Announcements

- Homework
  - I’m still working on the HW3 grading, sorry.
  - Be working on Homework 4!
    - Problem 3 should be doable at least up till the animation
    - Problem 2 doable after today
    - Pieces of Problem 1 doable after today
- No presenter for CS Tea tomorrow, and it is just out in the meeting area between offices
  - Still pizza, cookies, and goodies though!
  - 11:35am
- Polling: `rembold-class.ddns.net`

Review Question

When the final line of the code to the right is run, what type of variable is x?

A) integer

B) float

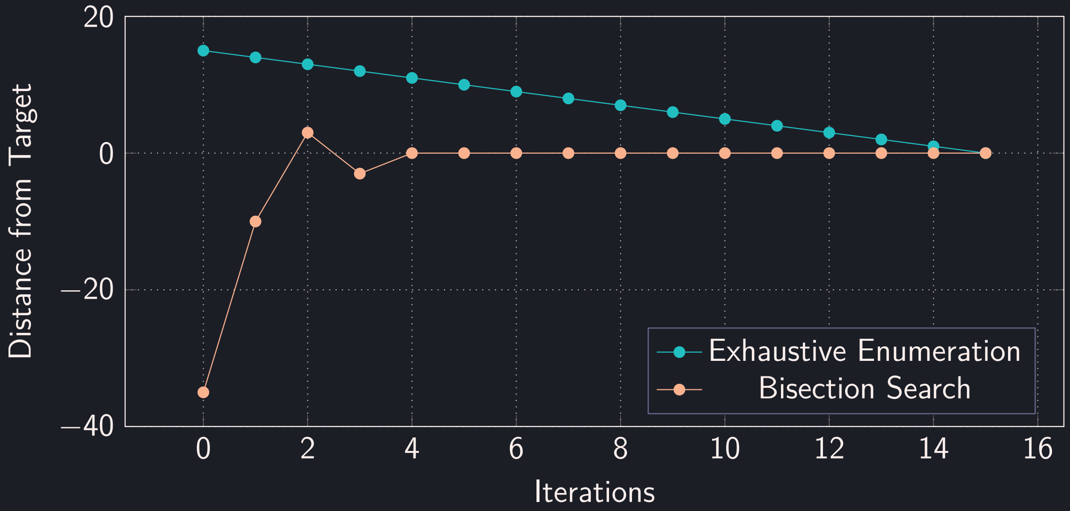
C) string

D) NoneType

```
def func(A):
    m = str(A)
    n = m * 10
    print(n)

y = 5.0
x = func(y)
print(type(x))
```

Convergence



Newton-Raphson

- Alternative method to find roots of polynomials

$$f(x) = 4x^3 + 2x + 10$$

- Requires an initial guess to work, but *does not require explicit bounds*!
- Revolves around the idea that, if g is a guess for the root of $f(x)$, then

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

  is an even better guess.

Mathematical Understanding

- Need to find the derivative (probably analytically for the time being.
    - For polynomials of form $f(x) = Ax^n$,

      $$f'(x) = Anx^{n-1}$$

    - Derivatives of sums add. If $f(x) = g(x) + h(x)$:

      $$f'(x) = g'(x) + h'(x)$$

    - Can always use an online calculator if your derivatives are shakey

Implementation Example

Example

Find the point where $8x^2 = 3$ with epsilon $= 0.01$.

Implementation Example

Example

Find the point where $8x^2 = 3$ with epsilon $= 0.01$.

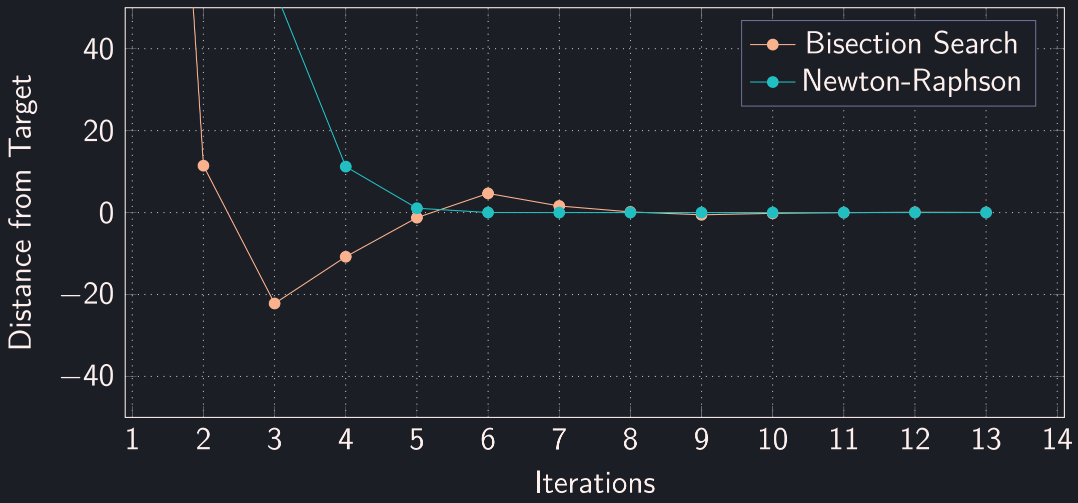
```
guess = 1
epsilon = 0.01
while abs(8*guess**2 - 3) > epsilon:
    val = 8*guess**2 - 3
    deriv = 16*guess
    guess = guess - val/deriv
print(guess)
```

