Lab 2. Plotting Graphs in Python

February 7, 2020

The College of Staten Island, Department of Mathematics, MTH229

Lab 2. Plotting Graphs in Python

1 Introduction

Mathematically, a function f is a rule that assigns to each value x in its domain, a corresponding value of y in its range. The graph of a function is then the collection of all points (x, y) such that $y = f(x)$, sketched in the Cartesian plane. Of course we can't realistically expect to draw the (typically) infinite collection of points. In a calculus class we learn to sketch graphs by focusing on their important features: zeroes, asymptotes, limits, points of discontinuity or non-smoothness, relative maxima and minima, and points of inflection. With these important features understood, a sketch then can be drawn accurately where need be and filled in broadly otherwise.

With Python we take a different approach. To plot the function f over the interval $[a, b]$, we actually plot as many pairs of points (x, y) as necessary to ensure an accurate representation of the graph. How many points? There are no good rules. We'll see examples (lines) where two are enough. As well, we will see examples where we can't possibly take enough points to do what we want. We'll just have to get used to experimenting to find the correct amount. With this approach, we need to be able to do the following:

- generate the x values of the points in our graph,
- generate the corresponding y values,
- plot the points and connect them with lines.

We've seen how to make regularly spaced x values using either the `r_[a:b:h]` construction or the `linspace(a,b,n)` function. In this project we'll learn how to make the corresponding y values, and how to then make the desired plot.

1.1 New Python topics

- Mathematics with arrays ("vectors")
- Graphing functions using pairs of arrays
- Printing a notebook to PDF

1.2 New Python commands

- `r_[ ]` - use to make an array of numbers
- `r_[a:b:h]` and `linspace(a,b,n)` – used to generate regularly spaced sequences of points.
- `plot(x,y)` – plot the two lists of numbers

- `title( )` – title our plot
- `grid()` – add a grid to our plot

2 Graphing with Python

Think back to how you first learned to graph a function or an equation. What would you do if you wanted to plot a graph of the parabola $y = x^2$ over the interval $-2 \le x \le 2$? We could choose a set of x values, say, $x = -2, -1, 0, 1, 2$, then square each x value to determine the corresponding y value ($y = 4, 1, 0, 1, 4$). These might be displayed together, as in the following table:

```
x = -2, -1, 0, 1, 2
y = 4, 1, 0, 1, 4
```

We would then mark each corresponding (x, y) pair as a point on a Cartesian coordinate system, and connect the points with straight lines.

2.1 Example 1

To graph $f(x) = x^2$ over the interval $[-2, 2]$ using Python we just need to know that the plot function will make the desired plot. The `#` and beyond are comments and need not have been typed in.

*Recall from **Lab 1** that we run a cell in Jupyter Lab by pressing SHIFT+ENTER. Occationally you may have to run a cell twice to get the plot to show.*

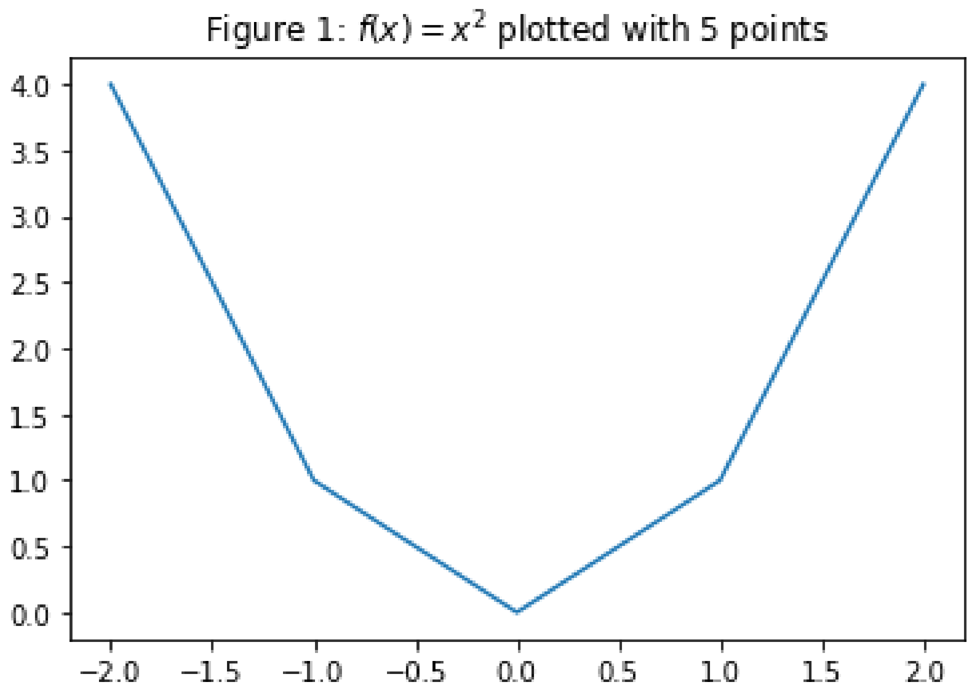
```
[2]: from matplotlib.pyplot import *
     from numpy import *

     x = r_[-2, -1, 0, 1, 2] # This is a comment!
     y = r_[4, 1, 0, 1, 4]

     plot(x,y)

     title('Figure 1: $f(x)=x^2$ plotted with 5 points')
```

```
[2]: Text(0.5, 1.0, 'Figure 1: $f(x)=x^2$ plotted with 5 points')
```



