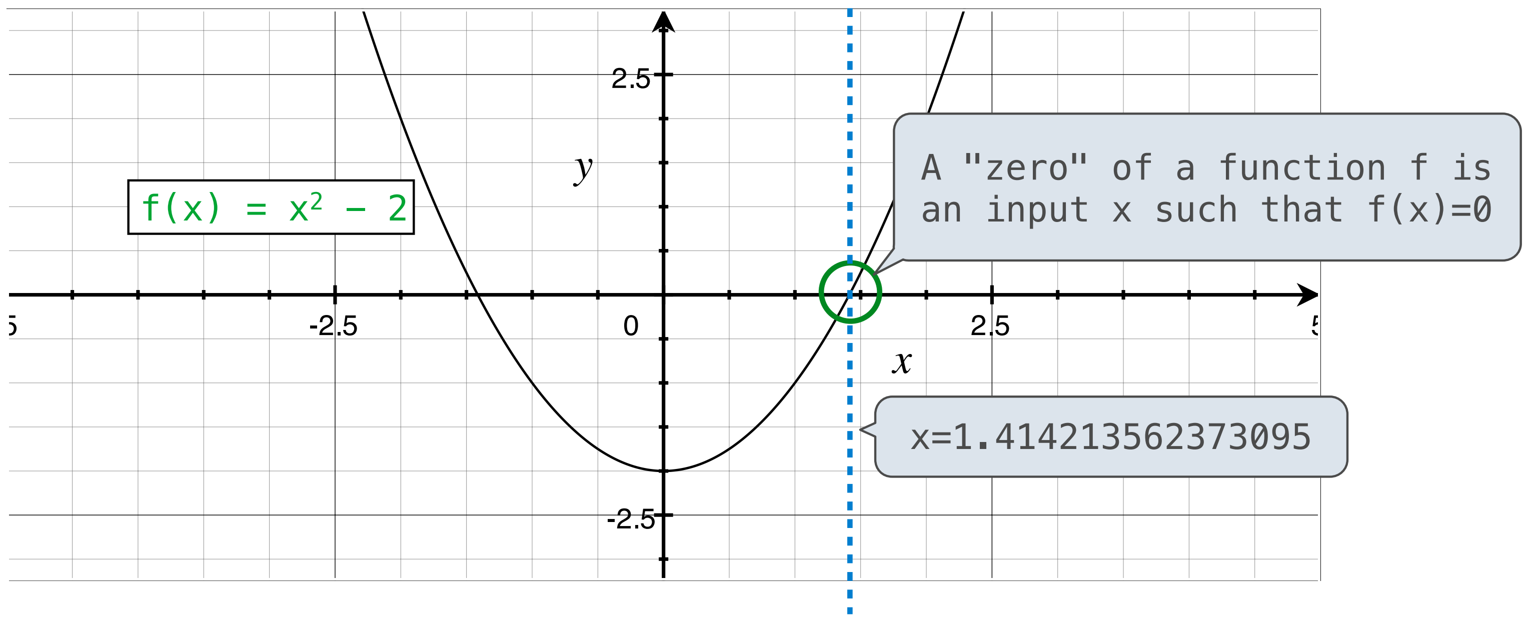
Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!



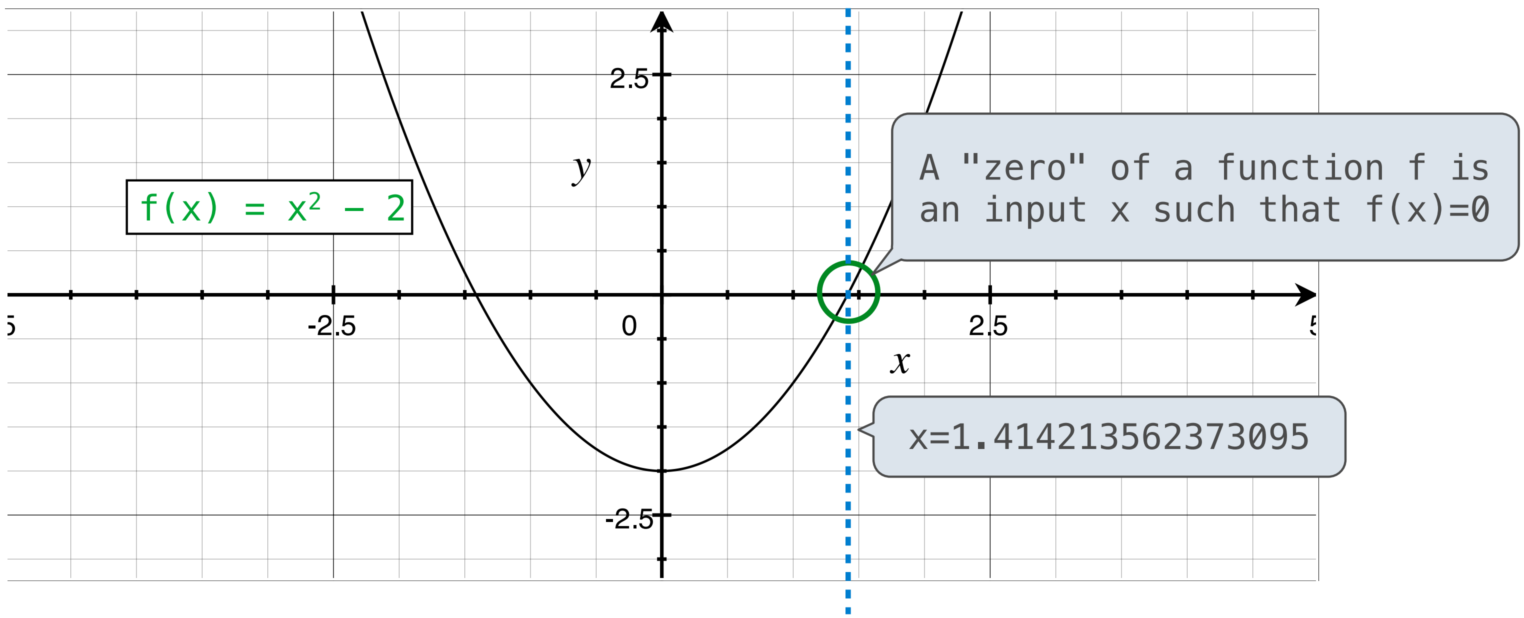
Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!



Newton's Method Background

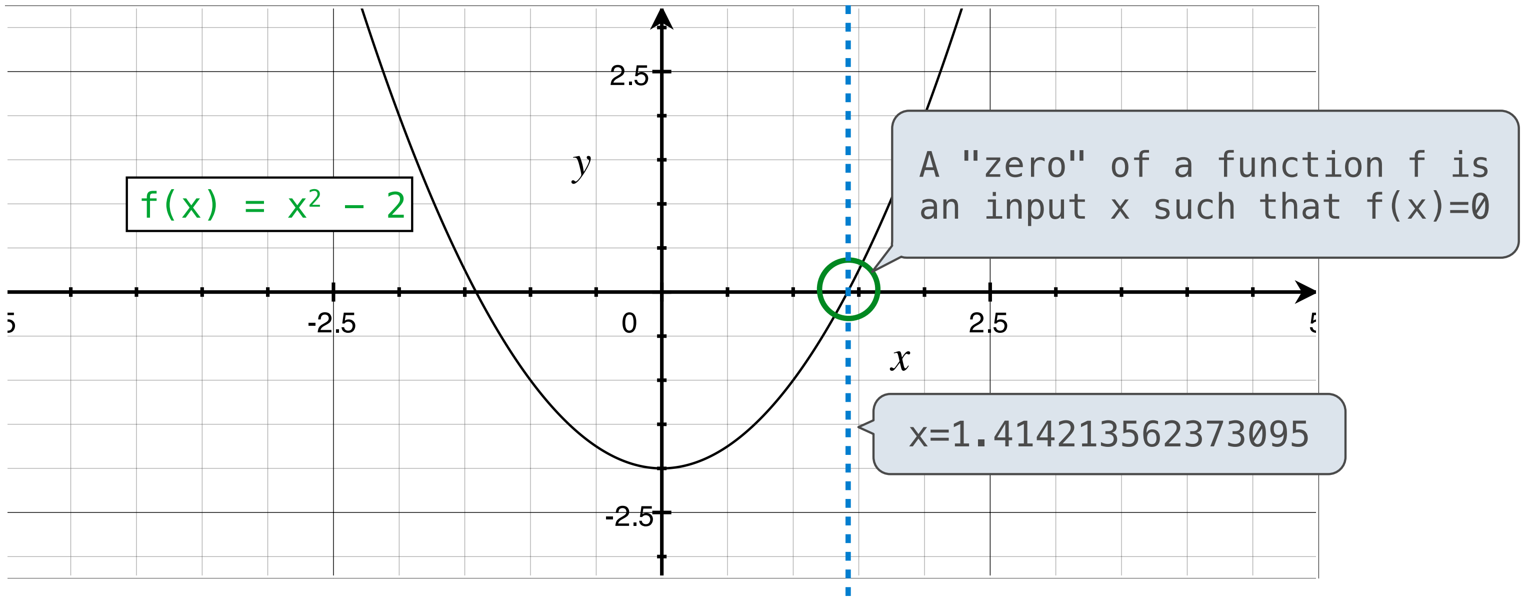
```
Quickly finds accurate approximations to zeroes of differentiable functions!
```




```
Application: a method for computing square roots, cube roots, etc.
```

Newton's Method Background

Quickly finds accurate approximations to zeroes of differentiable functions!



Application: a method for computing square roots, cube roots, etc.

The positive zero of $f(x) = x^2 - a$ is $\sqrt{a}$. (We're solving the equation $x^2 = a$.)

