Lab A: Using Python as a Calculator
https://mybinder.org/v2/gh/anniebmcc/pycalclab/master?filepath=mat301a.ipynb
2020 Summer — Calculus 1
Dr Matthew H Sunderland

Jupyter Notebooks

A1. **RUN** the following "code cell" (gray rectangle with `In[ ]` next to it), by CLICKING the code cell and pressing SHIFT+RETURN. Notice that only the last result will display.

In [1]:
```
1 + 2 + 3
50 - 3
100*5
```
Out[1]: 500

A2. **RUN** the following. As always, only the last result displays, but the last result has 2 parts because of the comma.

In [2]:
```
1 + 2 + 3
50 - 3, 1000*1000
100*5, 7*7
```
Out[2]: (500, 49)

A3. The "+" on the toolbar adds a code cell. The "scissors" deletes a cell.

Python arithmetic + - * / **

A4. **RUN** the following.

In [3]:
```
3 + 10*5,  5**2,  27/10
```
Out[3]: (53, 25, 2.7)

A5. **EXERCISE.**
a) What does each of the 5 arithmetic operations do?
b) Do spaces around the 5 operations matter, or is it just style?

In [4]:
```
# TYPE YOUR ANSWERS BELOW
#
#  a) + is   addition
#     - is   subtraction
#     * is   multiplication
#     / is   division
#    ** is   exponentiation
#
#  b) No, spaces aroung + - * / ** don't matter
```

Python # and =

A6. **RUN** the following. You will notice python ignores everything after #

In [5]:
```
# This is a comment
1 + 1 # This is also a comment
```

Out[5]: 2

A7. **RUN** the following. Notice we assign variables using = Assignment itself does NOT produce output.

In [6]:
```
a = 10
a
```

Out[6]: 10

In [7]:
```
b = 20
```

In [8]:
```
a = 18
b = 21
c = a - b
c
```

Out[8]: -3

A8. **RUN** the following. Notice you can assign multiple variables at once with a comma.

In [9]:
```
x, y = 100, 500
x
```

Out[9]: 100

In [10]:
```
a,b,c = 3,4,5
a + b/c
```

Out[10]: 3.8

A9. **RUN** the following. See that we can compute $\frac{(2-3)*-3}{-1+2}$ all at once (1st cell below), or we can assign variables to help us (2nd cell below).

In [11]:
```
(2 - 3)*-3/(-1 + 2)
```

Out[11]: `3.0`

In [12]:
```
top = (2 - 3)*-3
bottom = -1 + 2
top/bottom
```

Out[12]: `3.0`

A10. **EXERCISE.** Assign variables to help you compute $3 - \frac{3^2-2\cdot3}{2\cdot3-2}$

In [13]:
```
# Type your answer below and press SHIFT+ENTER

top = 3**2 - 2*3
bottom = 2*3 - 2
3 - top/bottom
```

Out[13]: `2.25`

Order of Operations

A11. **RUN** the following. Notice $a - b * c = a - (b * c)$, but they do not equal $(a - b) * c$.

In [14]:
```
a,b,c = 3,4,5

a - b*c,   a - (b*c),   (a - b)*c
```

Out[14]: `(-17, -17, -5)`

A12. **EXERCISE.** In each row, identify NON-equivalent choice. For example, the answer to (1) is $(a - b) * c$ because $a - b * c = a - (b * c)$

(1)	$a - b * c$	$a - (b * c)$	$(a - b) * c$
(2)	$a * (b - c)$	$(a * b) - c$	$a * b - c$
(3)	$a \,/\, b + c$	$a \,/\, (b + c)$	$(a \,/\, b) + c$
(4)	$(a + b) \,/\, c$	$a + (b \,/\, c)$	$a + b \,/\, c$
(5)	$a ** (b * c)$	$(a ** b) * c$	$a ** b * c$
(6)	$a * (b ** c)$	$a * b ** c$	$(a * b) ** c$
(7)	$a/b ** c$	$(a/b) ** c$	$a/(b ** c)$
(8)	$a ** b/c$	$(a ** b)/c$	$a ** (b/c)$
(9)	$(3 - 3) - 3$	$3 - 3 - 3$	$3 - (3 - 3)$
(10)	$(2 ** 3) ** 2$	$2 ** (3 ** 2)$	$2 ** 3 ** 2$
(11)	$6 \,/\, 3 \,/\, 2$	$6 \,/\, (3 \,/\, 2)$	$(6 \,/\, 3) \,/\, 2$

In [15]:
```
# TYPE YOUR ANSWERS BELOW.
#
#  (1)   (a - b)*c
#  (2)   a*(b - c)
#  (3)   a/(b + c)
#  (4)   (a + b)/c
#  (5)   a ** (b*c)
#  (6)   (a*b) ** c
#  (7)   (a/b) ** c
#  (8)   a ** (b/c)
#  (9)   3 - (3 - 3)
# (10)   (2 ** 3) ** 2
# (11)   6/(3/2)
```

A13. **RUN** the following example, where we add 2 sets of parentheses which show the order of the 2 operations.