After storing our list of x-coordinates in the computer as `x` and the y-coordinates as `y` we can easily call them back.

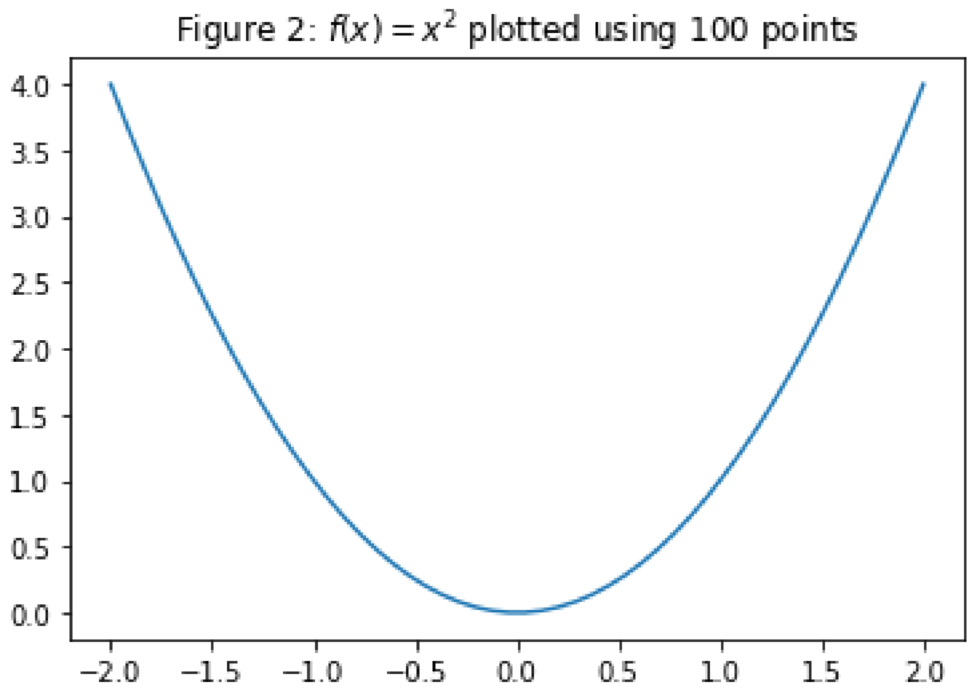
```
[3]: x
```

```
[3]: array([-2, -1,  0,  1,  2])
```

Figure 1 shows that our graph isn't quite what we expected. Rather than looking like a familiar parabola, we see a sequence of straight lines. How can we fix that? By taking more points. We have an easy way of taking more points, we can generate a 100 of them with the command `linspace(-2,2)` and, if need be, 1,000 with the command `linspace(-2,2,1000)`. However, we don't want to do all the squaring by hand. Rather Python should do the work. The next example shows how to do this for $f(x) = x^2$, and later on in this project we'll see how to do this for other functions.

```
[4]: x = linspace(-2,2)
     y = x**2
     plot(x,y)
     title('Figure 2: $f(x)=x^2$ plotted using 100 points')
```

```
[4]: Text(0.5, 1.0, 'Figure 2: $f(x)=x^2$ plotted using 100 points')
```



Using more points requires no more labor on your part than using just 5 points. If we didn't have enough we'd take more and replot. However, our graph (in Figure 2) now looks okay. Even though it may no longer be evident, the graph still consists of a sequence of straight lines!