Newton's Method

Given a function f and initial guess x,

Newton's Method

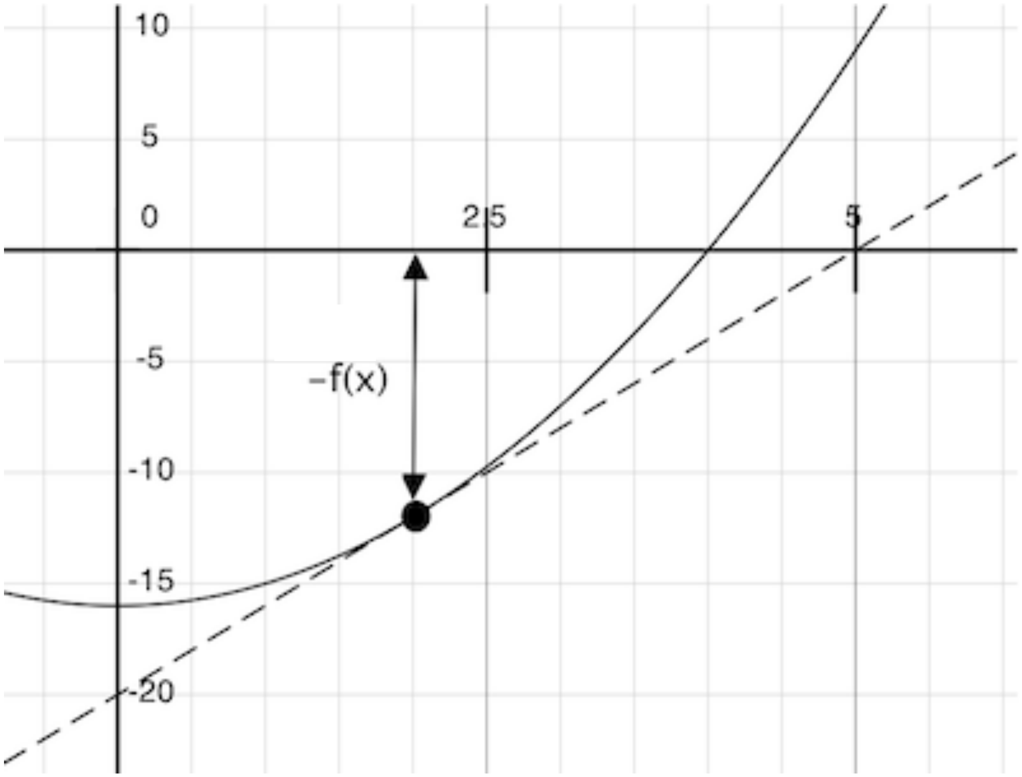
```
Given a function f and initial guess x,

Repeatedly improve x:
```

Newton's Method

```
Given a function f and initial guess x,
```

```
Repeatedly improve x:
```

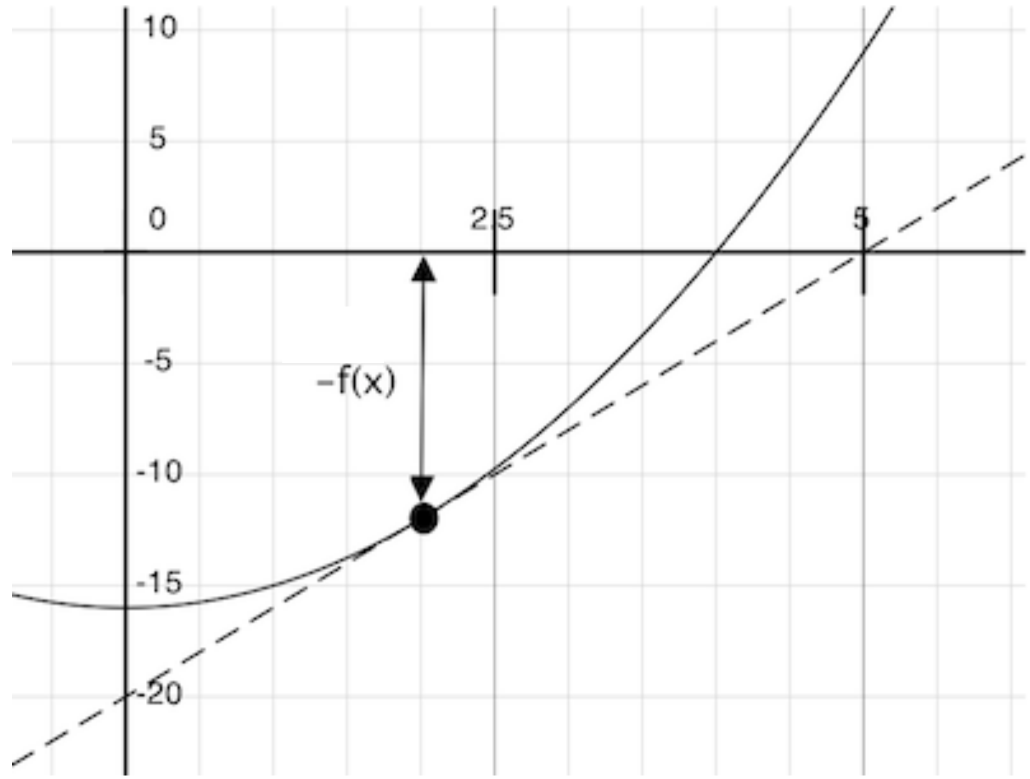


Newton's Method

```
Given a function f and initial guess x,


Repeatedly improve x:

     Compute the value of f
     at the guess: f(x)
```



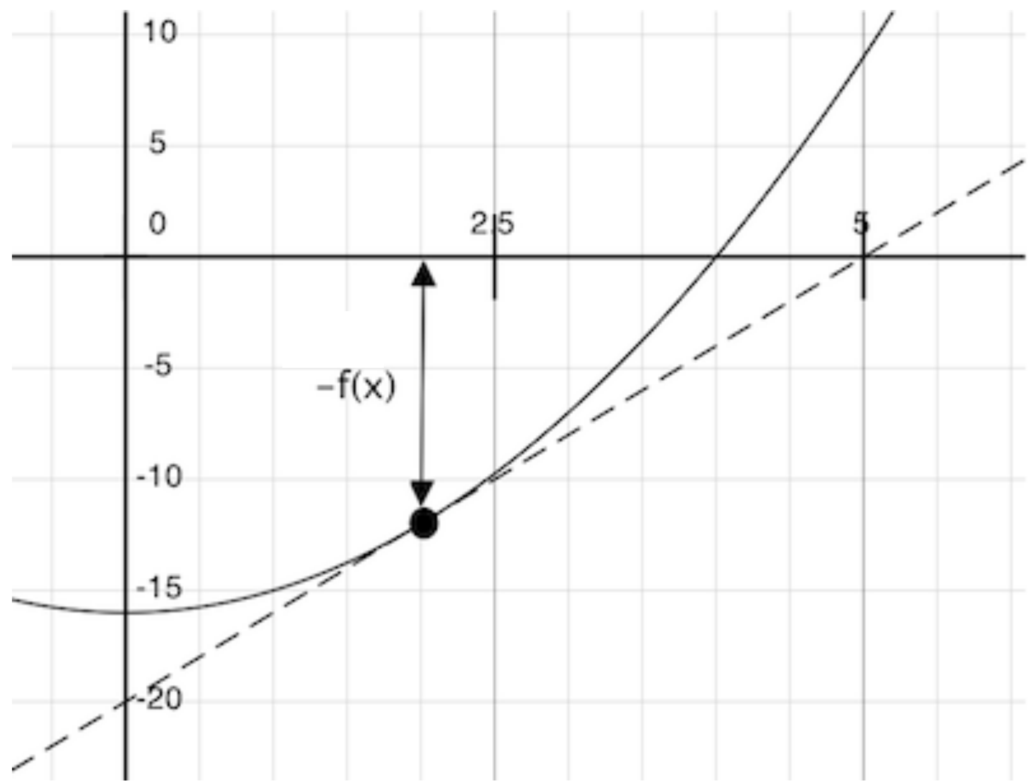
Newton's Method

Given a function f and initial guess x,

Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)



Newton's Method

Given a function f and initial guess x,

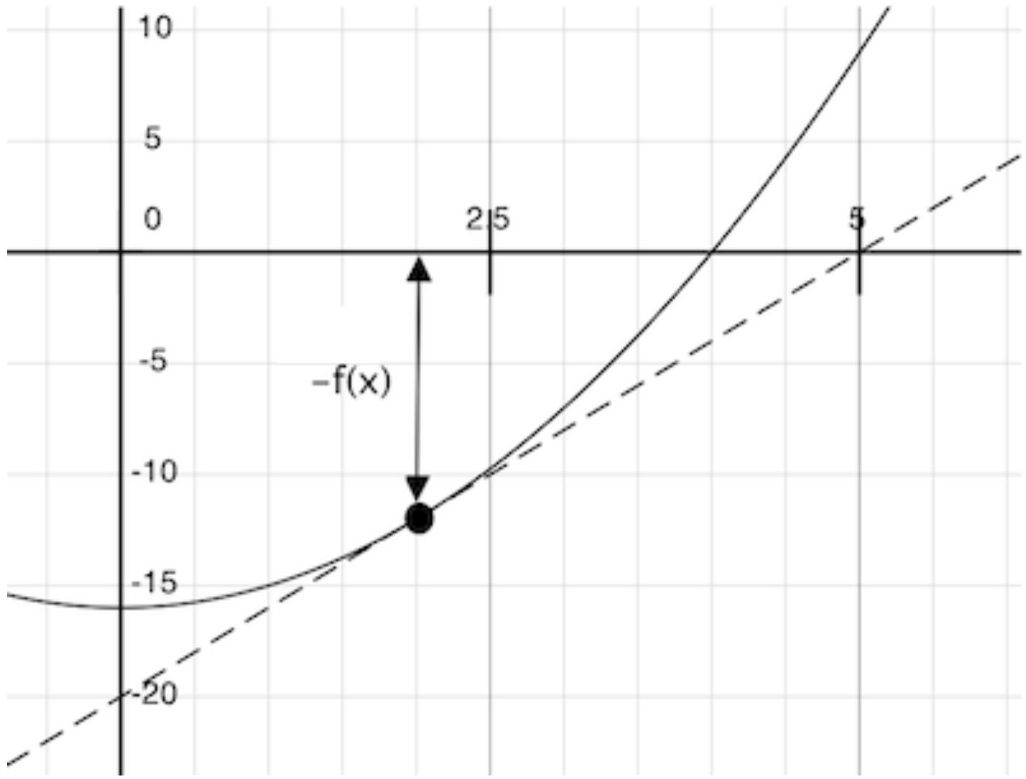
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

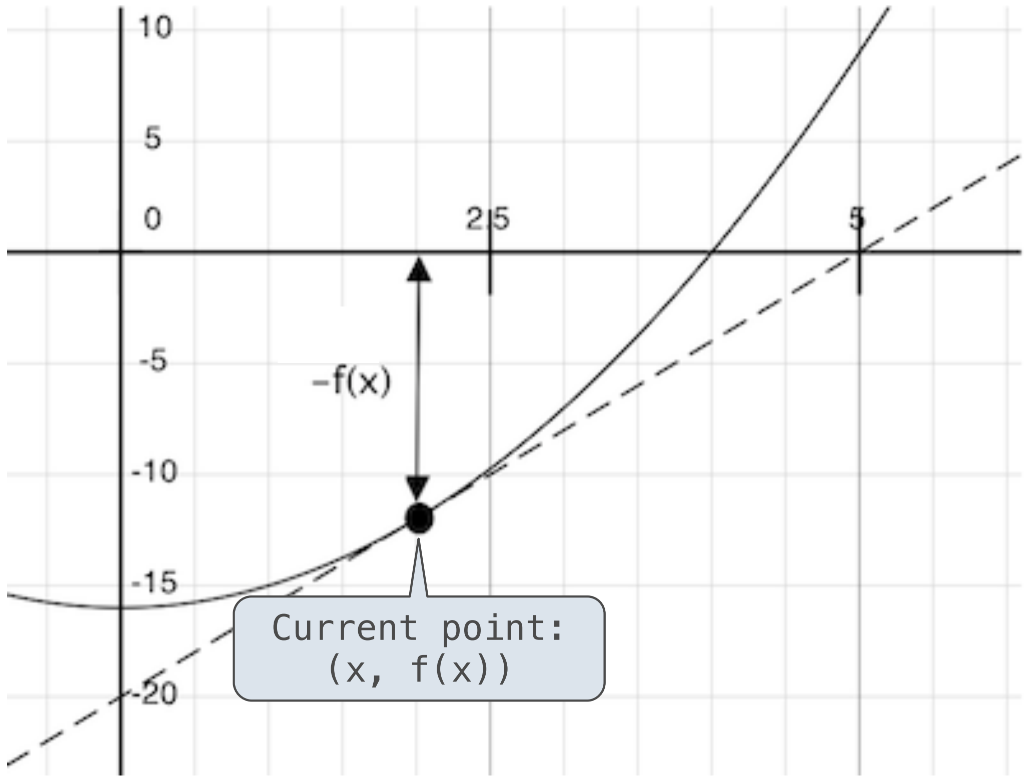
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