In [16]:
```
1 + 3/5
```
Out[16]:
```
1.6
```

In [17]:
```
(1 + (3/5))
```
Out[17]:
```
1.6
```

A14. **EXERCISE.** Add 4 sets of parentheses, which show the order of the 4 operations.

In [18]:
```
7 - 3 ** 2/9 + 4
```
Out[18]:
```
10.0
```

In [19]:
```
# Type your answer below and press SHIFT+ENTER

((7 - ((3 ** 2)/9)) + 4)
```
Out[19]: 10.0

A15. **EXERCISE.** Assign `a,b,c = 4,5,8` and then evaluate $\dfrac{a^b - c/b}{c-a}, \dfrac{a^{c-b}}{c-b}, \dfrac{a^{3/2}}{b}, \dfrac{a - b(c-a)}{c-a}$

In [20]:
```
# Type your answer below and press SHIFT+ENTER

a,b,c = 4,5,8

(a**b - c/b)/(c-a),  a**(c-b)/(c-b),  a**(3/2)/b,  (a - b*(c-a))/(c-a)
```
Out[20]: (255.6, 21.333333333333332, 1.6, -4.0)

Making python functions

A16. **RUN** the following.

In [21]:
```
def g(x):
    return x**2

g(7)
```
Out[21]: 49

In [22]:
```
def h(n): return n + 100

h(7)
```
Out[22]: 107

A17. **EXERCISE.** Make the function $P(x) = x^2 - 2x + 1$ and find $P(P(7))$.

In [23]:
```
# Type your answer below and press SHIFT+ENTER

def P(x):
    return x**2 - 2*x + 1

P(P(7))
```
Out[23]: 1225

Built-in `%pylab` functions

Python	Math notation	Meaning
`abs(x)`	$\lvert x \rvert$	absolute value
`sqrt(x)`	$\sqrt{x}$	square root
`exp(x)`	e^x	exponential function
`log(x)`	$\ln x$	natural logarithm
`sin(x)`	$\sin x$	sine
`arcsin(x)`	$\sin^{-1} x$	inverse sine
`radians(x)`		converts degrees to radians

A18. **RUN** the code cells below. The command `%pylab` only needs to be run once per lab; it loads "built-in functions" (from python packages numpy and matplotlib).

In [24]:
```
%pylab

sqrt(49)
```

```
Using matplotlib backend: MacOSX
Populating the interactive namespace from numpy and matplotlib
```

Out[24]:
```
7.0
```

In [25]:
```
pi, exp(1), sin(pi/2)
```

Out[25]:
```
(3.141592653589793, 2.718281828459045, 1.0)
```

A19. **EXERCISE.** Evaluate

1. $\sin 40^\circ$
2. $\sin^2 65^\circ$
3. $e^{(10-8.5)/3}$
4. $\arcsin(\sin(3\pi/4))$

Note. Python uses radians for all angle measurements, so you need to convert any degrees to radians.

In [26]:
```
# Type your answer below and press SHIFT+ENTER

sin(radians(40)),  sin(radians(65))**2,  exp((10-8.5)/3),  arcsin(sin(3*pi/
4))
```

Out[26]:
```
(0.6427876096865393,
 0.8213938048432696,
 1.6487212707001282,
 0.7853981633974484)
```

Making an array with `r_[ ]`

A20. **RUN** the following. (If you get an error, go back and run A17.) The function `r_[ ]` can make an array of numbers of your choice. We will need arrays for graphing (Lab B).

In [27]:
```
x = r_[2,3,4,5,10]
x**3
```

Out[27]:
```
array([   8,   27,   64,  125, 1000])
```

A21. **EXERCISE.** Use `r_[ ]` to store the numbers 2,3,5,7,11 in an array named `x`. Find `x*x`.

In [28]:
```
# Type your answer below and press SHIFT+ENTER

x = r_[2,3,5,7,11]
x*x
```

Out[28]:
```
array([  4,   9,  25,  49, 121])
```

Making an array with `r_[a:b:stride]`

A22. **RUN** the following. In general, `r_[a:b]` will list integers from a up to but *not* including b. A missing a is the same as 0.

