```
In [10]: from __future__ import print_function
%matplotlib inline
#import ganymede
#ganymede.configure('uav.beaver.works')
import matplotlib.pyplot as plt
import numpy as np
import sympy as sym
from IPython.display import YouTubeVideo, HTML
sym.init_printing(use_latex = "mathjax")
```

```
In [8]: #ganymede.name('Krisha')
def check(p):
    #ganymede.update(p,True)

check(0)
```

```
  Cell In[8], line 5
    check(0)
    ^
IndentationError: expected an indented block after function definition on line 2
```

Enter your name below and run the cell:

Individual cells can be run with `Ctrl` + `Enter`

```
In [ ]: YouTubeVideo('9vKqVkMQHKk', width=560, height=315) # Video by http://www.3blue1brown.com/
```

```
In [ ]: YouTubeVideo('bRZmfc1YFsQ', width=560, height=315) #Note: All Khan Academy content is available for free at khanacademy.org
```

Power Rule

The derivative of x^n is nx^{n-1}

Read more

Other derivative rules

```
In [12]: # Creating algebraic symbols
x = sym.symbols('x')
x
```

Out[12]: x

```
In [13]: x = sym.symbols('x')
expr = x ** 2
expr
```

Out[13]: x^2

```
In [14]: sym.Derivative(expr) # does not actually compute the derivative
```

Out[14]: $\frac{d}{dx}x^2$

```
In [15]: sym.Derivative(expr).doit()
```

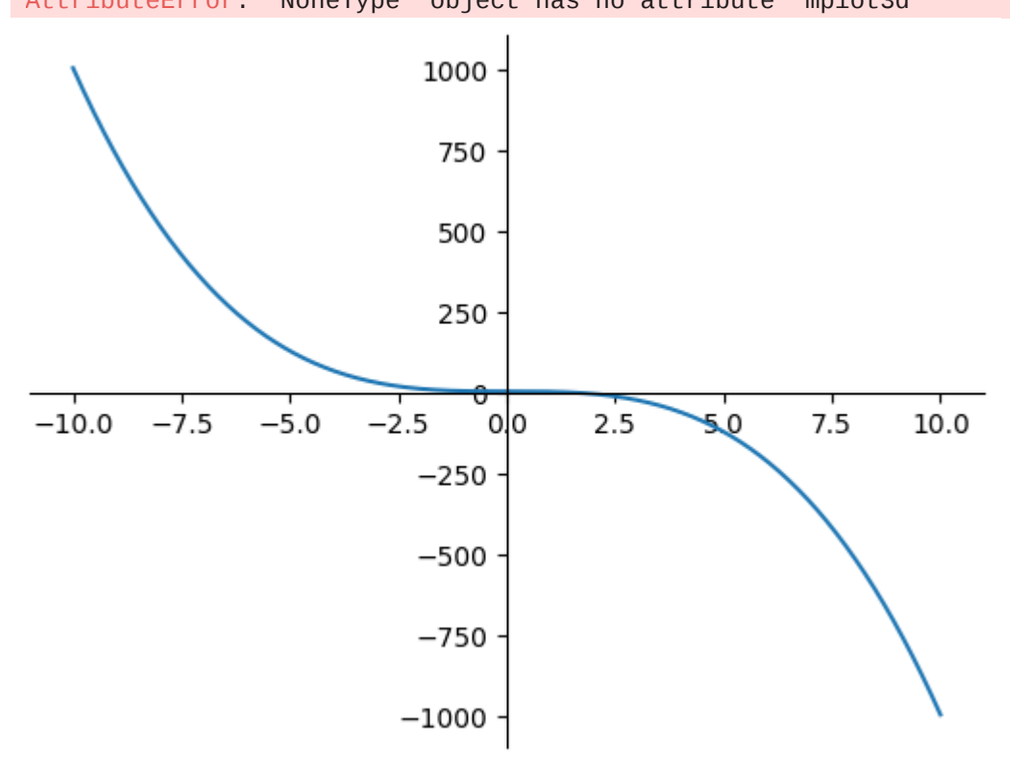
Out[15]: $2x$

```
In [17]: sym.diff(expr) #equivalent to doit()
```

