In [ ]:

```
%pylab inline
```

Today

- List comprehensions
- Mental models (math fundamentals)
- Algorithm analysis
- Big-O notation

Modifying a list

In [2]:

```
my_list = [1, 3, 5, 7, 9]
```

In [3]:

```
new_list = []
for element in my_list:
    new_element = element + 1
    new_list.append(new_element)
```

In [4]:

```
print(new_list)
```

```
[2, 4, 6, 8, 10]
```

In [5]:

```
my_list = [1, 3, 5, 7, 9]
new_list = []
for element in my_list:
    new_element = element * 2
    new_list.append(new_element)
print(new_list)
```

```
[2, 6, 10, 14, 18]
```

What is the difference between these two programs?

```
my_list = [1, 3, 5, 7, 9]
new_list = []
for element in my_list:
    new_element = element + 1
    new_list.append(new_element)
```

```
my_list = [1, 3, 5, 7, 9]
new_list = []
for element in my_list:
    new_element = element * 2
    new_list.append(new_element)
```

Introducing list comprehensions

In pseudo-code we want something like:

```
# For every element in a list:
#     Perform some operation on the element
# Give me a new list
```

```
# Perform operation on x every time we have an x in a list
```

```
# Perform operation on x for x in my_list
```

```
# x + 1 for x in my_list
```

In [ ]:

```
my_list = [1, 3, 5, 7, 9]
new_list = []
for x in my_list:
    new_element = x + 1
    new_list.append(new_element)
```

In [6]:

```
my_list = [1, 3, 5, 7, 9]
new_list = [x + 1 for x in my_list]
```

In [7]:

```python
print(new_list)
```

```
[2, 4, 6, 8, 10]
```

In [11]:

```python
my_list = [1, 3, 5, 7, 9]
new_list = [x ** 2 for x in my_list]
print(new_list)
```

```
[1, 9, 25, 49, 81]
```

List comprehension exercise:

This is the pseudo-code:

```python
# x + 1 for x in my_list
```

Using the list `[1, 3, 5, 7, 9]`, do the following:

- Increase all elements by 3
- Subtract 10 from all elements
- Multiply all elements by themselves