We repeat the previous example with a different function.

2.2 Example 2

We want to graph the function $f(x) = e^x$ over the interval $[-1, 1]$. We'll need to know that in Python the function `exp(x)` performs e^x. Other than that, this example follows the three steps in plotting.

Step 1. We want to plot the function over the interval [−1, 1]. To do so, we first choose evenly spaced points between −1 and 1 with a step size of 0.5

Step 2. We then define the y values for each x value using `exp`

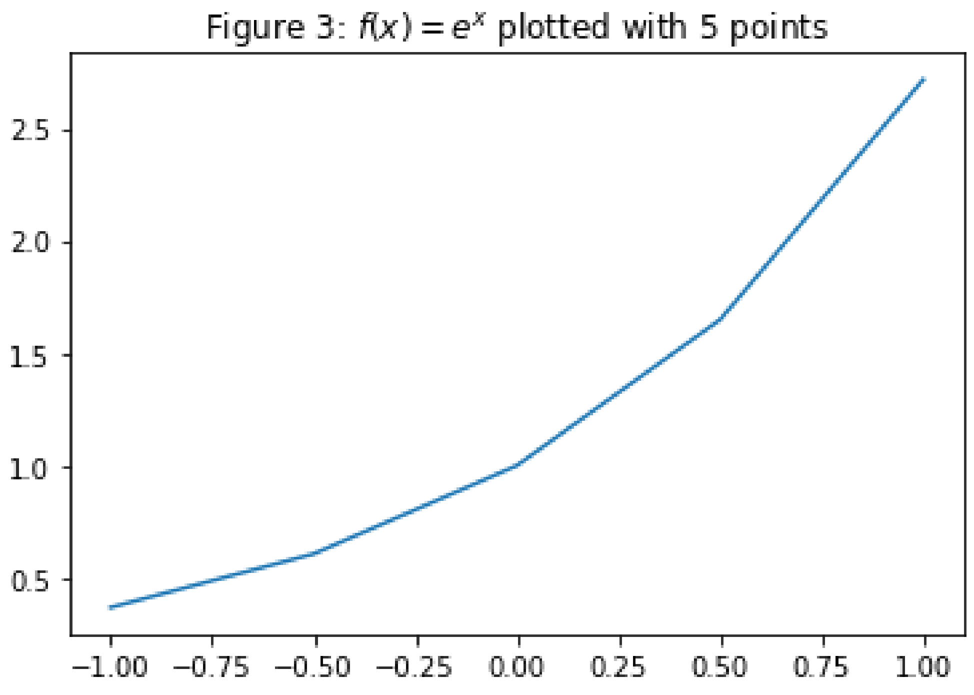
Step 3. Finally we use `plot( )` to view the function's graph

[5]:
```
x = r_[-1:1.5:0.5]
y = exp(x)
x, y
```

[5]:
```
(array([-1. , -0.5,  0. ,  0.5,  1. ]),
 array([0.36787944, 0.60653066, 1.        , 1.64872127, 2.71828183]))
```