In [29]:
```
r_[5:10]
```

Out[29]:
```
array([5, 6, 7, 8, 9])
```

In [30]:
```
r_[:5]
```

Out[30]:
```
array([0, 1, 2, 3, 4])
```

A23. **EXERCISE.** Use `r_[a:b]` to make the array 1,2,3,4,5,6,7,8,9

In [31]:
```
# Type your answer below and press SHIFT+ENTER

r_[1:10]
```

Out[31]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

A24. **RUN** the following. In general, `r_[a:b:stride]` spaces out your numbers by the amount `stride`.

In [32]:
```
r_[0:100:2]
```

Out[32]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
       34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66,
       68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98])
```

A25. **EXERCISE.** Use `r_[a:b:stride]` to make the array $1, 3, 5, \ldots, 99$

In [33]:
```
# Type your answer below and press SHIFT+ENTER

r_[1:100:2]
```

Out[33]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
       69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99])
```

Making an array with `linspace(a,b,n)`

A26. **RUN** the following. Observe that `linspace(a,b,n)` lists n numbers from a to b inclusive. This is useful for generating a lot of evenly-spaced numbers, such as when graphing (Lab B). Observe that `linspace(a,b)` lists 50 numbers from a to b inclusive.

In [34]:
```
linspace(0,10,6)
```

Out[34]:
```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

In [35]:
```
linspace(0,10)
```

Out[35]:
```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
        2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
        3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
        4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
        5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
        6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
        7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
        8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
        9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

A27. **EXERCISE.** Use `linspace(a,b,n)` to make the array $1, 1.5, 2, 2.5, 3, 3.5, 4$

In [36]:
```
# Type your answer below and press SHIFT+ENTER

linspace(1,4,7)
```

Out[36]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. ])
```

A28. **EXERCISE.**
Convert average body temperature $98.6^\circ F$ to Celsius using $C = 5/9(F - 32)$.

In [37]:
```
# Type your answer below and press SHIFT+ENTER

5/9*(98.6 - 32)
```

Out[37]:
```
37.0
```

A29. **RUN** the following.
Notice that `x` and `y` are arrays,
`c_[x,y]` puts them into a table.

```
In [38]: x = r_[:10]
         y = x**2
         c_[x,y]
```

```
Out[38]: array([[ 0,  0],
                [ 1,  1],
                [ 2,  4],
                [ 3,  9],
                [ 4, 16],
                [ 5, 25],
                [ 6, 36],
                [ 7, 49],
                [ 8, 64],
                [ 9, 81]])
```

A30. **EXERCISE.**
Use `r_` to make an array of Fahrenheit values `x` $= -100, -80, -60, \ldots, 100.$
Make the corresponding array of Celsius values `y`
Use `c_` to put `x` and `y` into a table.

```
In [39]: # Type your answer below and press SHIFT+ENTER

         x = r_[-100:101:20]
         y = 5/9*(x - 32)
         c_[x,y]
```

```
Out[39]: array([[-100.        ,  -73.33333333],
                [ -80.        ,  -62.22222222],
                [ -60.        ,  -51.11111111],
                [ -40.        ,  -40.        ],
                [ -20.        ,  -28.88888889],
                [   0.        ,  -17.77777778],
                [  20.        ,   -6.66666667],
                [  40.        ,    4.44444444],
                [  60.        ,   15.55555556],
                [  80.        ,   26.66666667],
                [ 100.        ,   37.77777778]])
```

Lab B: Plotting Graphs in Python
https://mybinder.org/v2/gh/anniebmcc/pycalclab/master?filepath=mat301b.ipynb
2020 Summer — Calculus 1
Dr Matthew H Sunderland

Plotting with `plot`

B1. Example. To graph $f(x) = x^2$ over $[-2, 2]$ by hand, make an xy table: choose some x values,

x	-2	-1	0	1	2
y					

and then use f to compute the corresponding y values.

B2. **RUN** the following. Notice that graphing in python is similar to B1: we make a list of x values and y values.

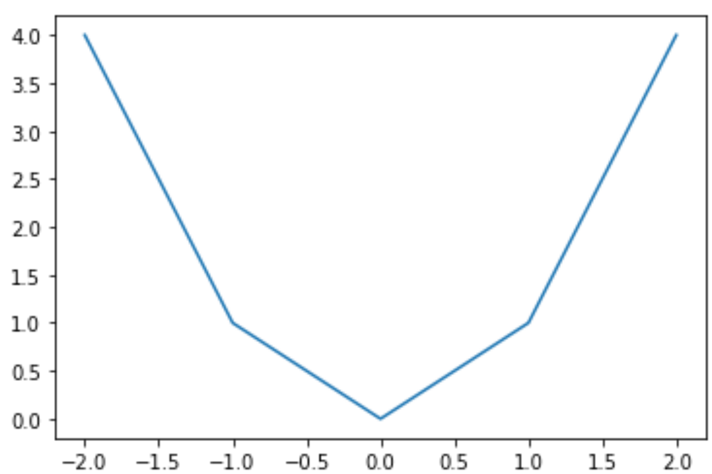
Note to instructor: you may remember that in A18 we wrote `%pylab` and here we write `%pylab inline`; the "inline" tells Jupyter to display images inline instead of as a pop-up.

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
x = r_[-2, -1, 0, 1, 2]
y = r_[4, 1, 0, 1, 4]
plot(x,y)
```