Understanding the world through models

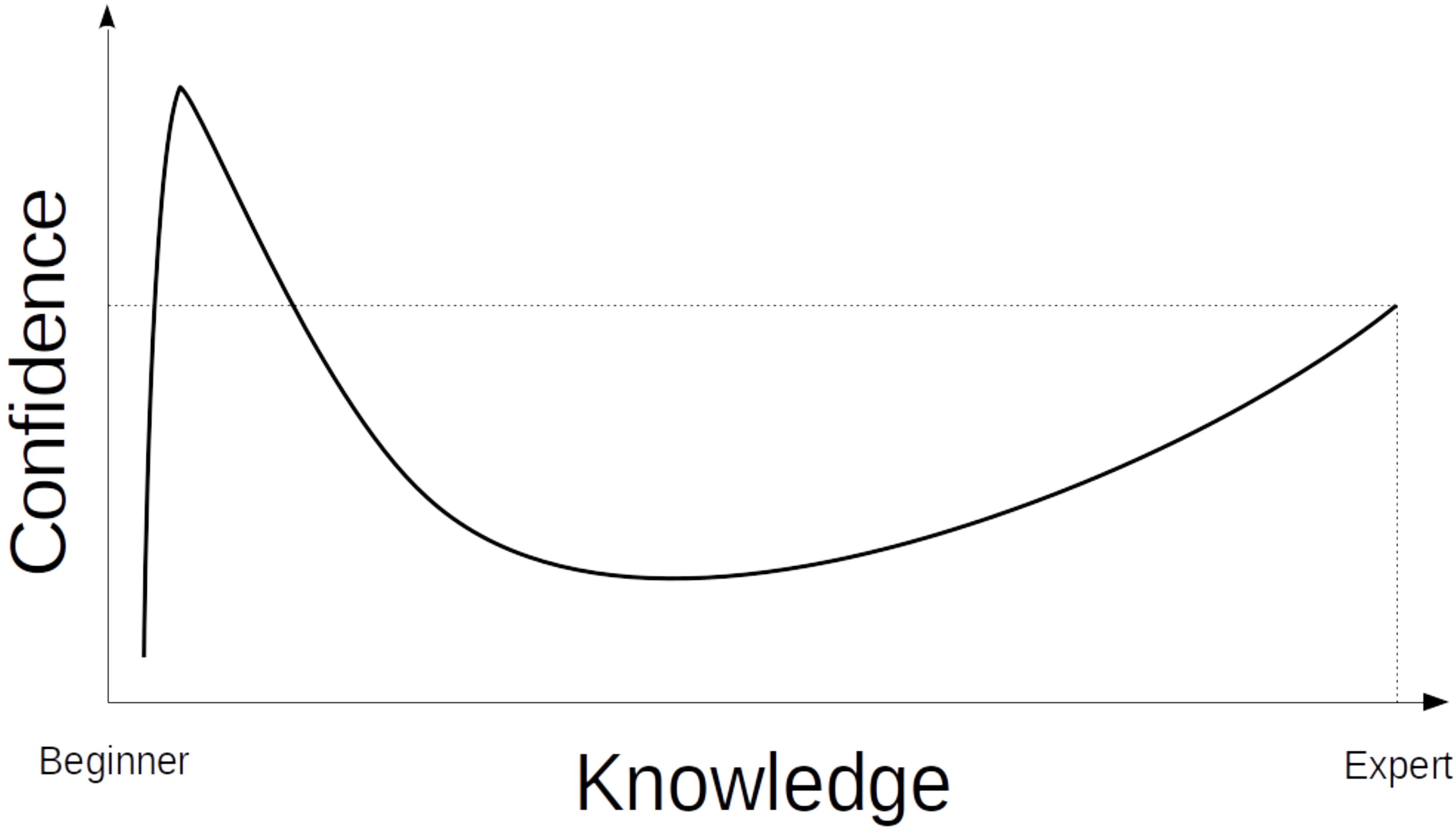
- The world is complex
  - We can never understand all of it

- ... So we think in models
  - "I heard that americans are stupid, therefore all americans I meet are stupid"

- Machines do this too:
  - ["... algorithms are ... wholly dependent on the data it is given" (https://medium.com/codait/cognitive-bias-in-machine-learning-d287838eeb4b)](https://medium.com/codait/cognitive-bias-in-machine-learning-d287838eeb4b)
  - Pretty dangerous in [for instance criminal prediction models (https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing)](https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing)