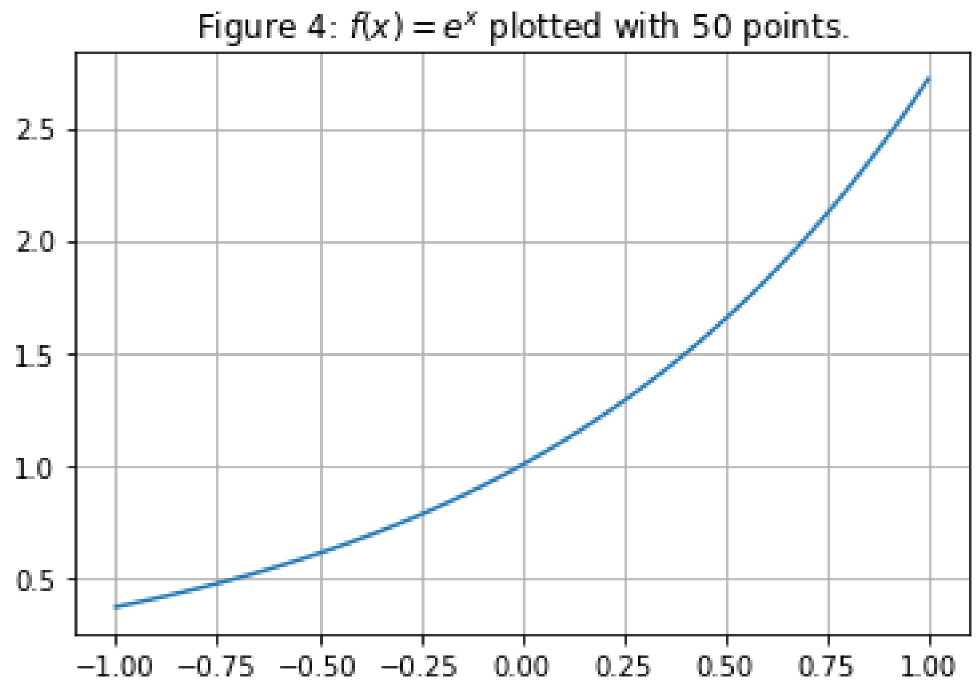
```
[6]: plot(x,y)
     title('Figure 3: $f(x)=e^x$ plotted with 5 points')
```

```
[6]: Text(0.5, 1.0, 'Figure 3: $f(x)=e^x$ plotted with 5 points')
```



As shown in Figure 3, this graph created with 5 sample points in $[-1, 1]$ is obviously not so smooth. To obtain a smooth looking curve one needs to define more x points. The exact number varies depending on how rapidly the function varies over its domain. We start with the simple default of **50** using `linspace` and see if this is enough:

```
[7]: x = linspace(-1,1)
     y = exp(x)
     plot(x,y)
     title('Figure 4: $f(x)=e^x$ plotted with 50 points.')
     grid()
```

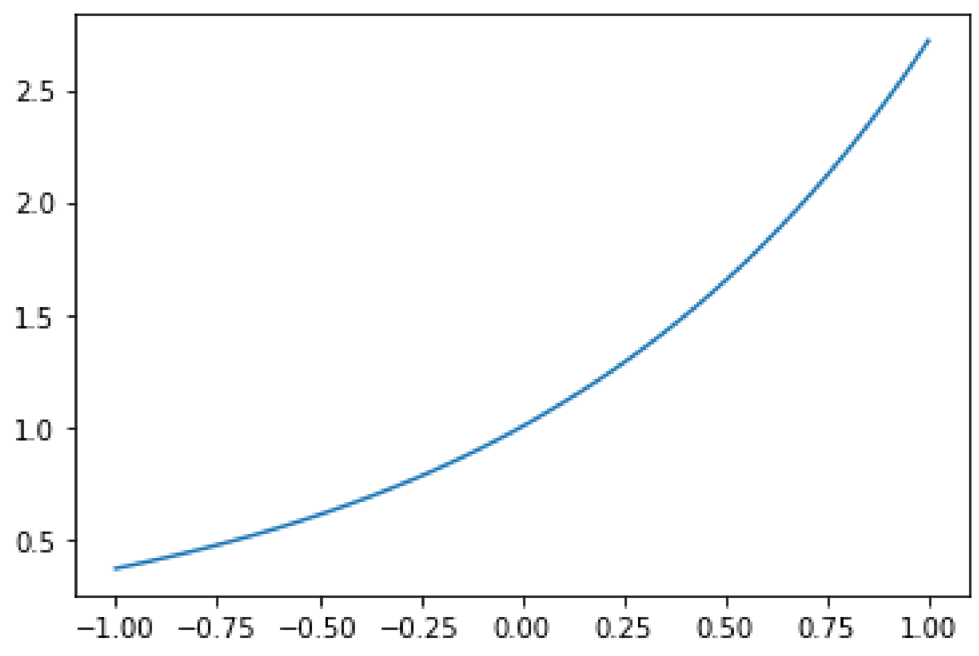


The graph in Figure 4 shows that 50 points is sufficient to present a smooth looking curve.