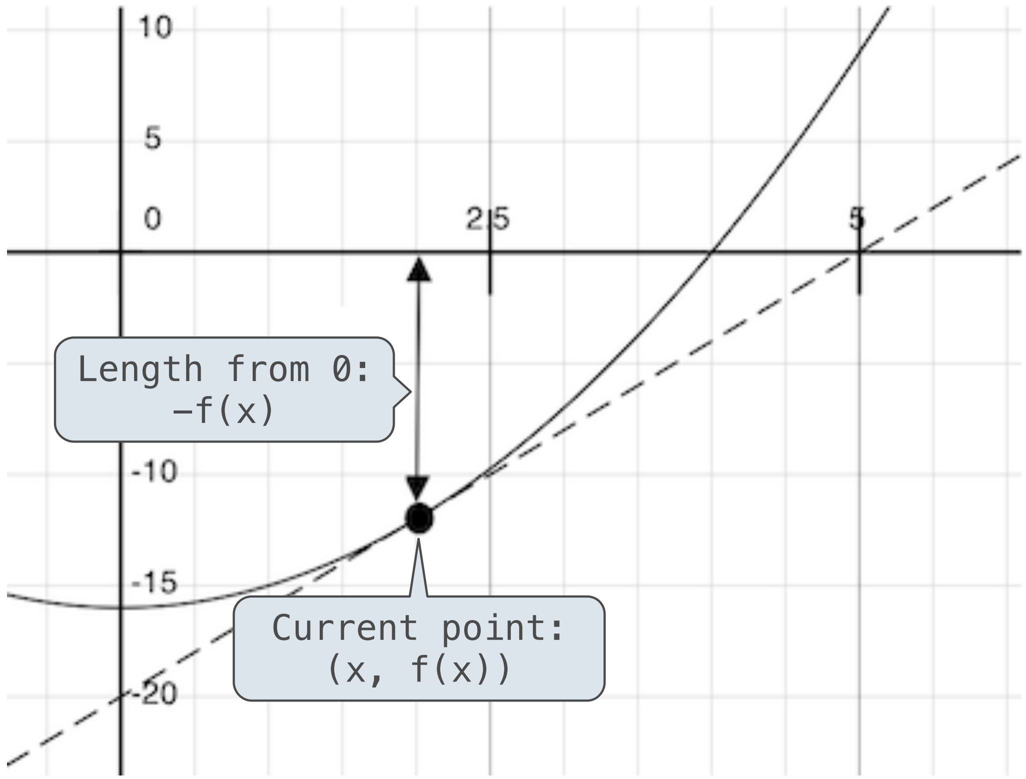
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

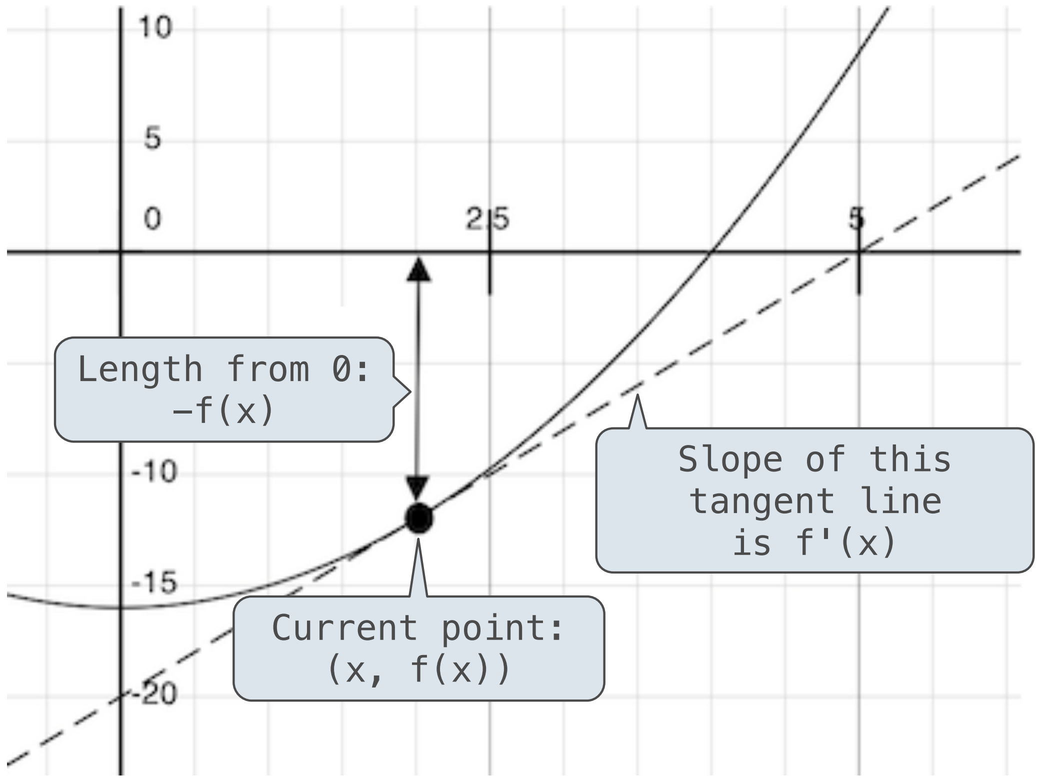
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

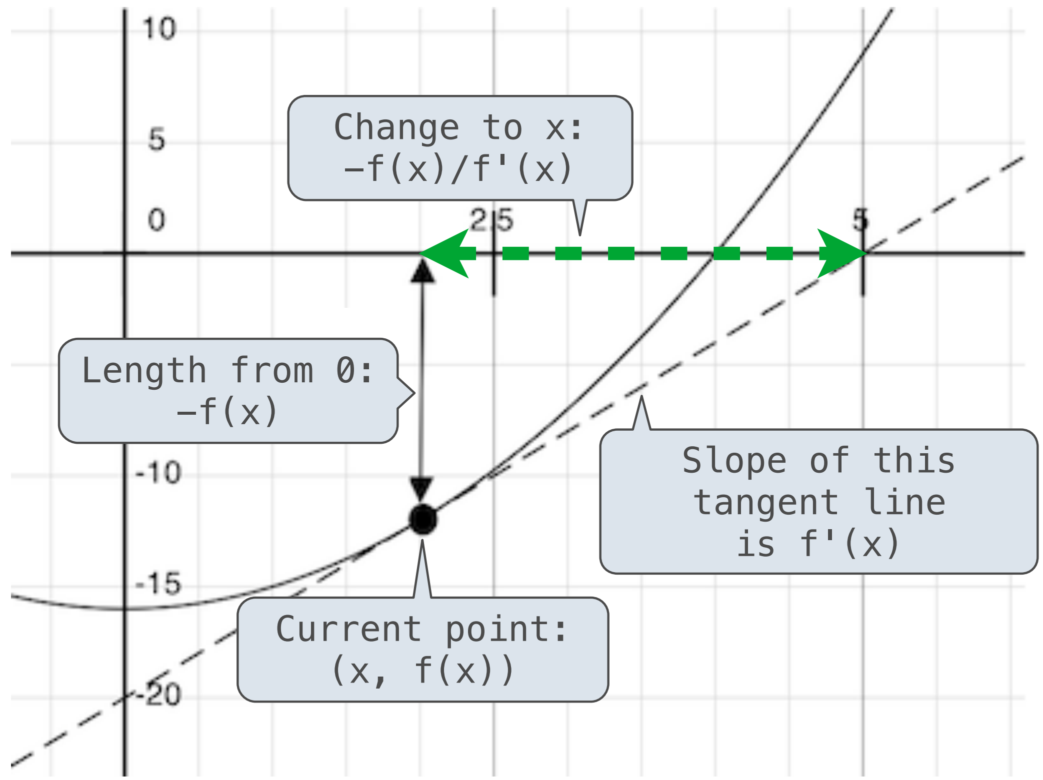
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

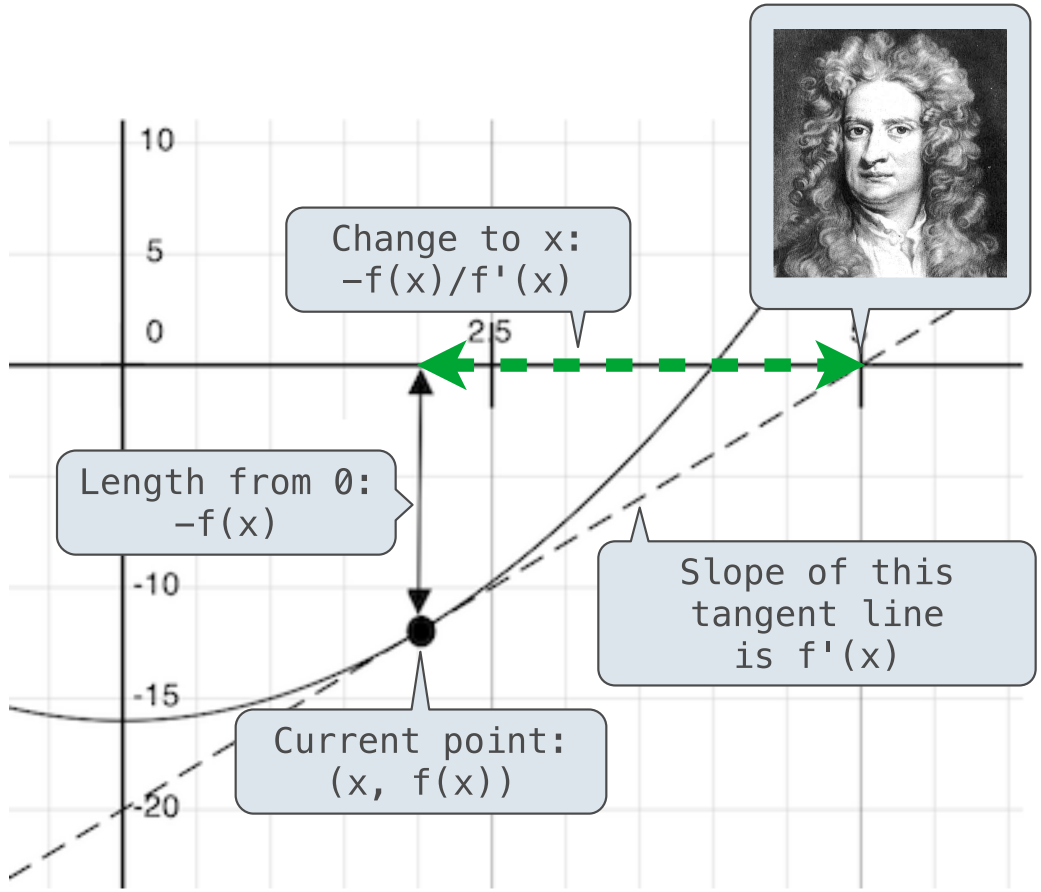
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

