1 Questions to be handed in for project 6b:

Read about this material here: Symbolic math in Julia.

Begin by loading our package for plotting *and* a new package that allows symbolic math:

```
using Gadfly            # ignore any warnings
using SymPy
```

1.0.1 Quick background Read the notes for more detail

There are many computer algebra systems (CAS) that do symbolic math. Most students are familiar with Wolfram's alpha site, as many homework problems can be easily done there. That site uses Wolfram's `Mathematica` program, though the interface relaxes greatly the syntax of that program. Other CASs include `Maple` and `Sage`. Both `Mathematica` and `Maple` are available for free to CUNY students and `Sage` is free to everyone, as it is open source.

On top of these, there is `SymPy`, an add-on to the popular `Python` programming language. Julia's `SymPy` package interfaces with `Python's SymPy` package in a fairly easy to use manner.

The first thing one does is create some symbolic variables:

```
x, y, z, h = Sym("x", "y", "z", "h")
```

That command creates symbolic variables that (more or less) magically interact with `Julia` functions. So for example, we can create a symbolic expression as follows:

```
p = 16x^2 - 96x + 128
```

Note a subtle but *big* difference: we did not define a function, rather `p` is an expression. (A function would be `p(x) = ...`). Symbolic expressions print slightly differently than functions, so this can be a clue. Also note that we *could* have defined a function, then evaluated it on the symbolic value of x:

```
q(x) = 16x^2 - 96x + 128    # a function
q(2)                # used as any other function call
q(x)                # evaluating with a symbolic value gives expression
```

What is so great about symbolic expressions? Well, they can be manipulated symbolically!

For example to *factor* the polynomial `p` we have the `factor` function:

```
factor(p)
```

(`factor` applied to a symbolic expression tries to factor as a polynomial, say; whereas `factor` applied to an integer tries to find prime factors. The `factor` function is generic, so can have different implementations depending on the type of its argument.)

To solve for `p=0` we have `solve`:

```
solve(p, x)         # or just solve(p) as x can be implied
```

The two answers are 2 and 4, as one could read from the factorization of `p`. The `solve` function tries to solve when the expression is equal to 0. To solve something of the type $g(x) = h(x)$, use $g(x) - h(x) = 0$, as is done with `fzero`, our numeric solver.

Symbolic expressions are not functions, so there isn't an easy way to replace the `x` with a concrete value, like 2. To replace a symbolic value with a number use the `replace` function (`replace(ex, x, value)`). For example here is how one can evaluate a polynomial at $x = 2$:

```
replace(x^2 - 2x + 3, x, 2)
```

1.0.2 Questions, Working with algebraic expressions

Algebra functions include `expand` to expand a polynomial expression, `simplify` to simplify algebraically an expression, `together` to combine expressions, and `factor` to factor expressions.

- Factor the polynomial $p(x) = -2x^4 - x^3 + 3x^2$

- Find the zeros and any vertical asymptotes of the rational function

$$f(x) = \frac{x^3 + 3x^2}{x^2 - x - 6}$$

(Use `factor` to get the factors, then read these values off.)

- Factor the polynomial $p(x) = 4x^4 + 2x^3 - 2x^2 - 3x - 1$. How many roots are there? How many real roots are there?

- Solve the equation $x^4 - 8x^2 + 8 = 0$. How many real roots are there?

- Simplify

$$\frac{\frac{1}{x+h} - \frac{1}{x}}{h}$$

From your simplified answer, what would be the value if $h = 0$?

1.0.3 Performing limits

The `limit` function from `SymPy` implements Gruntz's algorithm to find symbolic derivatives. It does not have the issues with floating point that a numeric approach does.

The basic form is `limit(expr, x, c)`, where `x` is the symbolic variable and `c` is where the limit is being taken. Optionally one can include `dir="+"` or `dir="-"` to find limits from the right or left. For example, the right limit at $c = 0$ of x^x is given by:

```
limit(x^x, x, 0, dir="+")
```

- Symbolically find

$$\lim_{x \to 0} \frac{3^x - 1}{x}$$

- Symbolically find

$$\lim_{x \to 0+} (1 + 3x)^{1/x}$$

- Symbolically find

$$\lim_{x \to 1} \frac{x^n - 1}{x^m - 1}$$

You can define m and n as symbols and your answer will include them:

```
m,n = Sym("m", "n")
```

- Symbolically find

$$\lim_{x \to \infty} (\frac{x}{x+1})^x$$

Limits at infinity just need `c=oo` (oh-oh, not zero-zero).

- What is the value of this expression and does it make sense?

```
f(x) = sin(x)
ex = (f(x + h) - f(x)) / h
limit(ex, h, 0)
```

1.0.4 Finding derivatives

`SymPy` provides the `diff` function for finding derivatives: `diff(ex, x)` will find the derivative in `x`, whereas `diff(ex, x, 2)` will find the second derivative. For example, here we see the chain rule in action:

```
f(x) = sin(exp(x))
diff(f(x), x)
```

(Just to be clear, the `D` function from the `Roots` package was `D(f)` which creates a function. Here we use `f(x)` which creates a symbolic expression. It is an important distinction, `diff(f,x)` will not work, as desired.)

- What is the derivative of $f(x) = \sin(x)/(\tan^{-1}(x) + \tan^2(x))$?

- What is the derivative of

$$f(x) = \frac{1}{\sqrt{(2\pi)}} e^{-x^2/2}?$$

- Find the second derivative of $f(x) = \tan^{-1}(x)$.

- Find the 10th derivative of $f(x) = xe^{-x}$.

- Does this limit give the first derivative? Check that it does or doesn't for $f(x) = \sin(x)$.

$$\lim_{h \to 0} \frac{f(x+h) - f(x-h)}{h}.$$

- Does this limit give the second derivative? Check that it does or doesn't for $f(x) = \sin(x)$.

$$\lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

- Find the critical points of $f(x) = 3x^4 32x^3 + 114x^2 144x + 2$ by solving for when the derivative is 0.

- Find the inflection points of $f(x) = e^{-x^2}$

- For the function $\sin(x)$ over the interval $[0, \pi/2]$ find a point c such that the slope of the tangent line at c is equal to the slope of the secant line from 0 to $\pi/2$.

1.0.5 Graphing expressions

Plotting expression(s) requires the following to be evaluated first:

```
using Gadfly
Gadfly.plot(ex::Sym, args...) = plot(convert(Function, ex), args...)
Gadfly.plot{T<:Sym}(exs::Vector{T}, args...) = plot(map(ex -> convert(Function, ex), exs), args...)
```

Then expressions can be plotted, in a similar manner as functions. For example:

```
plot([sin(x), diff(sin(x), x)], 0, 2pi)
```

- Graphically solve for when $f(x) = e^x$ and $g(x) = 10 + 5x$ intersect for $x > 0$.

- For the function $f(x) = e^x$. Over the interval $[0, 3]$, plot both $f(x)$ and the expression

$$f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2.$$

Where $n! = n \cdot (n-1) \cdots 2 \cdot 1$, and can be evaluated with `factorial(2)`. Do the two mostly agree on this interval?

(One tedious aspect of finding $f'(a)$ is that it is done in two steps like `replace(diff(f, x), x, a)` (find the derivative in `x`, the replace `x` with `a`). The following function can shorten the above so that `D(f,k)(a)` works on symbolic expression, as it did for functions with `D` from the `Roots` package.)

```
D(f, k=1) = a -> replace(diff(f(x), x, k), x, a)
```