It is important that even though only the line defining the values for x is different, you must again evaluate the second and third lines. The values stored in y **don't update automatically**. If you forgot to recreate the y values, you would be trying to pair off the 50 x values with only 5 y values and an error would occur. For such a simple function we could have avoided this step in Figure 4 by defining y within the plot function, as in:

```
[8]: x = linspace(-1,1)
     plot(x, exp(x))
```

```
[8]: [<matplotlib.lines.Line2D at 0x114efcc90>]
```



2.3 Exercise 1

Create a graph of $y = \cos 4x$ over $[0, \pi]$. To illustrate what happens when there are too few points in your domain, first try **a step size of** $\pi/10$ (`pi/10`).

(Q1) Which command gives the desired values for x?

- `x = 0:pi/10:pi`
- `x = 0:pi:pi/10`
- `x = linspace(0,pi)`

[9]:
```
# Type your answer in a comment here
```

(Q2) Which command gives the correct answer for y?

- `y = cos(4x)`
- `y = cos4*x`
- `y = cos(4*x)`

[10]:
```
# Type your answer in a comment here
```

(Q3) Plot your graph with the plot command.

[11]:
```
# Type and run your code here.
```

(Q3) Redo your plot, this time using the command `x=linspace(0,pi)` to define the x array. Which plot looks more like the plot of a cosine curve?

- The first one

- the second one
- both of them

[12]:
```
# Type an run your code here
```

2.4 Exercise 2

We wish to plot the function $f(x) = e^{\cos x}$ over the interval $[0, 2\pi]$.

(Q4) What command generates a sufficient number of values for x?

- `linspace(0,2*pi)`
- `linspace(0,100,2*pi)`
- `r_[0:2*pi]`
- `r_[0:0.01:2*pi]`

(Q5) Which command will generate the corresponding y$ $values

- `exp^cos(x)`
- `e^cos(x)`
- `exp(cos(x))`
- `exp(x)cos(x)`

3 Algebraic expressions with arrays ("vectors")

We now know pretty well how to create the values for x using `linspace` or `r_`. To create the values of y is by applying operations to x is just as easy. We want to create the values simultaneously and so must use the proper Python syntax. For concreteness we call a single number a *scalar* and a set of numbers an *array* (known in MATLAB as a *vector*).

Suppose we have two terms `a` and `b` and want to find one of these arithmetic operations: `a+b`, `a-b`, `a*b`, `a/b`, or `a**b`. The answer depends on the whether the terms are scalars or arrays:

⚠ The only things to watch out for is this: 1. if both `a` and `b` are arrays then make sure **they have the same length**. 1. x^2 in Python is written as `x**2`, *not* with a ^

3.1 Example 3

Let's see what happens if you don't know how to work with arrays.

First define x to be the numbers 1 through 5.

[13]:
```
x = r_[1:6]
x
```

[13]:
```
array([1, 2, 3, 4, 5])
```

Do the following lines do what you expect?

[14]:
```
x+10
```

[14]:
```
array([11, 12, 13, 14, 15])
```

```
[15]: 10*x, x/10, x+10
```

```
[15]: (array([10, 20, 30, 40, 50]),
       array([0.1, 0.2, 0.3, 0.4, 0.5]),
       array([11, 12, 13, 14, 15]))
```

```
[16]: 10/x
```

```
[16]: array([10.        ,  5.        ,  3.33333333,  2.5       ,  2.        ])
```

So what's the fuss? In Python, there isn't any! In MATLAB and/or Julia, you may have learned to "worry about dots." **In Python we do *NOT* use dots** when applying `+ - / * **` to arrays or scalars.

