In [1]:
```
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")
```

```
---------------------------------------------------------------------------
-
ModuleNotFoundError                       Traceback (most recent call las
t)
Cell In[1], line 3
      1 from __future__ import print_function
      2 get_ipython().run_line_magic('matplotlib', 'inline')
----> 3 import ganymede
      4 ganymede.configure('uav.beaver.works')
      5 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'ganymede'
```

Enter your name below and run the cell:

Individual cells can be run with `Ctrl + Enter`

In [ ]:
```
ganymede.name('YOUR NAME HERE')
def check(p):
    ganymede.update(p,True)
check(0)
```

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

In [ ]:
```
YouTubeVideo('bRZmfc1YFsQ', width=560, height=315) #Note: All Khan Academ
```

Power Rule

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

Read more

Other derivative rules

In [4]:
```
# Creating algebraic symbols
x = sym.symbols('x')
x
```

Out[4]:

$$x$$

In [5]:
```
x = sym.symbols('x')
expr = x ** 2
expr
```

Out[5]:

$$x^2$$

In [6]:
```
sym.Derivative(expr) # does not actually compute the derivative
```

Out[6]:

$$\frac{d}{dx} x^2$$

In [7]:
```
sym.Derivative(expr).doit()
```

Out[7]:

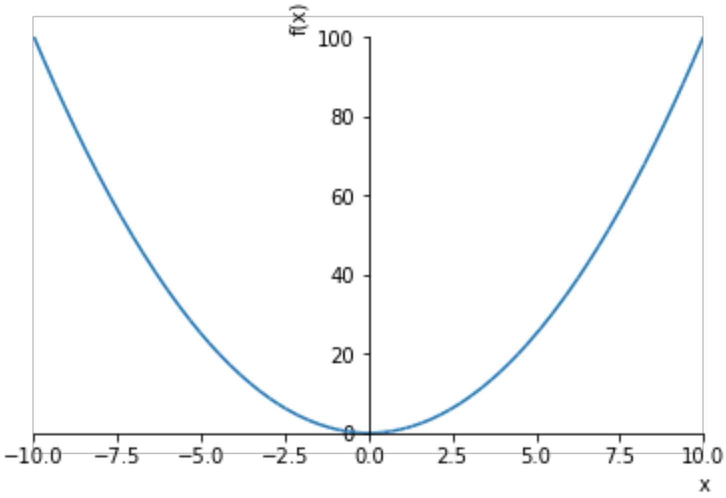
$$2x$$

In [8]:
```
sym.diff(expr) #equivalent to doit()
```

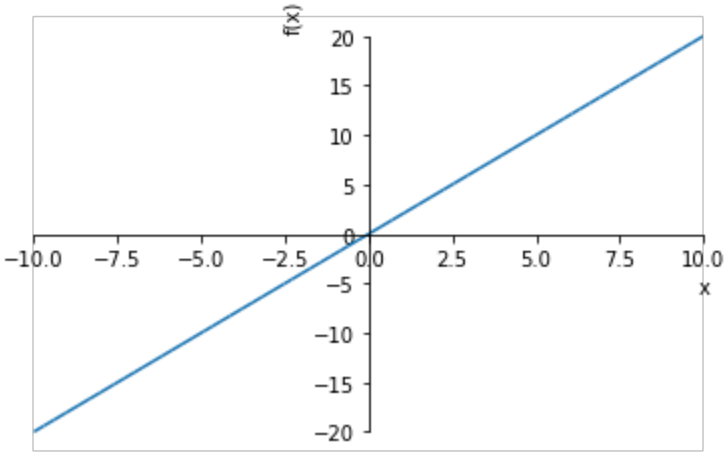
Out[8]:

$$2x$$

In [9]:
```
sym.plot(expr);
```



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



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

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

In [12]:
```
sym.Derivative(expr)
```

Out[12]:

$$\frac{d}{dx}\left(-x^3 + 2\right)$$

In [13]:
```
sym.Derivative(expr).doit()
```

Out[13]:

$$-3x^2$$

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

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

In [ ]:
```
ys = np.array([0, 1, 0, 1, 2, 0, 2, 3, 2, 1, 2, 102, 108,  95, 100,  98,

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

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

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

Question: Which one of these chunks would you say is the most "interesting"? The array([2, 102]) is the most interesting because it has the largest change in values and is an anomaly.

Question If we always divide up the signal as we did above, will we always find something "interesting"? We won't always find something interesting. The presence of interesting features or anomalies in the data depends on the nature of the signal. If the signal is mostly uniform or lacks significant changes, dividing it up may not reveal any notable features.

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 [1]:

```
def make_windows(sequence, windowsize):
    list = []
    for i in range(len(sequence)-windowsize+1):
        temp = []
        for j in range(windowsize):
            temp.append(sequence[i+j])
        list.append(temp)
    return list
    raise NotImplementedError # TODO
```

In [2]:
```
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)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 8
      5 make_windows(sequence=series, windowsize=2)
      6 make_windows(sequence=series, windowsize=3)
----> 8 check(3)

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

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`