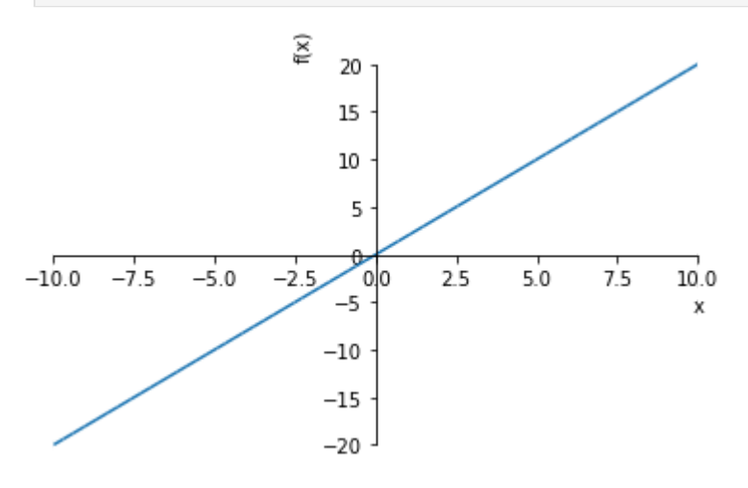
Out[17]: $2x$

```
In [31]: sym.plot(expr);
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[31], line 1
----> 1 sym.plot(expr);

File ~/.local/lib/python3.10/site-packages/sympy/plotting/plot.py:419, in plot(show, *args, **kwargs)
    417 plots = plot_factory(*series, **kwargs)
    418 if show:
--> 419     plots.show()
    420 return plots

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:303, in MatplotlibBackend.show(self)
    302 def show(self):
--> 303     self.process_series()
    304     #TODO after fixing https://github.com/ipython/ipython/issues/1255
    305     # you can uncomment the next line and remove the pyplot.show() call
    306     #self.fig.show()
    307     if base_backend._show:

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:300, in MatplotlibBackend.process_series(self)
    295 """
    296 Iterates over every ``Plot`` object and further calls
    297 _process_series()
    298 """
    299 self._create_figure()
--> 300 self._process_series(self._series, self.ax)

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:210, in MatplotlibBackend._process_series(self, series, ax)
    204     else:
    205         raise NotImplementedError(
    206             '{} is not supported in the SymPy plotting module '
    207             'with matplotlib backend. Please report this issue.'
    208             .format(ax))
--> 210 Axes3D = mpl_toolkits.mplot3d.Axes3D
    211 if not isinstance(ax, Axes3D):
    212     ax.autoscale_view(
    213         scalex=ax.get_autoscalex_on(),
    214         scaley=ax.get_autoscaley_on())

AttributeError: 'NoneType' object has no attribute 'mplot3d'
```