No models are true

- ... but some are useful



Linear $f(x) = x$

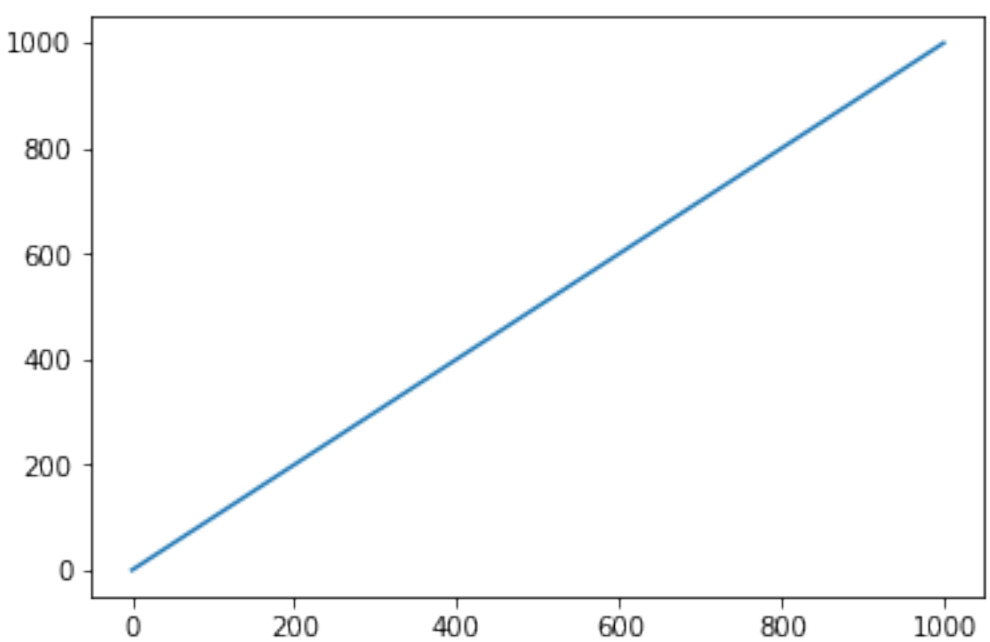
In [13]:

```python
import matplotlib.pyplot as plt


xs = range(0, 1000)
ys = [x for x in xs]
plt.plot(xs, ys)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7f51f803bdd8>]
```



- Running time -> calories burned
- Fuel in car -> distance it can drive

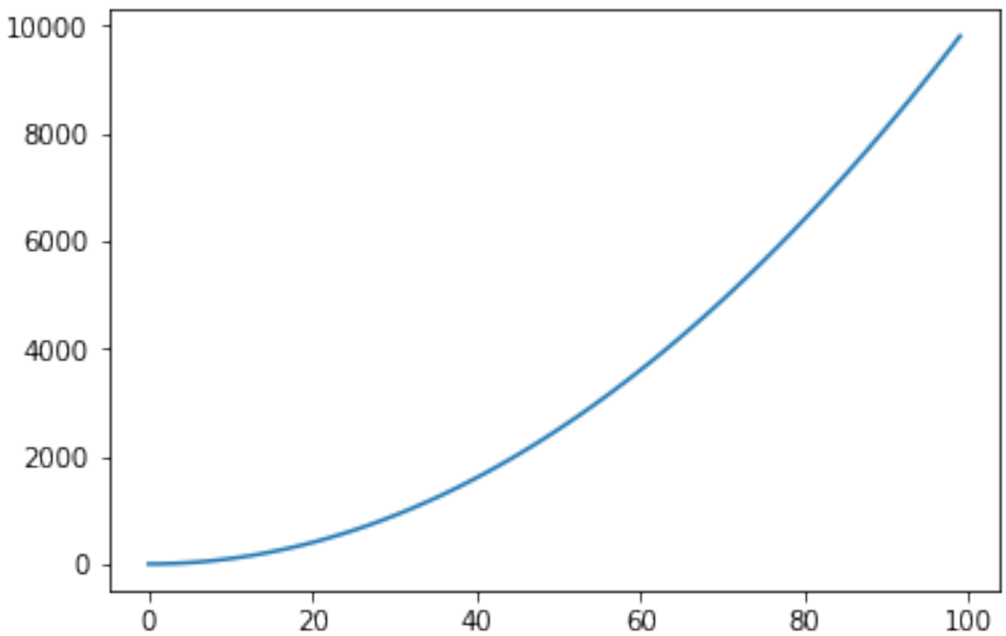
Quadratic $f(x) = x^2$

In [14]:

```python
xs = range(0, 100)
ys = [x * x  for x in xs]
plt.plot(xs, ys)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x7f51b4e0dba8>]
```