⚠ *Notice that* ***in Python, power is written*** `**`:

```
[17]: x, x**2, 10**x, x**x
```

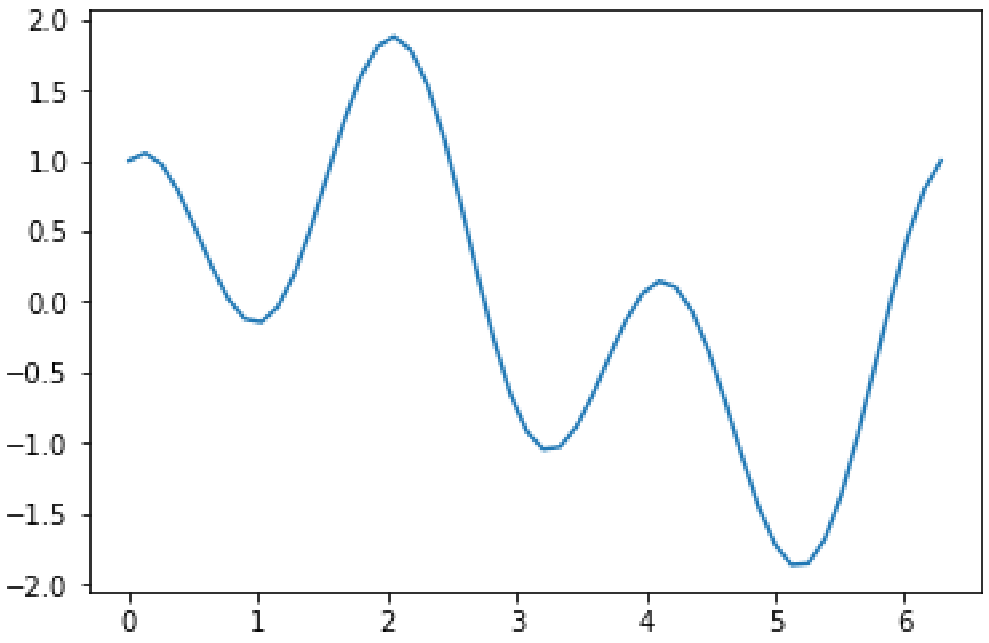
```
[17]: (array([1, 2, 3, 4, 5]),
       array([ 1,  4,  9, 16, 25]),
       array([    10,    100,   1000,  10000, 100000]),
       array([   1,    4,   27,  256, 3125]))
```

3.2 Example 4

Plot $y = \sin x + \cos 3x$ over the domain $[0, 2\pi]$.

```
[18]: x = linspace(0,2*pi)
      y = sin(x)+cos(3*x)
      plot(x,y)
```

```
[18]: [<matplotlib.lines.Line2D at 0x114f62510>]
```

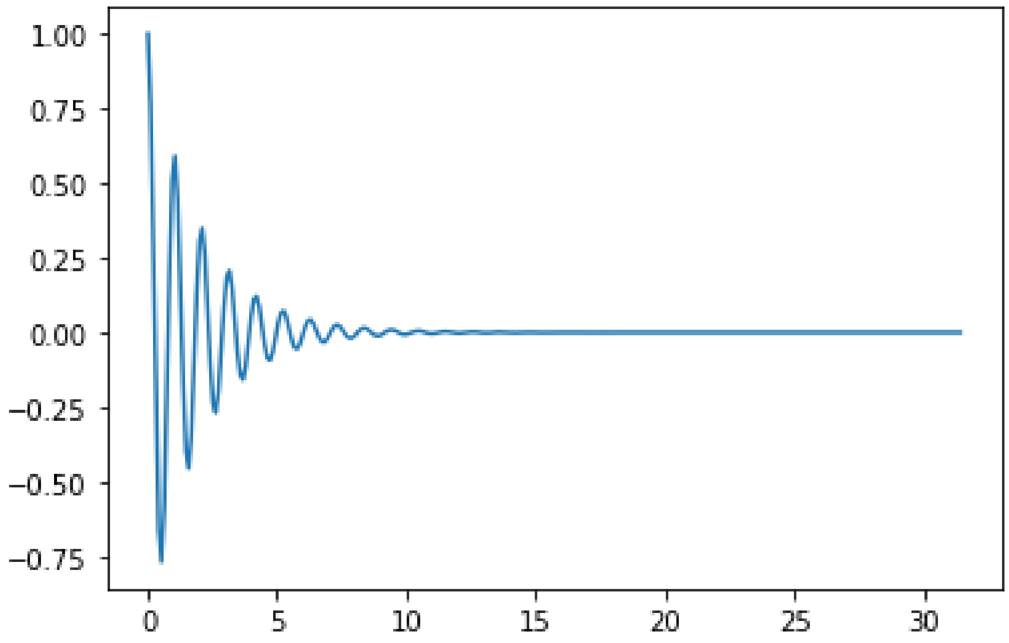


3.3 Example 5

Plot $y = e^{-x/2} \cos 6x$ over the domain $[0, 10\pi]$:

[19]:
```
x = linspace(0, 10*pi, 300)
y1 = exp(-x/2) # Here we break up the
y2 = cos(6*x)  # computation into
y = y1*y2      # bite-sized pieces
plot(x,y)
```

[19]:
```
[<matplotlib.lines.Line2D at 0x1150b1050>]
```



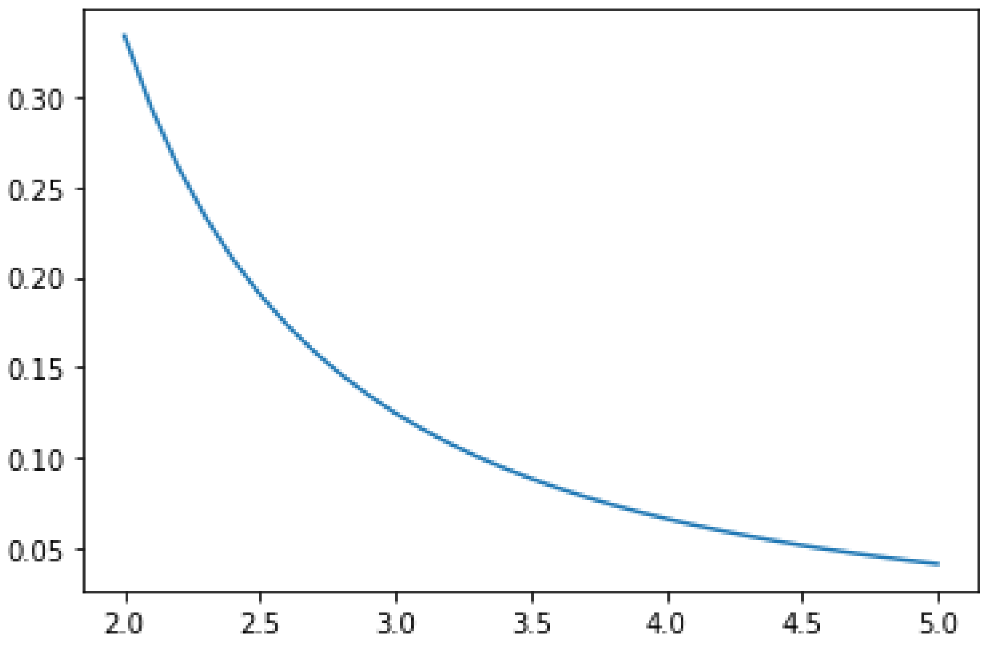
Again, python knows that the multiplication of these two *same-sized arrays* `y1` and `y2`is to be carried out element-by-element when we type `y1*y2` . To minimize the chance of errors, we broke the problem into intermediate calculations by using two variables `y1` and `y2`.

3.4 Example 6

Plot $y = 1/(x^2 - 1)$ over the domain $[2, 5]$:

[20]:
```
x = r_[2:5.1:0.1]
y = 1/(x**2-1)
plot(x,y)
```

[20]:
```
[<matplotlib.lines.Line2D at 0x11518d1d0>]
```



3.5 Exercise 3

Define a, b, and c by

[21]:
```
a = r_[1:21:2]
b = r_[1:11]
c = r_[1:12:2]
```

Which of the following is/are defined?

(Q6) `b+c`

(Q7) `a + b`

(Q8) `a./ b`

(Q9) `a * b`

(Q10) `a ^ 2`

[ ]:
```

```

[22]:
```
# Type your answer here in a comment:
#
# Q6
# Q7
# Q8
# Q9
# Q10
```

3.6 Exercise 4

For all problems in this exercise, let $x = 1, 2, 3$.