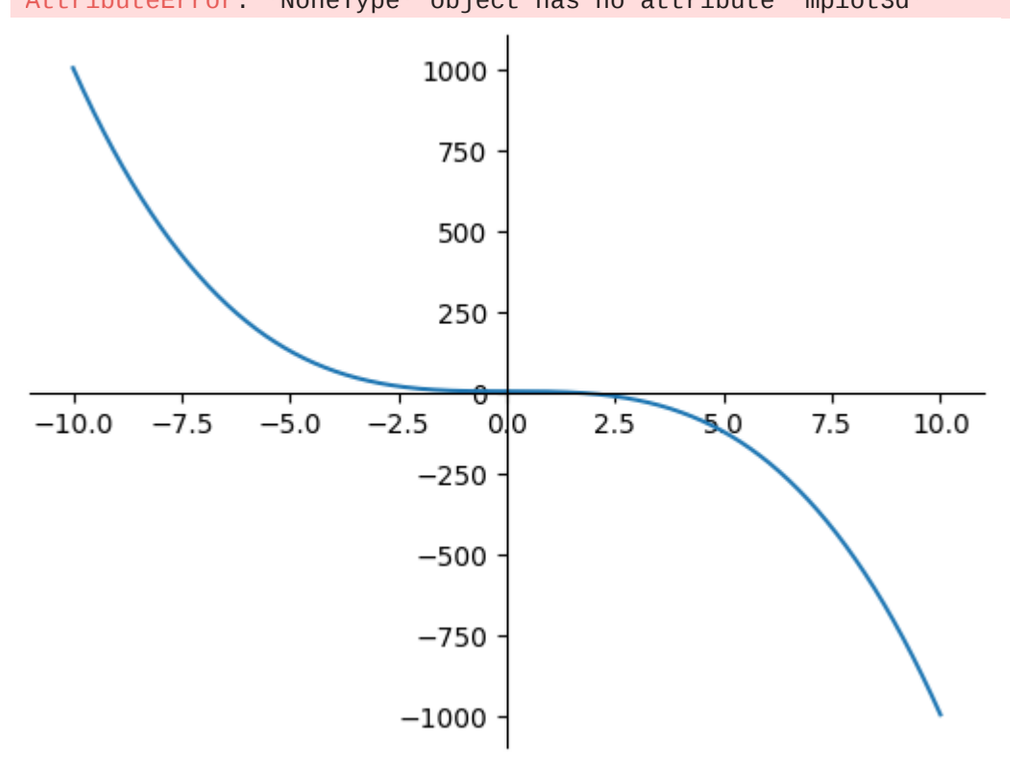



```
In [10]: sym.plot(sym.diff(expr));
```




```
In [19]: x = sym.symbols('x')
expr = -x ** 3 + 2

sym.plot(expr, xlim=(-2, 2), ylim=(-10, 10));
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[19], line 4
      1 x = sym.symbols('x')
      2 expr = -x ** 3 + 2
----> 4 sym.plot(expr, xlim=(-2, 2), ylim=(-10, 10));

File ~/.local/lib/python3.10/site-packages/sympy/plotting/plot.py:419, in plot(show, *args, **kwargs)
    417 plots = plot_factory(*series, **kwargs)
    418 if show:
--> 419     plots.show()
    420 return plots

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:303, in MatplotlibBackend.show(self)
    302 def show(self):
--> 303     self.process_series()
    304     #TODO after fixing https://github.com/ipython/ipython/issues/1255
    305     # you can uncomment the next line and remove the pyplot.show() call
    306     #self.fig.show()
    307     if base_backend._show:

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:300, in MatplotlibBackend.process_series(self)
    295 """
    296 Iterates over every ``Plot`` object and further calls
    297 _process_series()
    298 """
    299 self._create_figure()
--> 300 self._process_series(self._series, self.ax)

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:210, in MatplotlibBackend._process_series(self, series, ax)
    204     else:
    205         raise NotImplementedError(
    206             '{} is not supported in the SymPy plotting module '
    207             'with matplotlib backend. Please report this issue.'
    208             .format(ax))
--> 210 Axes3D = mpl_toolkits.mplot3d.Axes3D
    211 if not isinstance(ax, Axes3D):
    212     ax.autoscale_view(
    213         scalex=ax.get_autoscalex_on(),
    214         scaley=ax.get_autoscaley_on())

AttributeError: 'NoneType' object has no attribute 'mplot3d'
```




```
In [20]: sym.Derivative(expr)
```