- Heat -> Ice cream sales
- Size of a square -> area

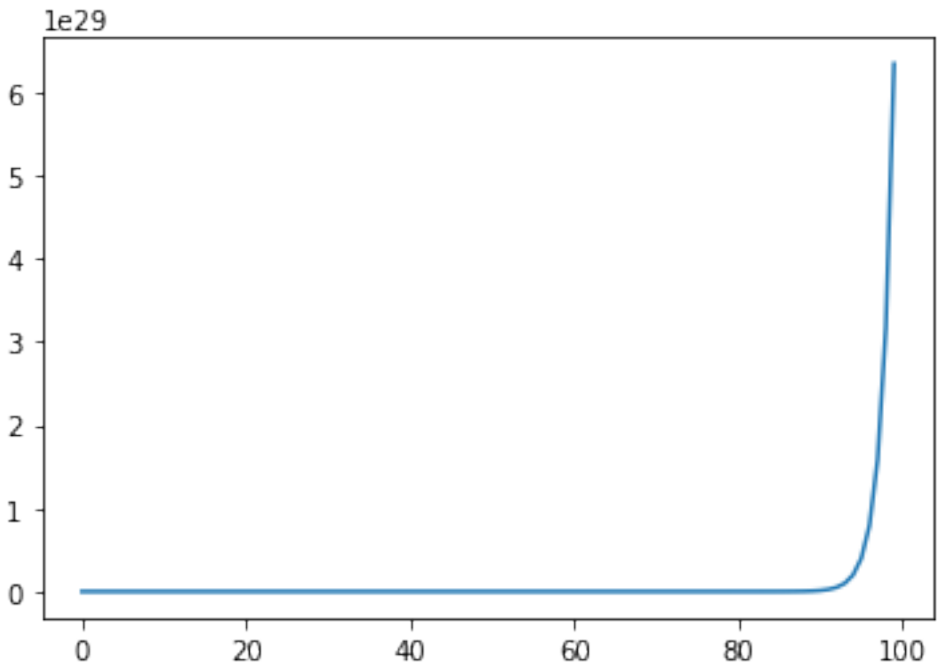
Exponential $f(x) = 2^x$

In [15]:

```python
xs = range(0, 100)
ys = [2 ** x for x in xs]
plt.plot(xs, ys)
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x7f51b4de7f98>]
```



- Loans work like this (compound interest: Pe^{rt})
- Radioactive decay ($Ne^{-t/\tau}$)
- Population growth

Logarithmic $f(x) = log_2(x)$

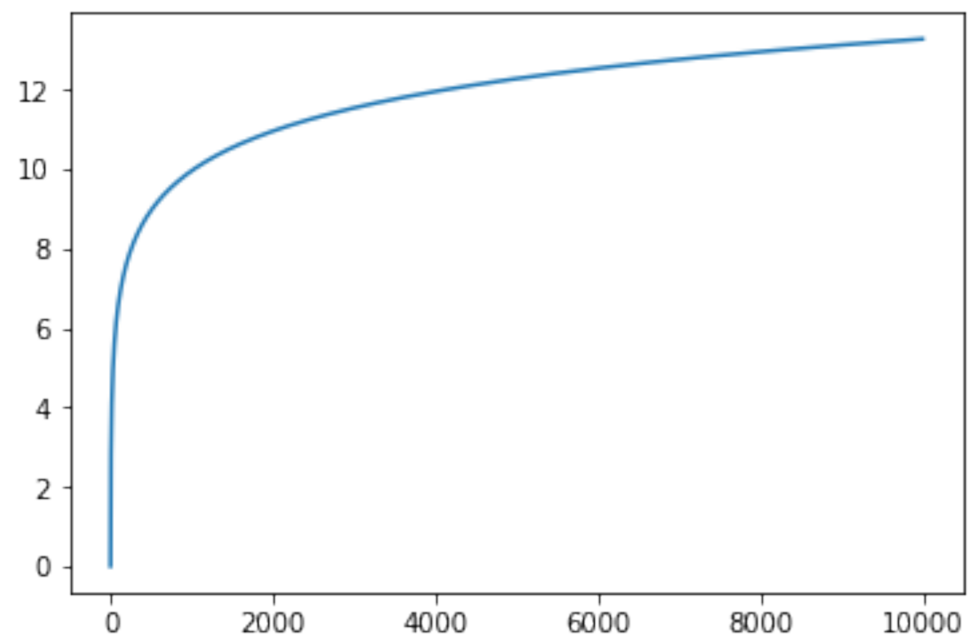
In [17]:

```python
import math


xs = range(1, 10000)
ys = [math.log2(x) for x in xs]
plt.plot(xs, ys)
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x7f51b4ce16d8>]
```