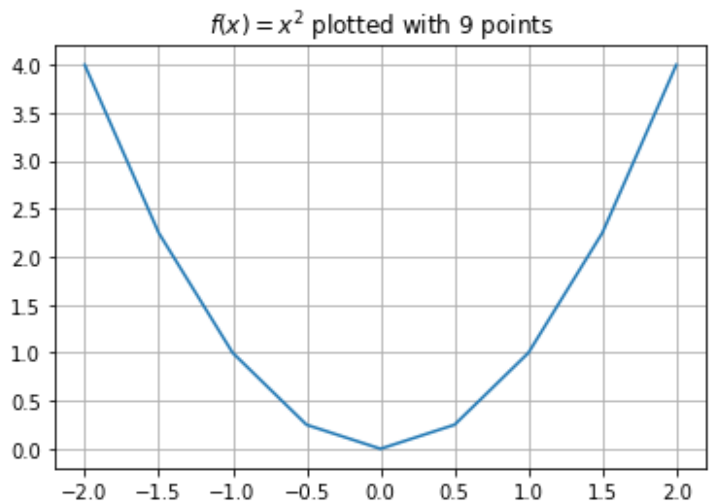
Out[2]:
```
[<matplotlib.lines.Line2D at 0x7f9e427944d0>]
```



B3. **RUN** the following. Notice that we save time by making the x array using `linspace` (see A27) and making the y array by doing arithmetic on x (see A29). For illustrative purposes, we use `c_[x,y]` to make a table out of the arrays x and y (see A29).

In [3]:
```
x = linspace(-2,2,9)
y = x**2

plot(x,y)
title('$f(x) = x^2$ plotted with 9 points')
grid()

c_[x,y]
```

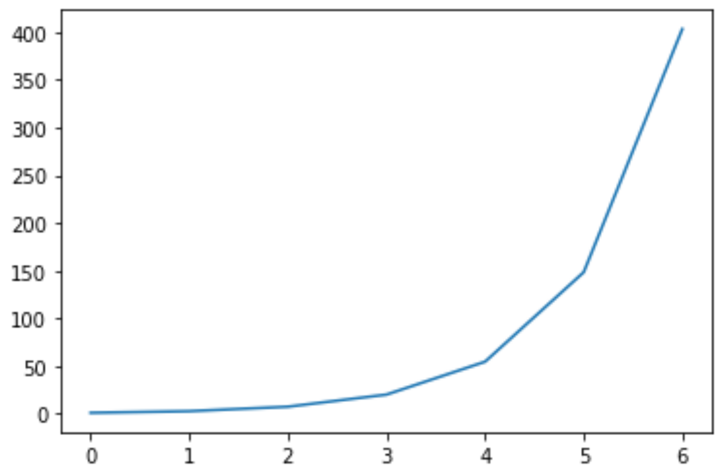
Out[3]:
```
array([[-2.  ,  4.  ],
       [-1.5 ,  2.25],
       [-1.  ,  1.  ],
       [-0.5 ,  0.25],
       [ 0.  ,  0.  ],
       [ 0.5 ,  0.25],
       [ 1.  ,  1.  ],
       [ 1.5 ,  2.25],
       [ 2.  ,  4.  ]])
```



B4. **RUN** the following, which graph $f(x) = e^x$ over the interval $[0, 7]$. Here we make our array x using `r_[a:b:stride]` (see A22). Remember that `exp(x)` is how you write e^x in python (see A18).

```
In [4]: x = r_[:7]
        y = exp(x)
        plot(x,y)
```