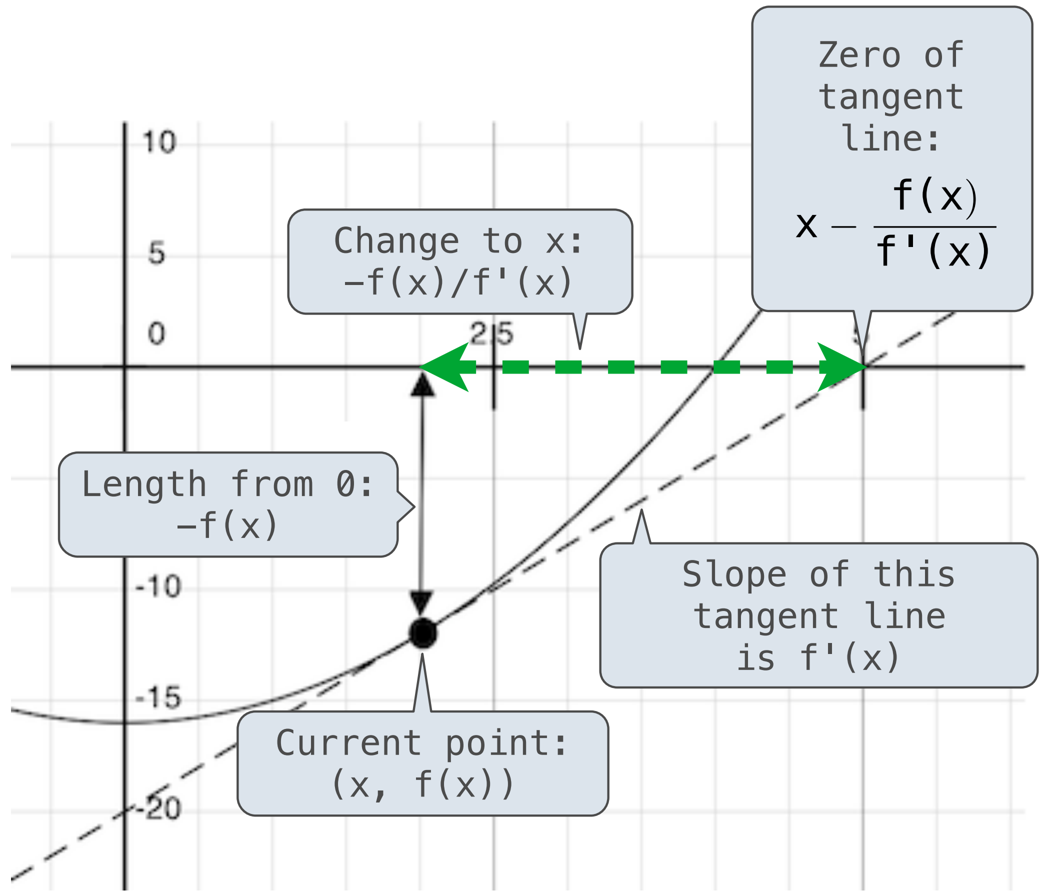
Repeatedly improve x:

Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$



Newton's Method

Given a function f and initial guess x,

Repeatedly improve x:

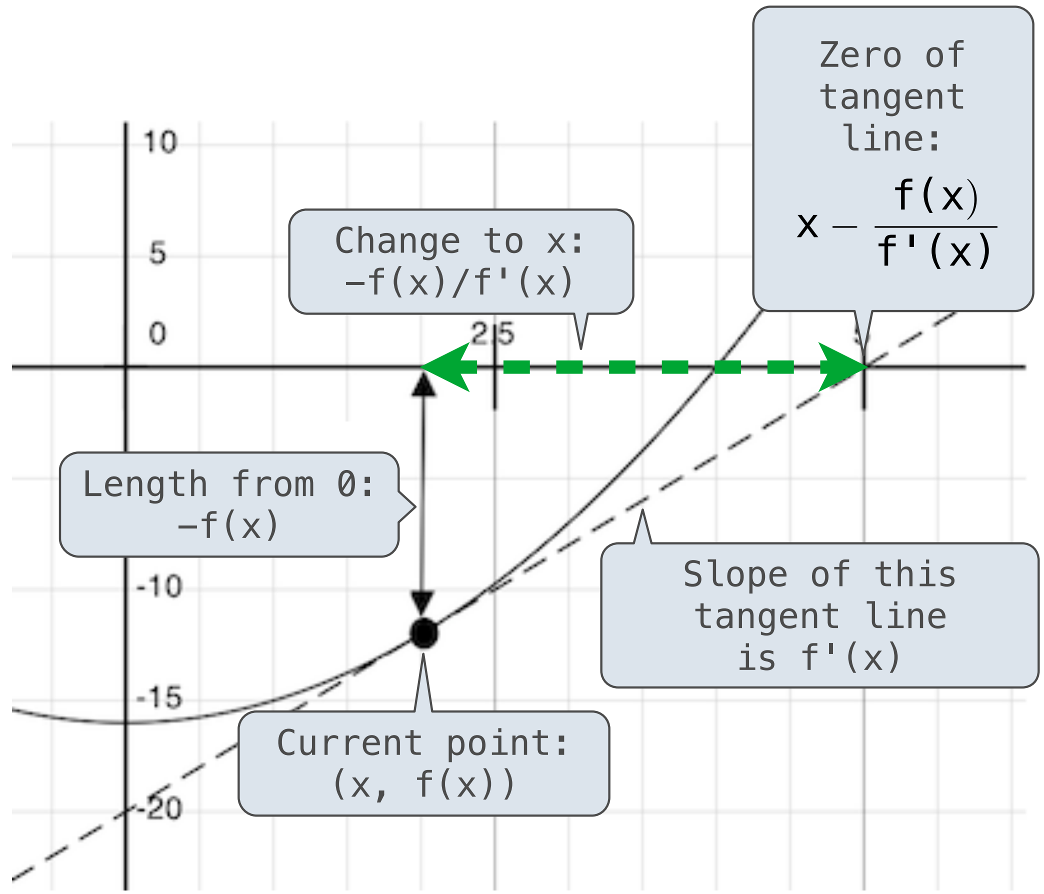
Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$

Finish when f(x) = 0 (or close enough)



Newton's Method

Given a function f and initial guess x,

Repeatedly improve x:

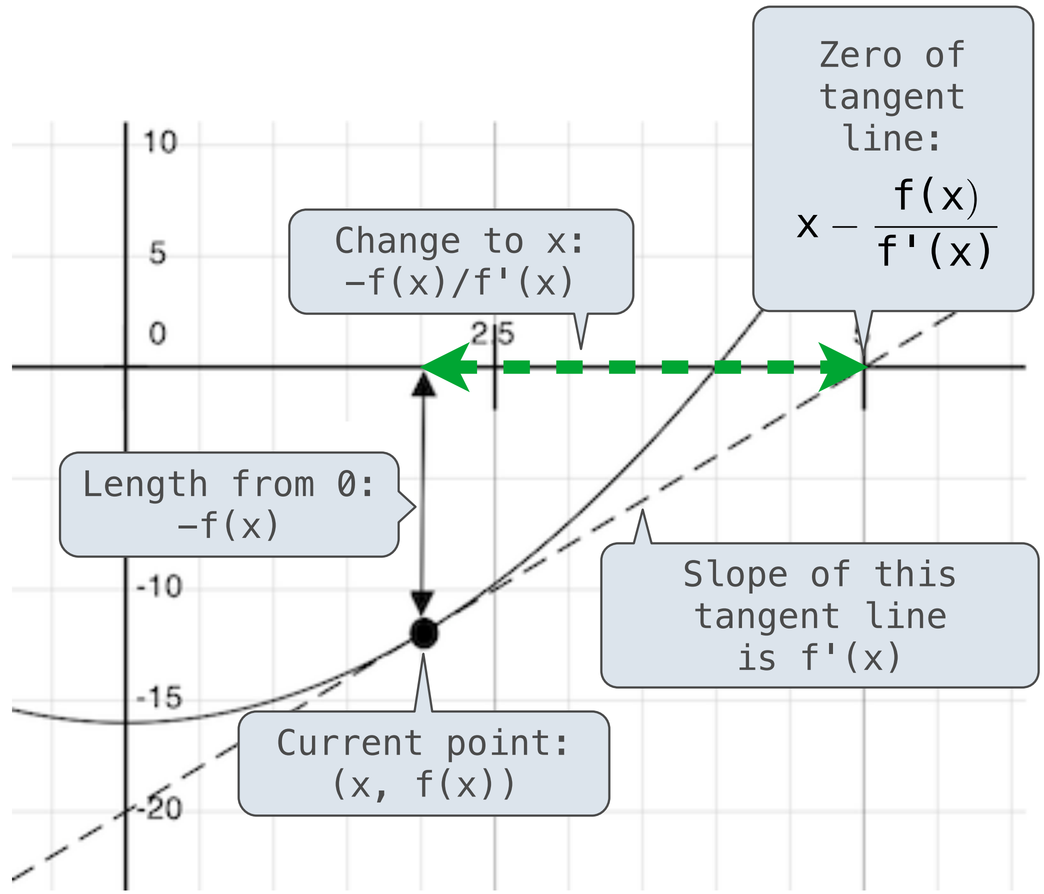
Compute the value of f at the guess: f(x)

Compute the derivative of f at the guess: f'(x)

Update guess x to be:

$$x - \frac{f(x)}{f'(x)}$$

Finish when f(x) = 0 (or close enough)



http://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif

Using Newton's Method

Using Newton's Method

```
How to find the square root of 2?
```

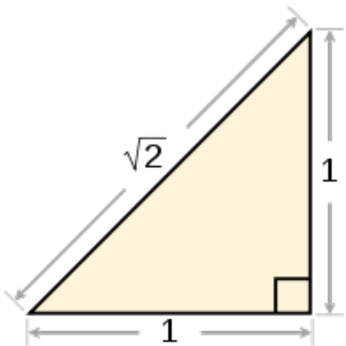
Using Newton's Method

How to find the square root of 2?

```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

Using Newton's Method

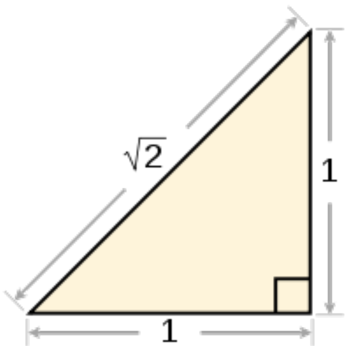
How to find the square root of 2?




```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

Using Newton's Method

How to find the square root of 2?



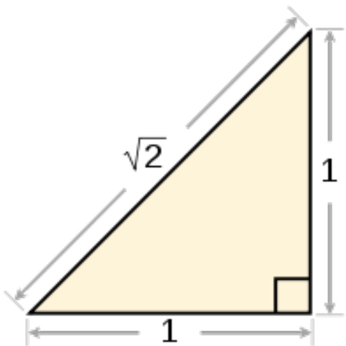

```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

$f(x) = x^2 - 2$

$f'(x) = 2x$

Using Newton's Method

How to find the square root of 2?