[23]:
```
x = r_[1:4]
```

Translate the following math statements into Python commands. *To help, the correct output values for the function when $x = 1, 2, 3$ is given to you.*

(For example) Write Python commands to compute x^3. The output you get should be `array([ 1,  8, 27])`

[24]:
```
# Write out and run your code here:

x = r_[1:4]
x**3
```

[24]:
```
array([ 1,  8, 27])
```

(Q10) Write Python commands to compute $\cos x \sin x$. The output you should get is `array([ 0.45464871, -0.37840125, -0.13970775])`

[25]:
```
# Write out and run your code here:
```

(Q11) Write Python commands to compute $\sin^2 x$. Your code should produce `array([0.70807342, 0.82682181, 0.01991486])`

[26]:
```
# Write out and run your code here:
```

(Q12) Write Python commands to compute... $\sin x^2$

You should get `array([ 0.84147098, -0.7568025 ,  0.41211849])`

[27]:
```
# Write out and run your code here:
```

(Q13) Write Python commands to compute: $f(x) = 7x^2 \sin \frac{1}{7x^2}$

You should get `array([0.99660211, 0.99978743, 0.99995801])`

[28]:
```
# Write out and run your code here:
```

(Q14) Write Python commands to compute: $f(x) = x - \frac{\cos x - \sin x}{\sin x + \cos x}$

You should get `array([1.2179581 , 4.68770694, 1.66751188])`

[29]:
```
# Write out and run your code here:
```

(Q15) Write Python commands to compute:

$$f(x) = \frac{1}{10}\left(x - \frac{x^{3/2}}{10}\right)^2$$

You should get `array([0.081      , 0.29486292, 0.61523085])`

[30]:
```
# Write out and run your code here:
```

4 How to submit your work as a PDF

Plan A. At the top of the webpage, go to…

- **File** > **Export Notebook as…** > **Export Notebook to PDF**

Plan B. If your computer's Python has not been set up correctly, you may get an error (you must have TeXLive or MacTeX, and Anaconda).

So **IF** "Export to PDF" does not work, go to…

- **File** > **Export Notebook as…** > **Export Notebook to HTML** >
- a new browser window will open up: print it to PDF (**Command+P** then select printer: "save to PDF")

4.1 Exercise 5

(Q19) Graph the function $f(x) = \sin(\frac{\pi}{2}x) + \sin(\frac{2}{5}\pi x)$ over the interval $[0, 40]$.

[31]:
```
# Make your graph here
```

(Q16) How many peaks (relative maxima) does the graph have?

- 2
- 3
- 4
- 5
- none of the above

(Q17) This function is periodic. How many periods are graphed in $[0, 40]$?

- 2
- 3
- 4
- 5
- none of the above

(Q18) Estimate from your graph the value of $f(10)$ to at least 1 decimal point.

[32]:
```
# Type your answer in a comment here:
```

4.2 Exercise 6

(Q20) Graph the function $f(x) = \cos^2 x - \sin^2 x$ over the interval $[-2\pi, 2\pi]$. Use 50 points in the domain.

[33]:
```
# Make your graph here
```

(Q21) Does the graph resemble any graph that you are familiar with?

- $\cos 2x$
- $\cos x/2$
- $\cos x$

[34]:
```
# Type your answer in a comment here:
```

4.3 Exercise 7

For this exercise we look at the graph of the polynomial function $f(x) = x^3 - 20x^2 + 10x - 1$.

[35]:
```
# Type your answer in a comment here:
```

(Q22) First plot the function over the interval $[-10, 10]$. What is the approximate range for the y-axis?

- $[-10, 10]$
- $(-10, 10)$
- $[-3100, 0]$
- $[0, 2\pi]$

[36]:
```
# Type your answer in a comment here:
```

(Q23) We wish to investigate when (if) this function is positive. We can't readily tell from our graph so we will replot over a smaller domain. Which of these domains seems appropriate for this task?

- $[0, 500]$
- $[0, 10]$
- $[-1, 1]$
- $[0, 2\pi]$

[37]:
```
# Type your answer in a comment here:
```

(Q24) Replot the graph over the selected domain. Turn on the grid by entering the command `grid()` on its own line. From your graph, which of these x values have $f(x) > 0$? Indicate all that apply:

- 0
- 0.25
- 0.50
- 0.75

[38]:
```
# Type your answer in a comment here:
```

[39]:
```
# Enter your code that produces the graph WITH A GRID here
```