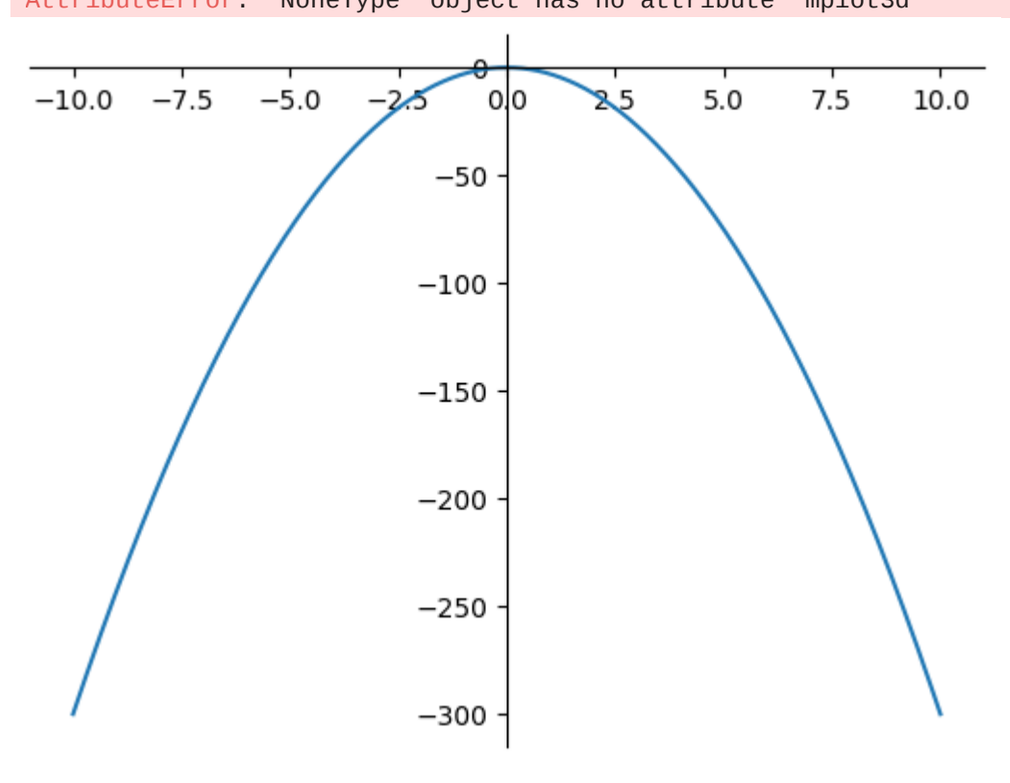
Out[20]: $\frac{d}{dx}\left(2 - x^3\right)$

```
In [21]: sym.Derivative(expr).doit()
```

Out[21]: $-3x^2$

```
In [22]: sym.plot(sym.diff(expr));
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[22], line 1
----> 1 sym.plot(sym.diff(expr));

File ~/.local/lib/python3.10/site-packages/sympy/plotting/plot.py:419, in plot(show, *args, **kwargs)
    417 plots = plot_factory(*series, **kwargs)
    418 if show:
--> 419     plots.show()
    420 return plots

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:303, in MatplotlibBackend.show(self)
    302 def show(self):
--> 303     self.process_series()
    304     #TODO after fixing https://github.com/ipython/ipython/issues/1255
    305     # you can uncomment the next line and remove the pyplot.show() call
    306     #self.fig.show()
    307     if base_backend._show:

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:300, in MatplotlibBackend.process_series(self)
    295 """
    296 Iterates over every ``Plot`` object and further calls
    297 _process_series()
    298 """
    299 self._create_figure()
--> 300 self._process_series(self._series, self.ax)

File ~/.local/lib/python3.10/site-packages/sympy/plotting/backends/matplotlibbackend/matplotlib.py:210, in MatplotlibBackend._process_series(self, series, ax)
    204     else:
    205         raise NotImplementedError(
    206             '{} is not supported in the SymPy plotting module '
    207             'with matplotlib backend. Please report this issue.'
    208             .format(ax))
--> 210 Axes3D = mpl_toolkits.mplot3d.Axes3D
    211 if not isinstance(ax, Axes3D):
    212     ax.autoscale_view(
    213         scalex=ax.get_autoscalex_on(),
    214         scaley=ax.get_autoscaley_on())

AttributeError: 'NoneType' object has no attribute 'mplot3d'
```