- Money -> Happiness
- Decibel and richter scale

Logartihm examples

In [18]:

```python
import math
```

In [19]:

```python
print(math.log10(1))
```

```
0.0
```

In [20]:

```python
print(math.log10(10))
```

```
1.0
```

In [21]:

```
print(math.log10(100))
```

```
2.0
```

In [22]:

```
print(math.log10(1000))
```

```
3.0
```

In [ ]:

```
print(math.log10(20))
```

Log_2

In [23]:

```
print(math.log2(1))
```

```
0.0
```

In [24]:

```
print(math.log2(2))
```

```
1.0
```

In [25]:

```
print(math.log2(4))
```

```
2.0
```

In [26]:

```
print(math.log2(8))
```

```
3.0
```

In [27]:

```
print(math.log2(16))
```

```
4.0
```

In [28]:

```
print(math.log2(3))
```

```
1.584962500721156
```

Logarithm definition

Answers: What is the exponent m that I need to raise the base b with to get x?

- $b^m = x$

Question: What is the exponent m that I need to raise the base 10 with to get 100?

Answer: $10^m = 100$

In [ ]:

```
math.log10(100)
```

Question: What is the exponent m that I need to raise the base 2 with to get 4?

Answer: $2^m = 4$

In [ ]:

```
math.log2(4)
```

In [ ]:

```
2 ** 2
```

Order of growth

Given 1 increase in input, how much output do we get?

- Linear: $f(x) = x$

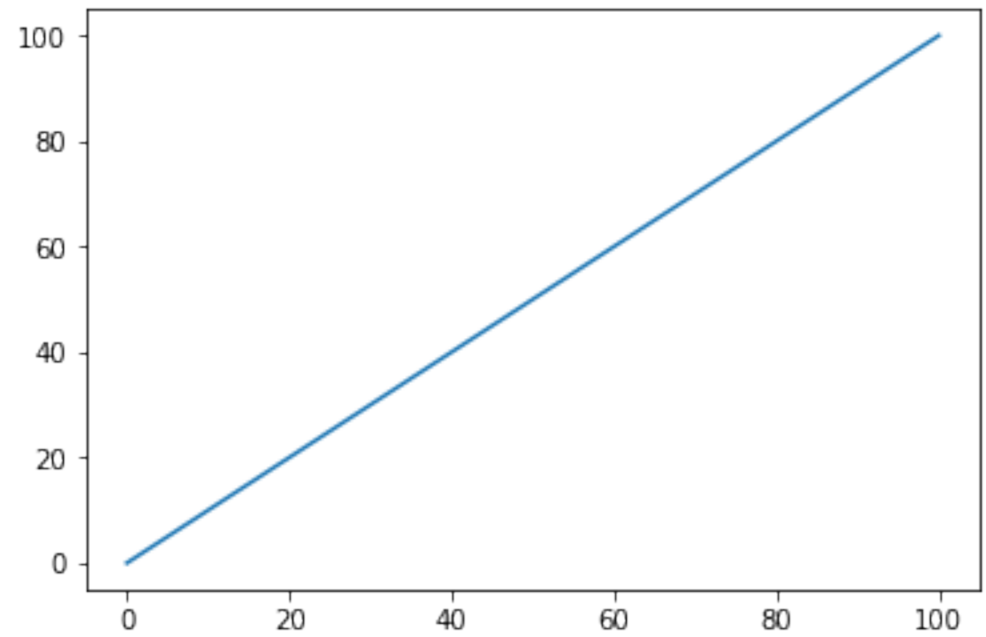
In [29]:

```python
import matplotlib.pyplot as plt


xs = [0, 100]
ys = [0, 100]
plt.plot(xs, ys)
```