```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

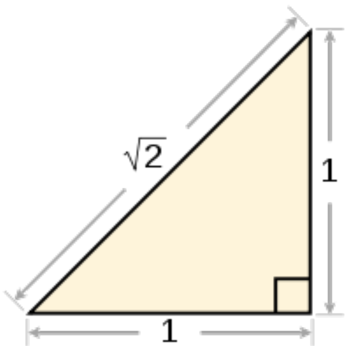
$f(x) = x^2 - 2$

$f'(x) = 2x$

Applies Newton's method

Using Newton's Method

How to find the square root of 2?




```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

$f(x) = x^2 - 2$

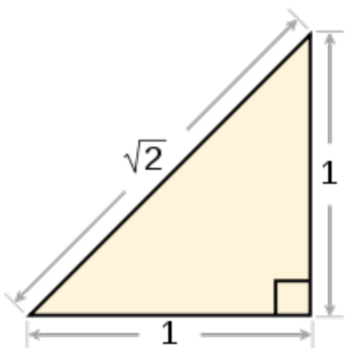
$f'(x) = 2x$

Applies Newton's method

How to find the cube root of 729?

Using Newton's Method

How to find the square root of 2?



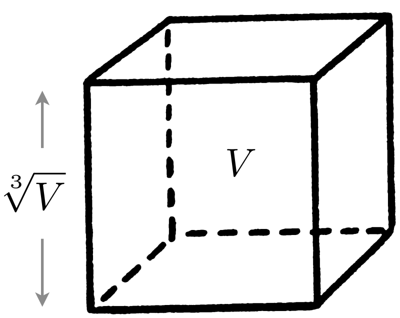

```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

$f(x) = x^2 - 2$

$f'(x) = 2x$

Applies Newton's method

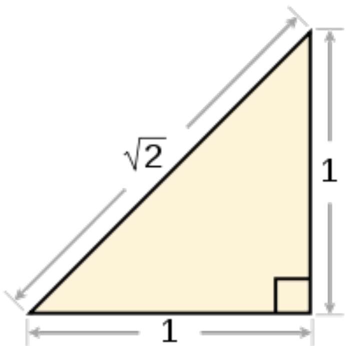
How to find the cube root of 729?



$\sqrt[3]{V}$

Using Newton's Method

How to find the square root of 2?

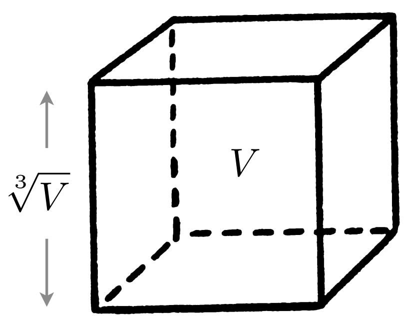



```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

$f(x) = x^2 - 2$
$f'(x) = 2x$

Applies Newton's method

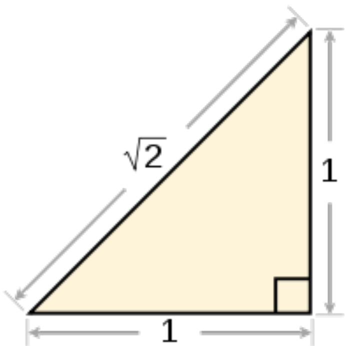
How to find the cube root of 729?




```
>>> g  = lambda x: x*x*x - 729
>>> dg = lambda x: 3*x*x
>>> find_zero(g, dg)
9.0
```

Using Newton's Method

How to find the square root of 2?

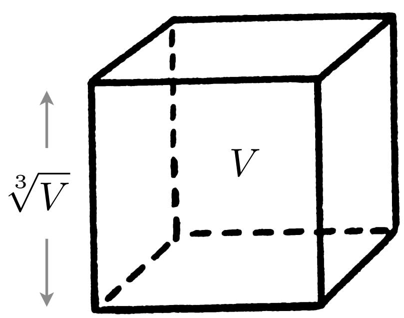



```
>>> f  = lambda x: x*x - 2
>>> df = lambda x: 2*x
>>> find_zero(f, df)
1.4142135623730951
```

$f(x) = x^2 - 2$
$f'(x) = 2x$

Applies Newton's method

How to find the cube root of 729?




```
>>> g  = lambda x: x*x*x - 729
>>> dg = lambda x: 3*x*x
>>> find_zero(g, dg)
9.0
```

$g(x) = x^3 - 729$
$g'(x) = 3x^2$

Iterative Improvement

Special Case: Square Roots

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update:

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$

Babylonian Method

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$

Babylonian Method

(Demo)

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$ (Demo)

Babylonian Method

Implementation questions:

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$

(Demo)

Babylonian Method

Implementation questions:

What guess should start the computation?

Special Case: Square Roots

How to compute square_root(a)

Idea: Iteratively refine a guess x about the square root of a

Update: $$x = \frac{x + \frac{a}{x}}{2}$$

Babylonian Method

(Demo)

Implementation questions:

What guess should start the computation?

How do we know when we are finished?

Special Case: Cube Roots

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update:

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update:

$$x = \frac{2 \cdot x + \frac{a}{x^2}}{3}$$

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update: $$x = \frac{2 \cdot x + \frac{a}{x^2}}{3}$$ (Demo)

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update: $$x = \frac{2 \cdot x + \frac{a}{x^2}}{3}$$ (Demo)

Implementation questions:

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update: $$x = \frac{2 \cdot x + \frac{a}{x^2}}{3}$$ (Demo)

Implementation questions:

What guess should start the computation?

Special Case: Cube Roots

How to compute cube_root(a)

Idea: Iteratively refine a guess x about the cube root of a

Update: $$x = \frac{2 \cdot x + \frac{a}{x^2}}{3}$$ (Demo)

Implementation questions:

What guess should start the computation?

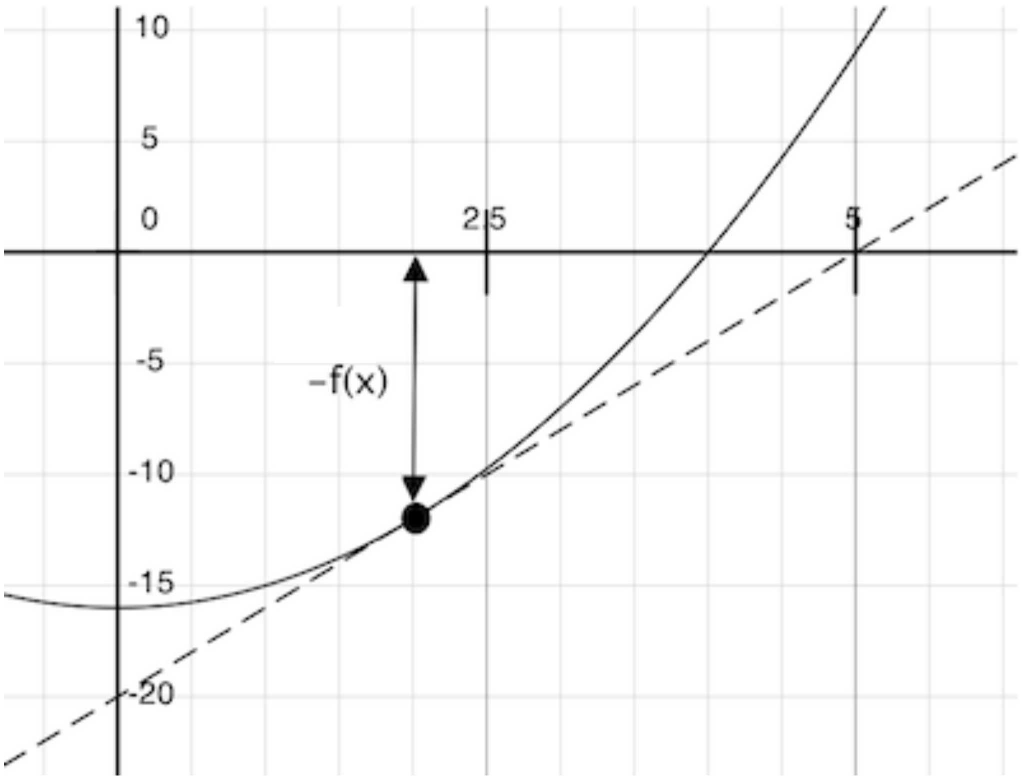
How do we know when we are finished?

Implementing Newton's Method

(Demo)

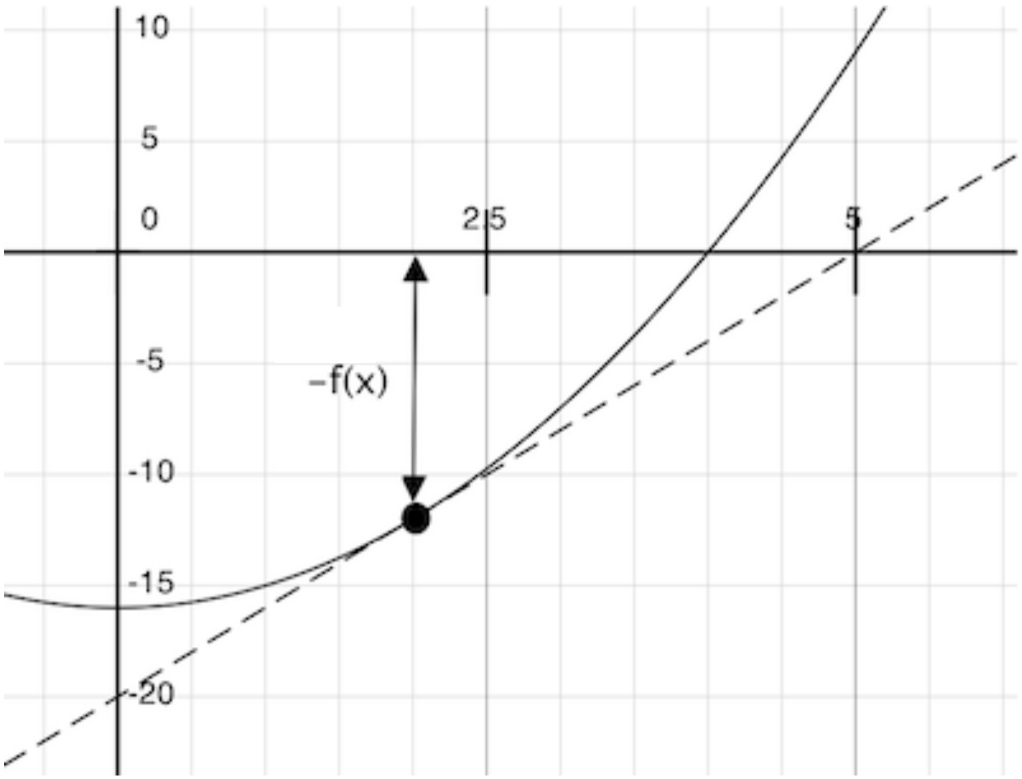
Approximate Differentiation

Approximate Differentiation



Approximate Differentiation

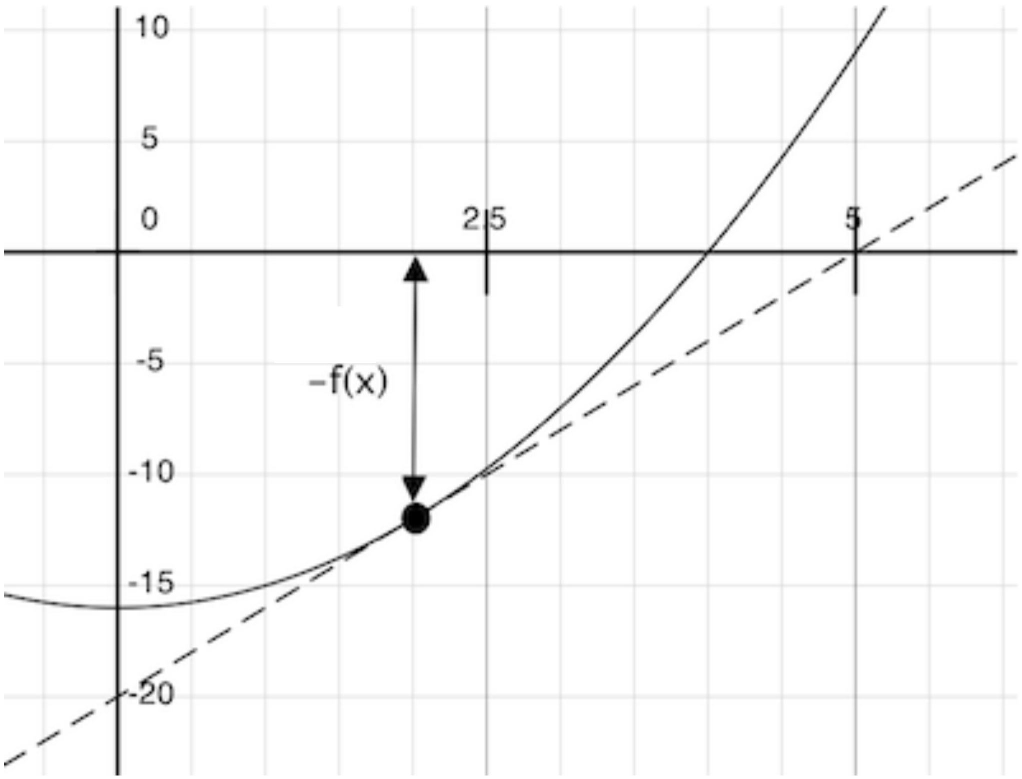
```
Differentiation can be performed
symbolically or numerically
```



Approximate Differentiation