Now, let's generate a fake one-dimensional signal:

```
In [7]: ys = np.array([0, 1, 0, 1, 2, 0, 2, 3, 2, 1, 2, 102, 108,  95, 100,  98,  99, 104, 110, 103, 100,  96, 102, 101])

fig,ax = plt.subplots()
ax.plot([i for i in range(len(ys))], ys);
#check(1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 ys = np.array([0, 1, 0, 1, 2, 0, 2, 3, 2, 1, 2, 102, 108,  95, 100,  98,  99, 104, 110, 103, 100,  96, 102, 101])
      3 fig,ax = plt.subplots()
      4 ax.plot([i for i in range(len(ys))], ys);

NameError: name 'np' is not defined
```

Next, let's look at small chunks of our fake signal:

```
In [25]: chunks = np.split(ys, len(ys)//2)
print(chunks)
#check(2)
```

```
[array([0, 1]), array([0, 1]), array([2, 0]), array([2, 3]), array([2, 1]), array([  2, 102]), array([108,  95]), array([100,  98]), array([ 99, 104]), array([110, 103]), array([100,  96]), array([102, 101])]
```

Question: Which one of these chunks would you say is the most "interesting"? When the graph spikes up, array([2,102]) **Question** If we always divide up the signal as we did above, will we always find something "interesting"? no, the signal could be slowly rising

Convolutions

Derivatives and convolutions are one technique to help us tackle the above problem.

First, you'll need to generate windows into the signal. Write a function that can generate windows with a user-supplied windowsize, and print them out.

An example signal with 3 window sizes is shown below. Your output does not need to replicate the formatting shown, but they should produce the same windows. E.g., given an input signal of `[10,20,30]` and a `windowsize=2`, your function should return `[[10,20], [20,30]]`.

A windowsize of 1:

```
signal:
        0   1   0   2   1   0   1 101 100  98 102 101
 0:     0
 1: ____    1
 2: ________    0
 3: ____________    2
 4: ________________    1
 5: ____________________    0
 6: ________________________    1
 7: ____________________________  101
 8: ________________________________  100
 9: ____________________________________   98
10: ________________________________________  102
11: ____________________________________________  101

         :::::::::::::::::::::::::::::::::::::::::::::::::::

i:    0 |  i + windowsize:     1  |  window:  [   0]
i:    1 |  i + windowsize:     2  |  window:  [   1]
i:    2 |  i + windowsize:     3  |  window:  [   0]
i:    3 |  i + windowsize:     4  |  window:  [   2]
i:    4 |  i + windowsize:     5  |  window:  [   1]
i:    5 |  i + windowsize:     6  |  window:  [   0]
i:    6 |  i + windowsize:     7  |  window:  [   1]
i:    7 |  i + windowsize:     8  |  window:  [ 101]
i:    8 |  i + windowsize:     9  |  window:  [ 100]
i:    9 |  i + windowsize:    10  |  window:  [  98]
i:   10 |  i + windowsize:    11  |  window:  [ 102]
i:   11 |  i + windowsize:    12  |  window:  [ 101]
```