Out[29]:

```
[<matplotlib.lines.Line2D at 0x7f51b4cc53c8>]
```



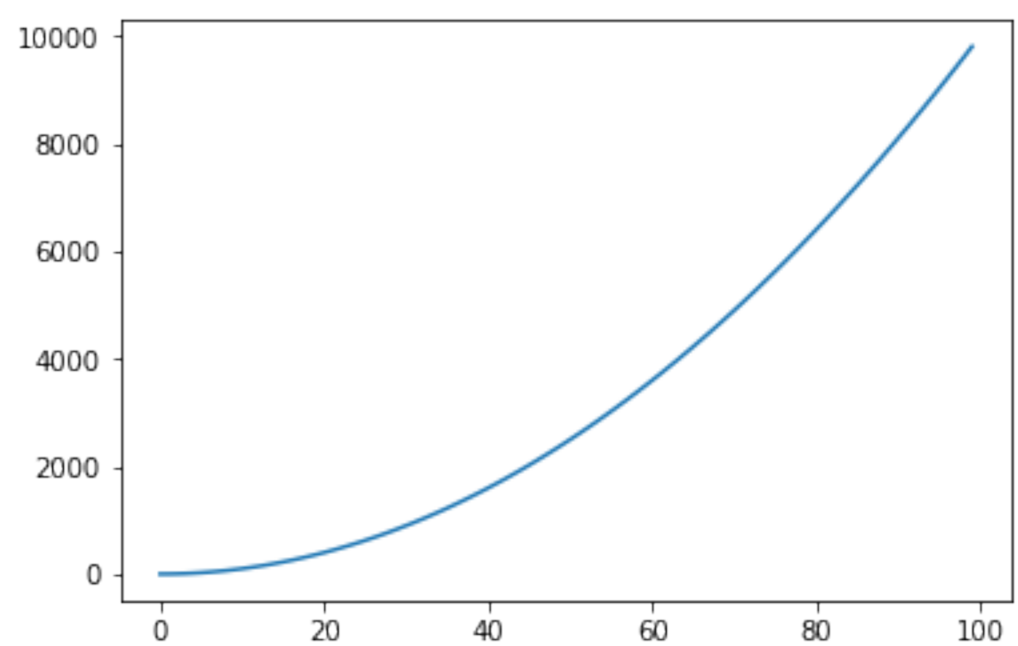
- Quadratic: $f(x) = x^2$

In [30]:

```python
xs = range(0, 100)
ys = [x * x for x in xs]
plt.plot(xs, ys)
```