Return of the Cube Root of 27

Example

Let's return again to computing the cube root of 27, but this time using the Newton-Raphson algorithm. We'll look for a solution that works for an epsilon of 0.01.

Comparison


Bisection Search
Newton-Raphson
Distance from Target
40
20
0
−20
−40
Iterations
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Be Careful

- Realize that any numeric method we have looked at will only get you a single solution!
- How do you work with equations that might have multiple possibilities?
  - You have to separate them out with separate ranges or guesses
  - Frequently might mean it would be useful to look at a quick plot of a function to understand near where certain roots exist
    - We'll learn to plot in Python more later in the semester, but for now you can always use something like Desmos or WolframAlpha
- Remember your expressions should be equal to 0
- Be careful of points with slope 0, as if you should hit them exactly your algorithm will break.

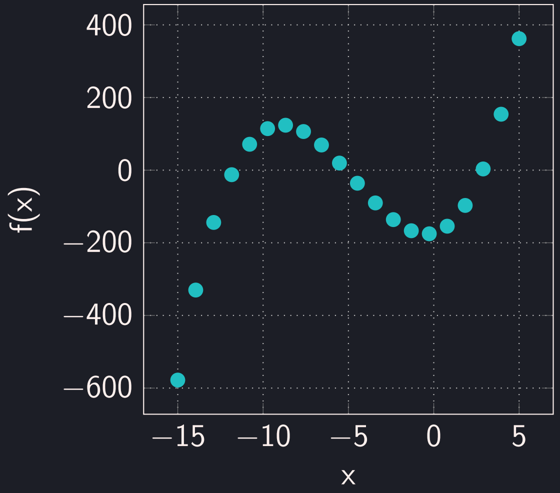
Multiroot Example

Example

Suppose you want to find when

$$(x+3)^3 + 5x^2 - 15x - 100 = 100$$

when epsilon $= 0.1$. Find all real solutions.



The Word is Key

- So far we have looked at a positional way to assign arguments to formal parameters

The Word is Key

- So far we have looked at a positional way to assign arguments to formal parameters
  - The first argument to the first parameter, the second to the second, etc

```python
def func(first, second, third):
    print(first, second, third)

func(1,2,3)
func(2,6,4)
```

The Word is Key

- So far we have looked at a positional way to assign arguments to formal parameters
    - The first argument to the first parameter, the second to the second, etc

```
def func(first, second, third):
    print(first, second, third)

func(1,2,3)
func(2,6,4)
```

- Can also write explicitly with keyword arguments!

The Word is Key

- So far we have looked at a positional way to assign arguments to formal parameters
  - The first argument to the first parameter, the second to the second, etc

```
def func(first, second, third):
    print(first, second, third)

func(1,2,3)
func(2,6,4)
```

- Can also write explicitly with keyword arguments!
  - If you do so, the position no longer matters

```
func(third=4, first=2, second=6)
```

The Word is Key

- So far we have looked at a positional way to assign arguments to formal parameters
  - The first argument to the first parameter, the second to the second, etc

```python
def func(first, second, third):
    print(first, second, third)

func(1,2,3)
func(2,6,4)
```

- Can also write explicitly with keyword arguments!
  - If you do so, the position no longer matters

```python
func(third=4, first=2, second=6)
```

- All keyword arguments must come after any positional arguments!

Default Slide Title

- Can also specify default values for a formal parameter

```
def func(name='Jed', age=34)
    print('My name is', name, 'and I am', age)
```

Default Slide Title

- Can also specify default values for a formal parameter

```python
def func(name='Jed', age=34)
    print('My name is', name, 'and I am', age)
```

- You then don't need to always provide that actual parameter

Default Slide Title

- Can also specify default values for a formal parameter

```python
def func(name='Jed', age=34)
    print('My name is', name, 'and I am', age)
```

- You then don't need to always provide that actual parameter
- If setting any parameters out of order though, you must indicate them through keywords.

```python
func()
func('Bob',25)
func('Larry')
func(age=68)
```