```
Differentiation can be performed
symbolically or numerically
```

f(x) = $x^2 - 16$

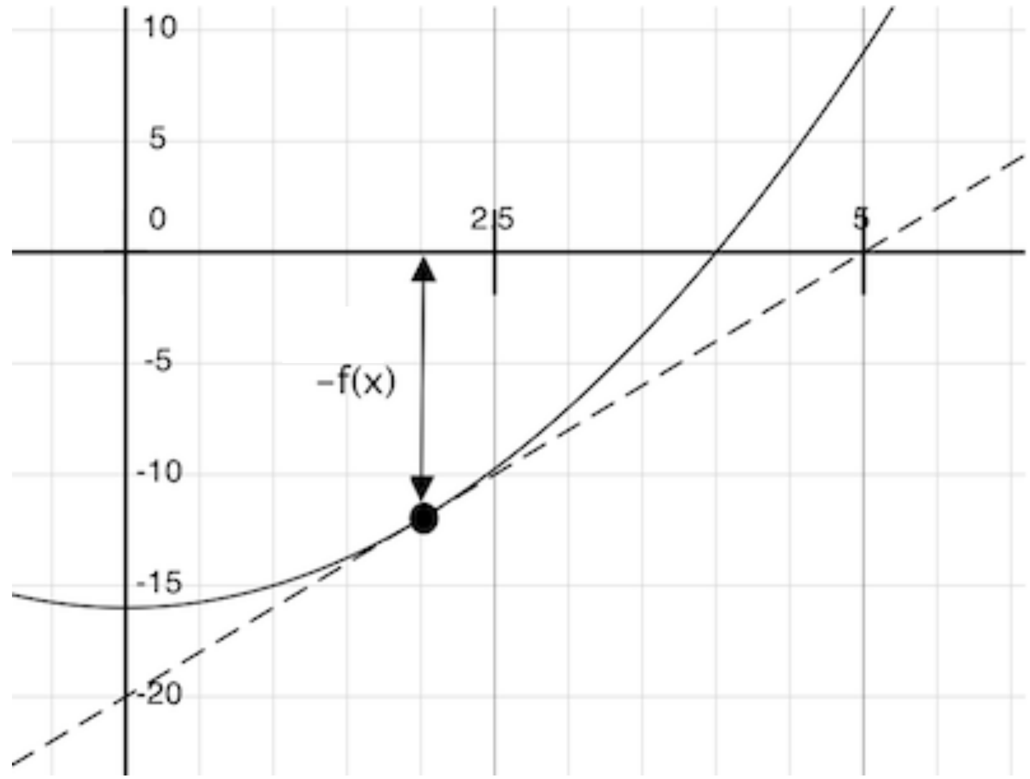


Approximate Differentiation

```
Differentiation can be performed
symbolically or numerically
```

$f(x) = x^2 - 16$

$f'(x) = 2x$



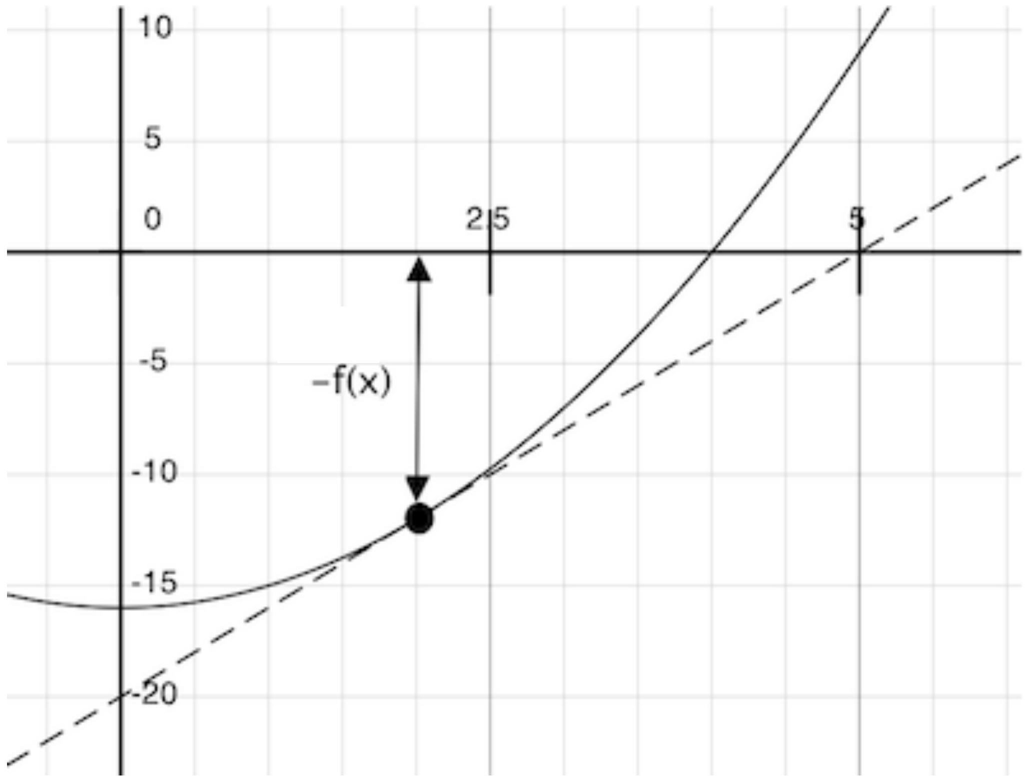
Approximate Differentiation

```
Differentiation can be performed
symbolically or numerically
```

$f(x) = x^2 - 16$

$f'(x) = 2x$

$f'(2) = 4$



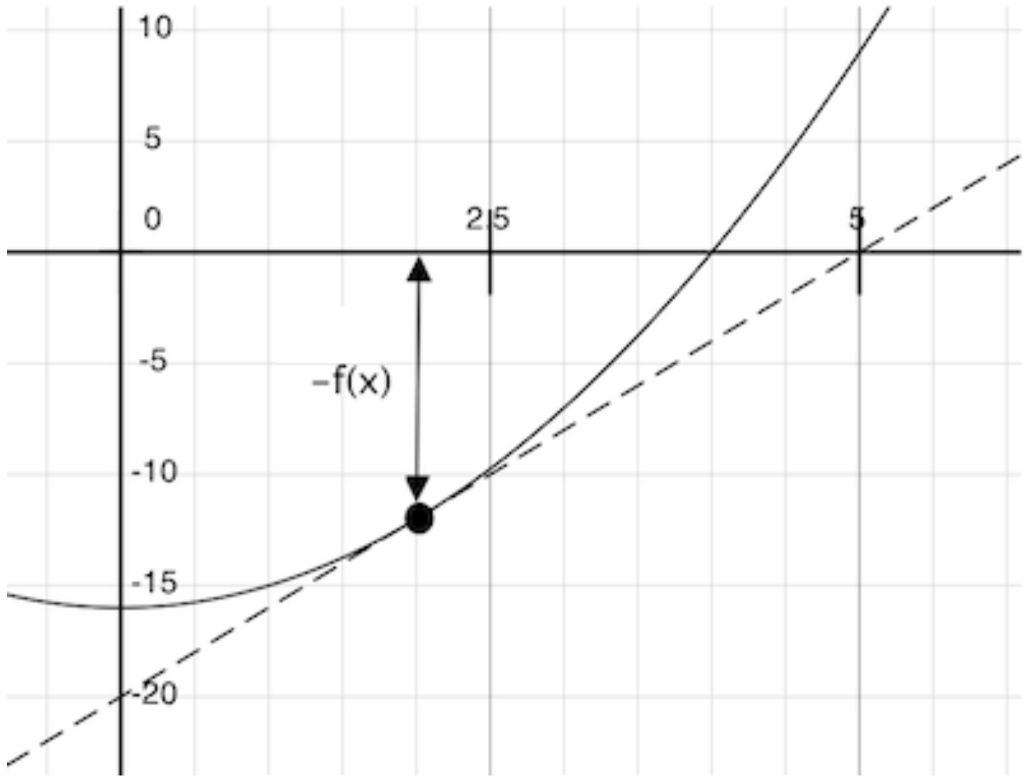
Approximate Differentiation

Differentiation can be performed symbolically or numerically

$$f(x) = \boxed{x^2} - 16$$

$$f'(x) = \boxed{2x}$$

$$f'(2) = 4$$



Approximate Differentiation

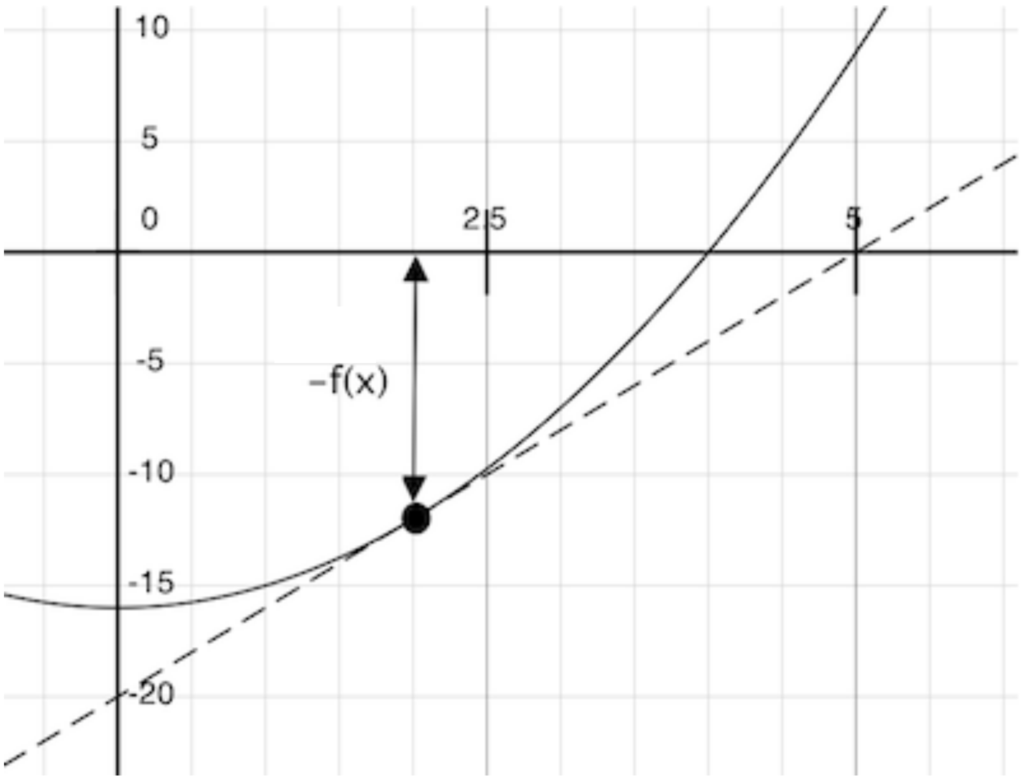
```
Differentiation can be performed
symbolically or numerically
```

$f(x) = x^2 - 16$

$f'(x) = 2x$

$f'(2) = 4$

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$



Approximate Differentiation