A windowsize of 2:

```
signal:
        0   1   0   2   1   0   1 101 100  98 102 101
 0:     0   1
 1: ____    1   0
 2: ________    0   2
 3: ____________    2   1
 4: ________________    1   0
 5: ____________________    0   1
 6: ________________________    1 101
 7: ____________________________  101 100
 8: ________________________________  100  98
 9: ____________________________________   98 102
10: ________________________________________  102 101

         :::::::::::::::::::::::::::::::::::::::::::::::::::

i:    0 |  i + windowsize:     2  |  window:  [   0,   1]
i:    1 |  i + windowsize:     3  |  window:  [   1,   0]
i:    2 |  i + windowsize:     4  |  window:  [   0,   2]
i:    3 |  i + windowsize:     5  |  window:  [   2,   1]
i:    4 |  i + windowsize:     6  |  window:  [   1,   0]
i:    5 |  i + windowsize:     7  |  window:  [   0,   1]
i:    6 |  i + windowsize:     8  |  window:  [   1, 101]
i:    7 |  i + windowsize:     9  |  window:  [ 101, 100]
i:    8 |  i + windowsize:    10  |  window:  [ 100,  98]
i:    9 |  i + windowsize:    11  |  window:  [  98, 102]
i:   10 |  i + windowsize:    12  |  window:  [ 102, 101]
```

A windowsize of 3

```
signal:
        0   1   0   2   1   0   1 101 100  98 102 101
 0:     0   1   0
 1: ____    1   0   2
 2: ________    0   2   1
 3: ____________    2   1   0
 4: ________________    1   0   1
 5: ____________________    0   1 101
 6: ________________________    1 101 100
 7: ____________________________  101 100  98
 8: ________________________________  100  98 102
 9: ____________________________________   98 102 101

         :::::::::::::::::::::::::::::::::::::::::::::::::::

i:    0 |  i + windowsize:     3  |  window:  [   0,   1,   0]
i:    1 |  i + windowsize:     4  |  window:  [   1,   0,   2]
i:    2 |  i + windowsize:     5  |  window:  [   0,   2,   1]
i:    3 |  i + windowsize:     6  |  window:  [   2,   1,   0]
i:    4 |  i + windowsize:     7  |  window:  [   1,   0,   1]
i:    5 |  i + windowsize:     8  |  window:  [   0,   1, 101]
i:    6 |  i + windowsize:     9  |  window:  [   1, 101, 100]
i:    7 |  i + windowsize:    10  |  window:  [ 101, 100,  98]
i:    8 |  i + windowsize:    11  |  window:  [ 100,  98, 102]
i:    9 |  i + windowsize:    12  |  window:  [  98, 102, 101]
```

The below resources may be helpful::

List Comprehensions

https://www.pythonlikeyoumeanit.com/Module2_EssentialsOfPython/Generators_and_Comprehensions.html#List-&-Tuple-Comprehensions

Numpy indexing with slices

http://www.pythonlikeyoumeanit.com/Module3_IntroducingNumpy/AccessingDataAlongMultipleDimensions.html#Slice-Indexing

Formatting numbers in python

https://pyformat.info/#number

input: `'{:4d}'.format(42)`

output: `_` `_` `4` `2`

input: `'{:06.2f}'.format(3.141592653589793)`

output: `003.14`

String concatenation

```
>>> print('a' + 'b' + 'c')
abc
>>> print(''.join(['a', 'b', 'c']))
abc
>>> print(''.join(['a', 'b', 'c']))
a,b,c
```

```
In [3]: def make_windows(sequence, windowsize):
    windows = []
    for i in range(len(sequence) - windowsize +1):
        window = sequence [i:i + windowsize]
        windows.append(window)

    return windows
```

```
In [6]: series = [0, 1, 0, 2, 1, 0, 1, 101, 100, 98, 102, 101]


make_windows(sequence=series, windowsize=1)
make_windows(sequence=series, windowsize=2)
make_windows(sequence=series, windowsize=3)

#check(3)
```

```
Out[6]: [[0, 1, 0],
 [1, 0, 2],
 [0, 2, 1],
 [2, 1, 0],
 [1, 0, 1],
 [0, 1, 101],
 [1, 101, 100],
 [101, 100, 98],
 [100, 98, 102],
 [98, 102, 101]]
```

When you are done:

Generate some example outputs in this notebook.

1. Double-check that you filled in your name at the top of the notebook!

2. Click `File` -> `Export Notebook As` -> `PDF`
3. Email the PDF to `YOURTEAMNAME@beaver.works`