Out[30]:

```
[<matplotlib.lines.Line2D at 0x7f51b4c5f710>]
```



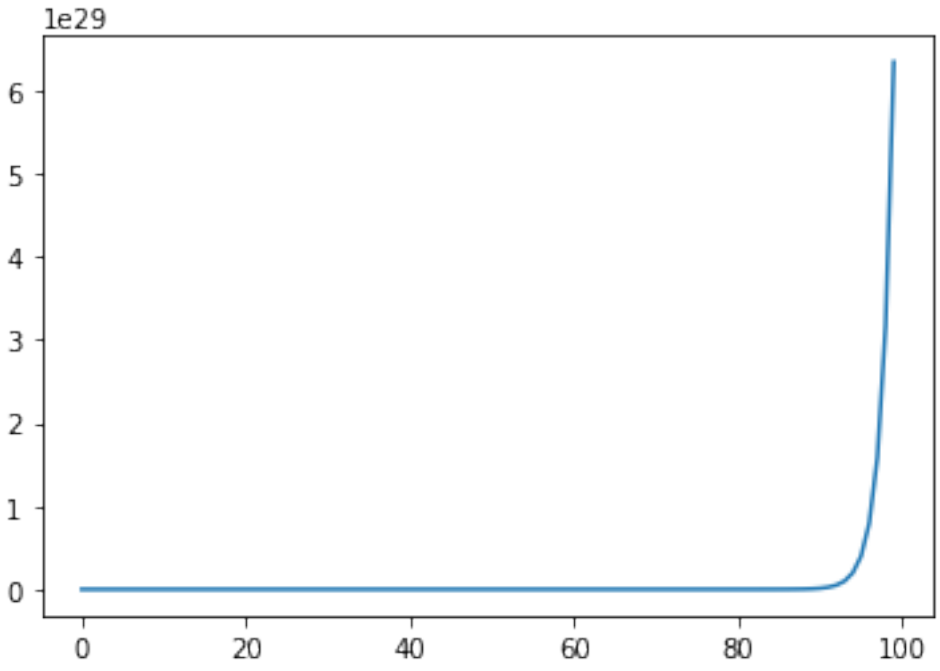
- Exponential: $f(x) = 2^x$

In [31]:

```
xs = range(0, 100)
ys = [2 ** x for x in xs]
plt.plot(xs, ys)
```

Out[31]:

```
[<matplotlib.lines.Line2D at 0x7f51b4c35b70>]
```



- Logarithm: $f(x) = log_2(x)$

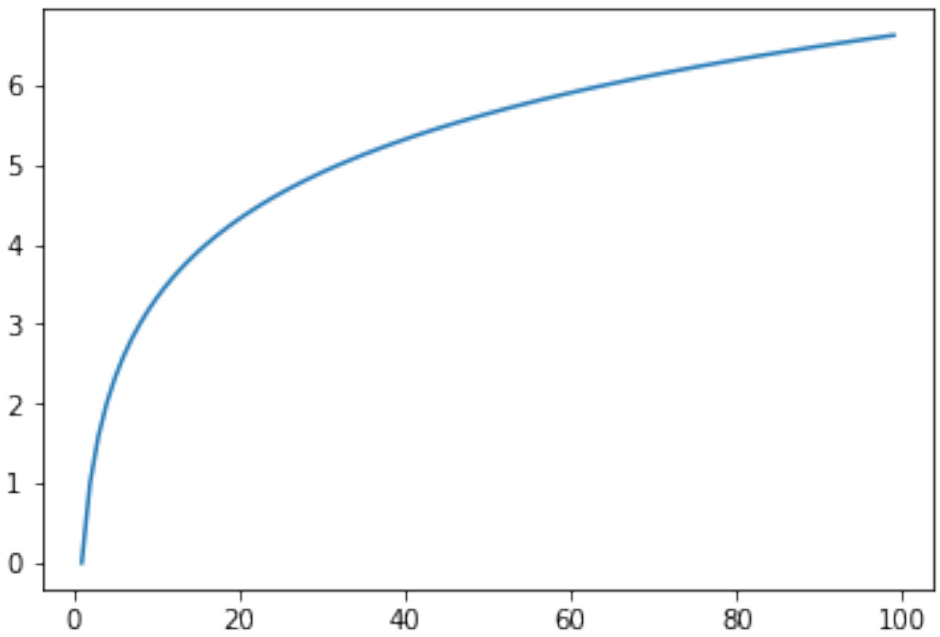
In [32]:

```python
import math


xs = range(1, 100)
ys = [math.log2(x) for x in xs]
plt.plot(xs, ys)
```

Out[32]:

```
[<matplotlib.lines.Line2D at 0x7f51b4b990b8>]
```



Viewing plots in Mu

To run programs that plot something:

```python
import matplotlib.pyplot as plt

xs = range(100)
ys = [x for x in xs]

plt.plot(xs, ys)
plt.show()
```

- If you run into problems, run the programs from the terminal using `pythonw` **instead of** `python` :
  - `$ pythonw my_python_file.py`