```
Differentiation can be performed
symbolically or numerically
```

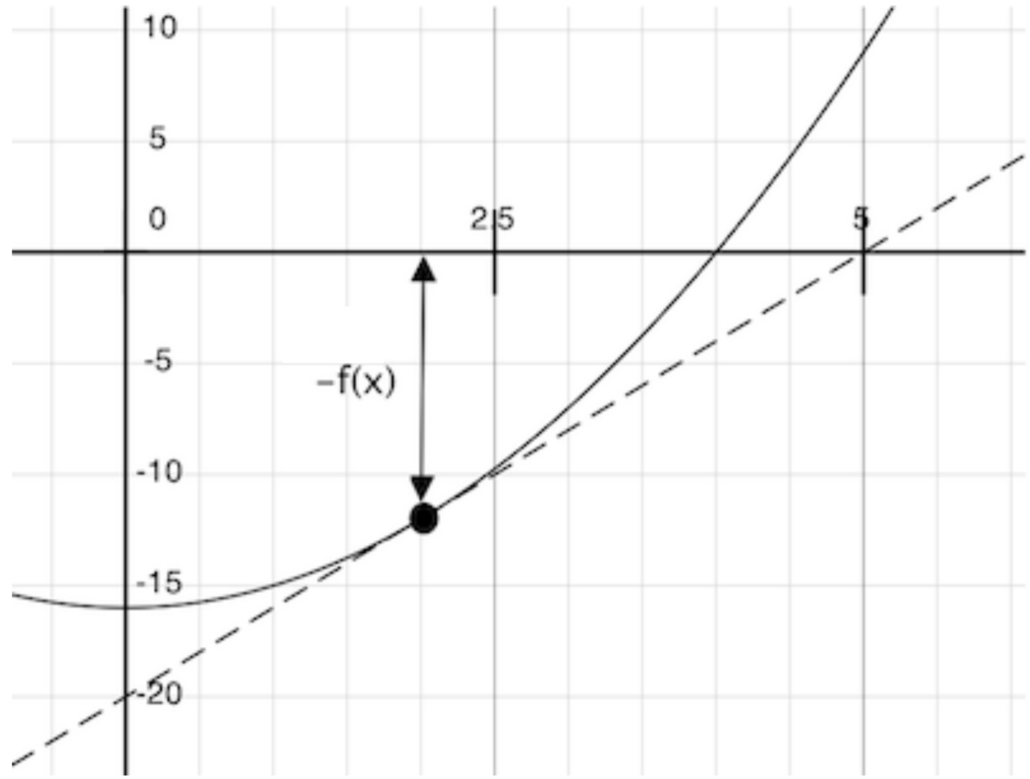
$f(x) = \boxed{x^2} - 16$

$f'(x) = \boxed{2x}$

$f'(2) = 4$

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$

$$f'(x) \approx \frac{f(x+a) - f(x)}{a}$$



Approximate Differentiation

Differentiation can be performed symbolically or numerically

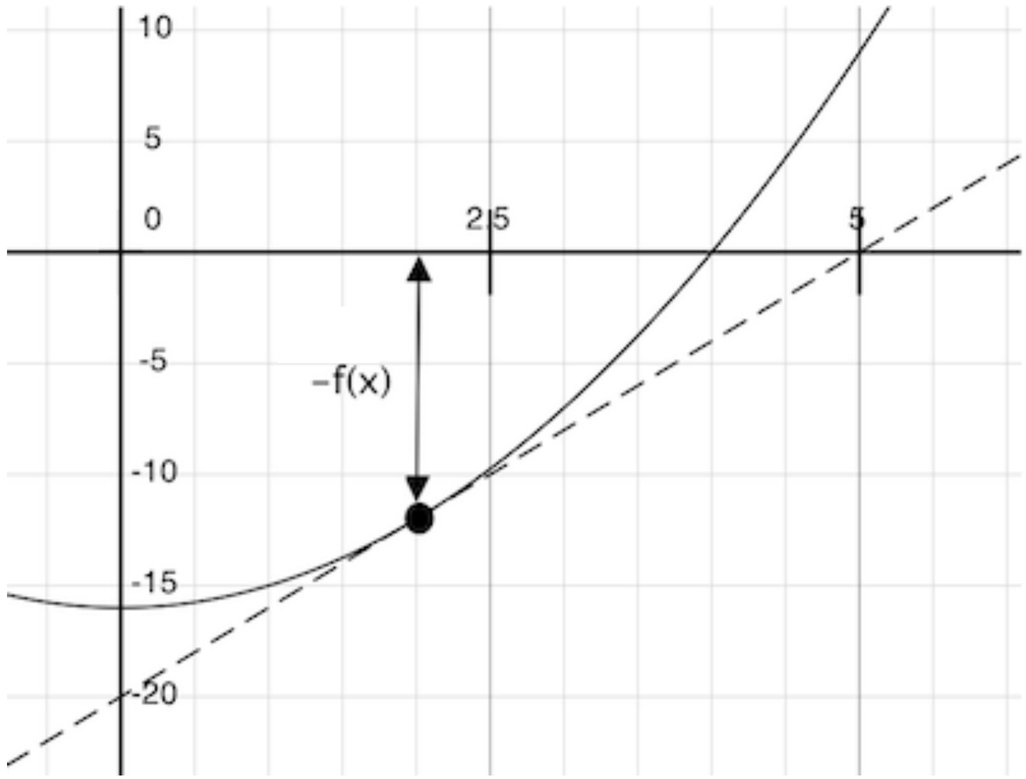
$f(x) = \boxed{x^2} - 16$

$f'(x) = \boxed{2x}$

$f'(2) = 4$

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$

$$f'(x) \approx \frac{f(x+a) - f(x)}{a}$$ (if a is small)



Approximate Differentiation

Differentiation can be performed symbolically or numerically

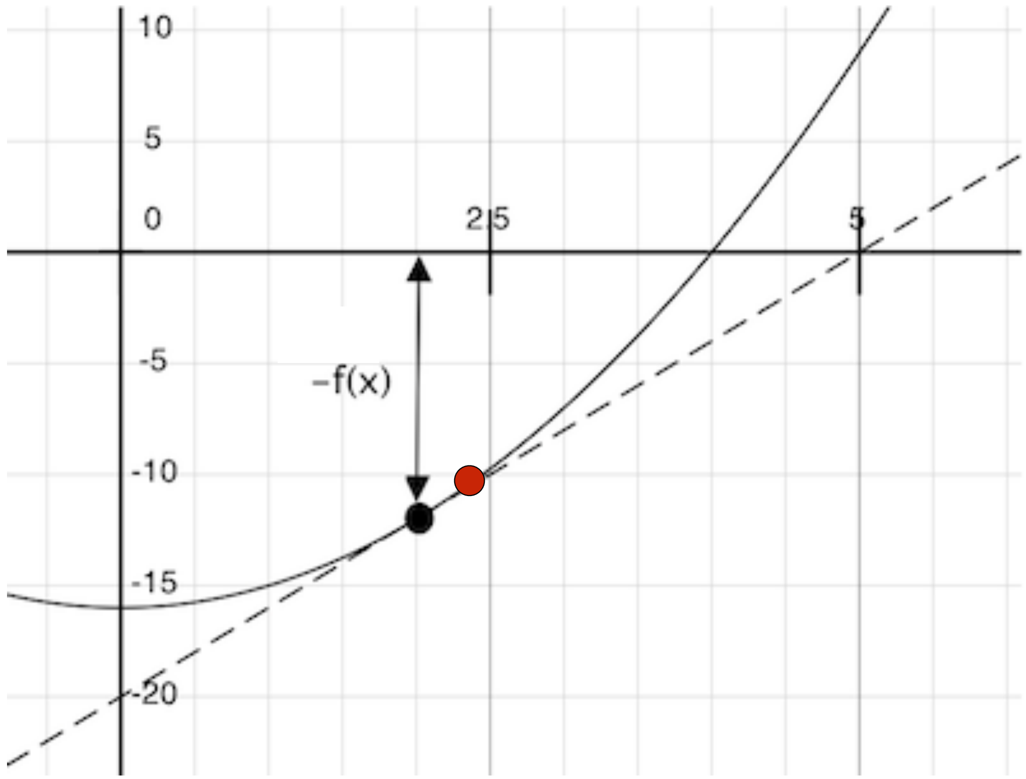
f(x) = x^2 – 16

f'(x) = 2x

f'(2) = 4

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$

$$f'(x) \approx \frac{f(x+a) - f(x)}{a}$$ (if a is small)



Approximate Differentiation

Differentiation can be performed symbolically or numerically

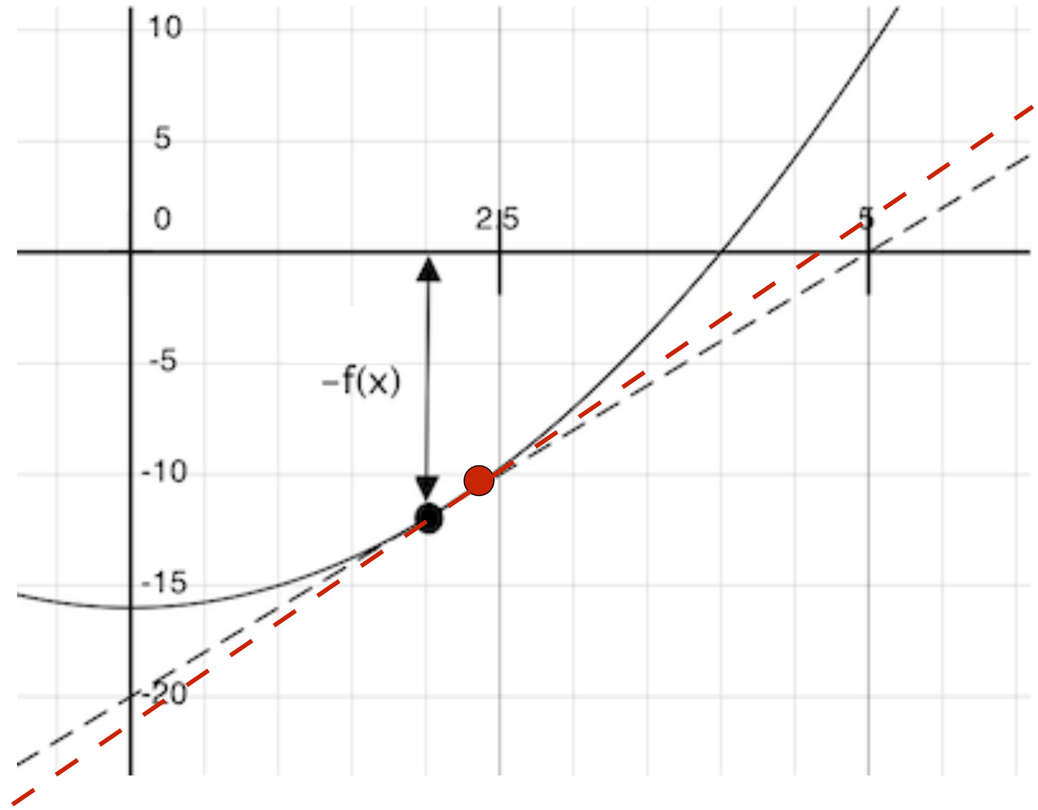
f(x) = $\boxed{x^2}$ – 16

f'(x) = $\boxed{2x}$

f'(2) = 4

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$

$$f'(x) \approx \frac{f(x+a) - f(x)}{a}$$ (if a is small)



Approximate Differentiation

Differentiation can be performed symbolically or numerically

f(x) = x^2 – 16

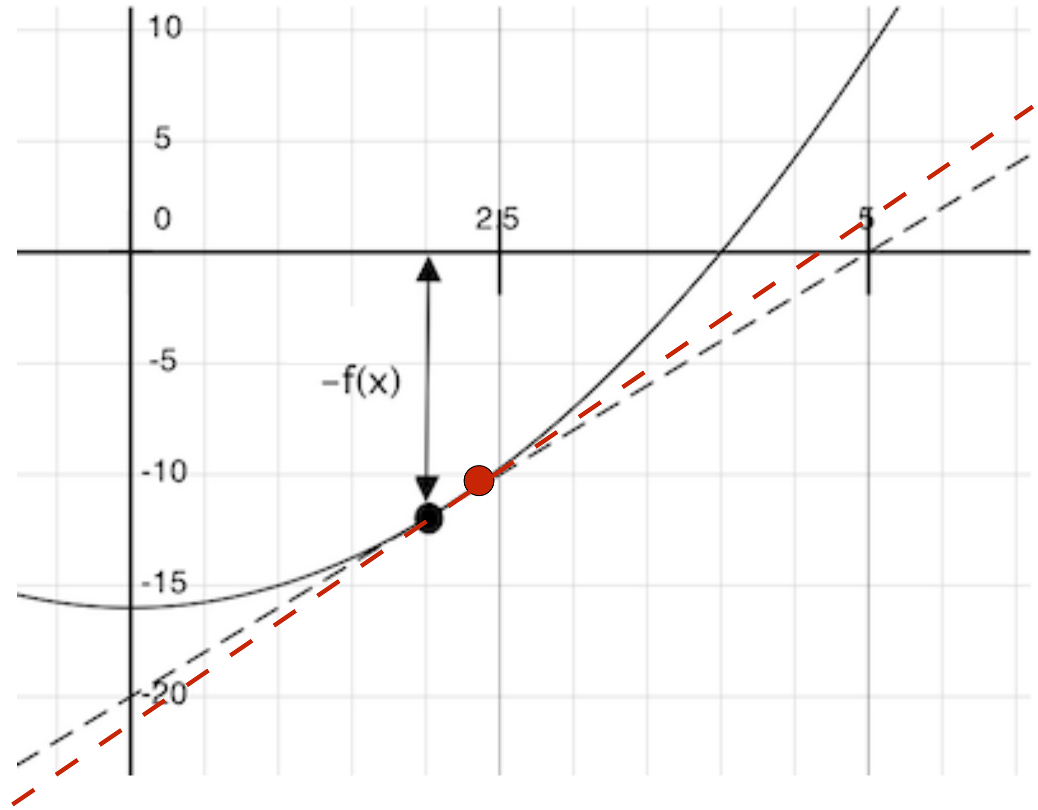
f'(x) = 2x

f'(2) = 4

$$f'(x) = \lim_{a \to 0} \frac{f(x+a) - f(x)}{a}$$

$$f'(x) \approx \frac{f(x+a) - f(x)}{a}$$ (if a is small)

(Demo)



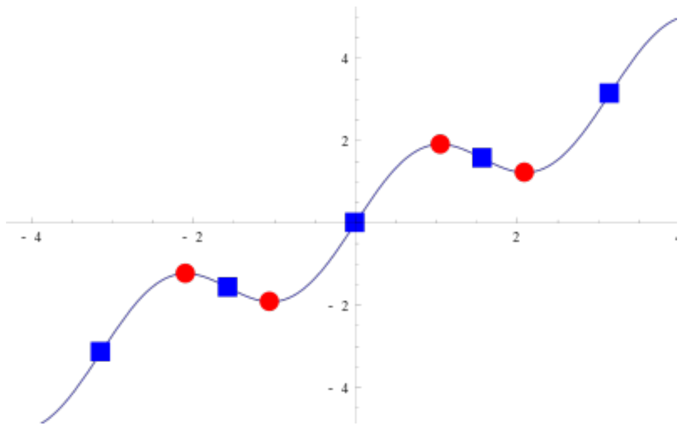
Critical Points and Inverses

Critical Points and Inverses

Maxima, minima, and inflection points of a differentiable function occur when the derivative is 0

Critical Points and Inverses

```
Maxima, minima, and inflection points of a differentiable function occur
when the derivative is 0
```

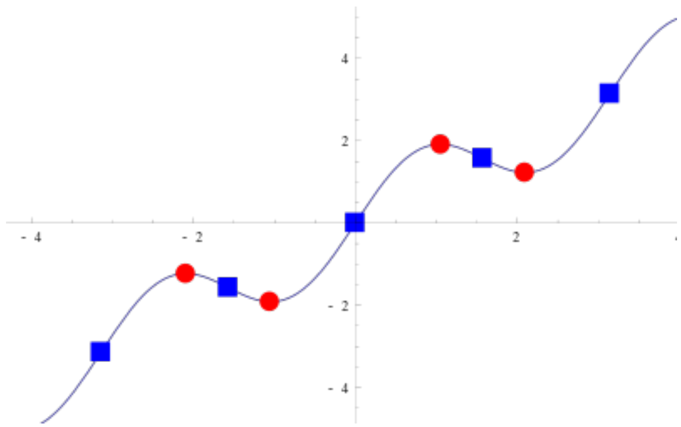


http://upload.wikimedia.org/wikipedia/commons/f/fd/Stationary_vs_inflection_pts.svg

Critical Points and Inverses

Maxima, minima, and inflection points of a differentiable function occur when the derivative is 0

```
derive = lambda f: lambda x: slope(f, x)
```



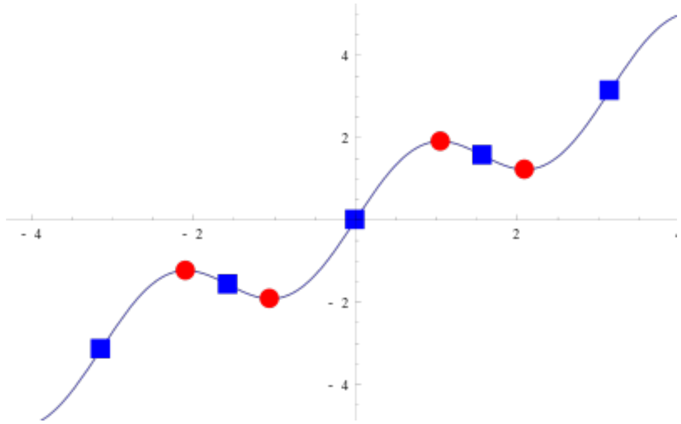
http://upload.wikimedia.org/wikipedia/commons/f/fd/Stationary_vs_inflection_pts.svg

Critical Points and Inverses

Maxima, minima, and inflection points of a differentiable function occur when the derivative is 0

```
derive = lambda f: lambda x: slope(f, x)
```

The inverse $f^{-1}(y)$ of a differentiable, one-to-one function computes the value x such that f(x) = y



http://upload.wikimedia.org/wikipedia/commons/f/fd/Stationary_vs_inflection_pts.svg

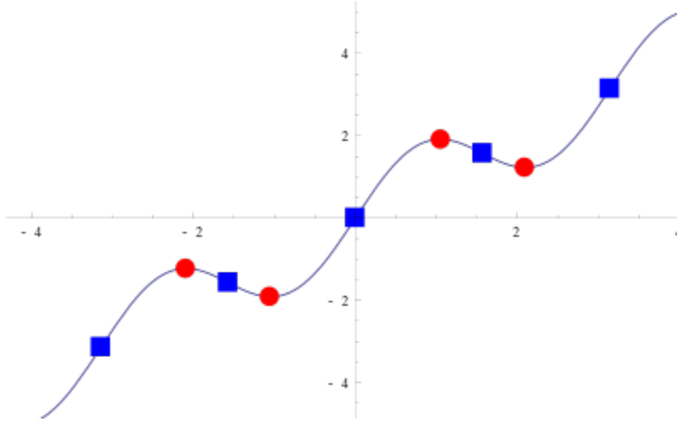
Critical Points and Inverses

Maxima, minima, and inflection points of a differentiable function occur when the derivative is 0

```
derive = lambda f: lambda x: slope(f, x)
```

The inverse $f^{-1}(y)$ of a differentiable, one-to-one function computes the value x such that f(x) = y

(Demo)



http://upload.wikimedia.org/wikipedia/commons/f/fd/Stationary_vs_inflection_pts.svg