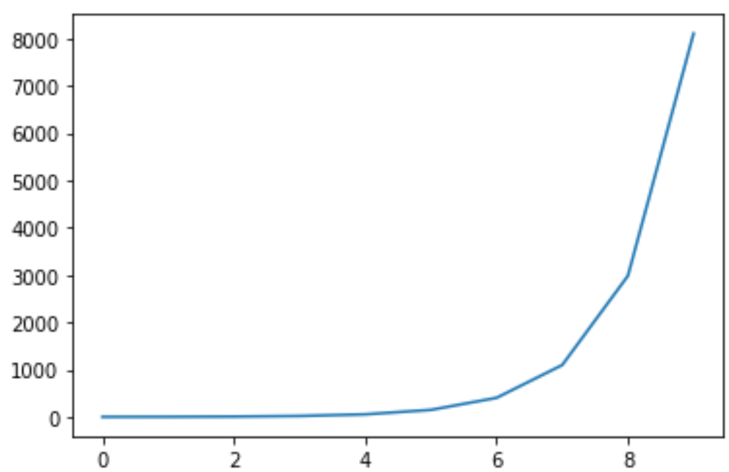
Out[4]: [<matplotlib.lines.Line2D at 0x7f9e42a15a90>]



B5. **RUN** the following. When we change the x we must recompute the y; there are two ways to do it (compare B4 to B5).

```
In [5]: x = r_[:10]
        plot(x, exp(x))
```

Out[5]: [<matplotlib.lines.Line2D at 0x7f9e42b71850>]



B6. **EXERCISE.**

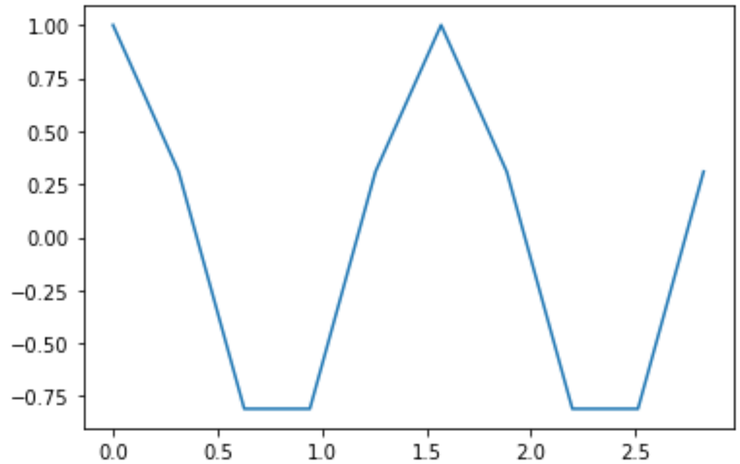
(1) Graph $y = \cos 4x$ over $[0, \pi]$ with a step size of `pi/10`

(2) Redo your plot from iii. using `x = linspace(0,pi)`

(3) Which plot looks more like the plot of a cosine curve?

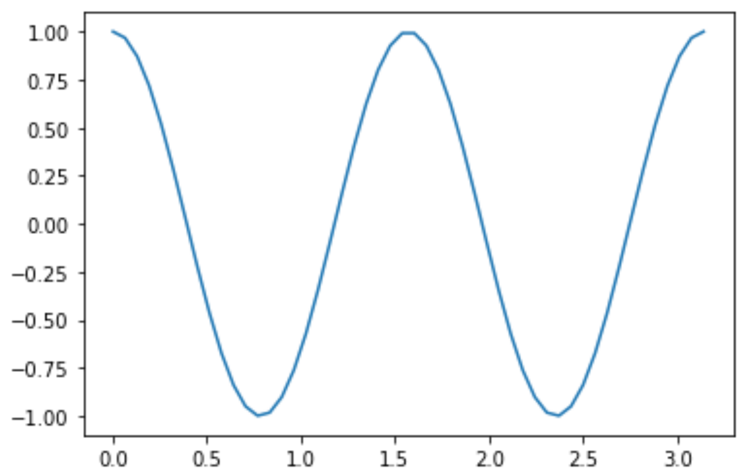
In [6]:
```
# (1) Type your answer below and press SHIFT+ENTER

x = r_[0:pi:pi/10]
y = cos(4*x)
plot(x,y)
```

Out[6]: [<matplotlib.lines.Line2D at 0x7f9e42be5090>]



In [7]:
```
# (2) Type your answer below and press SHIFT+ENTER

x = linspace(0,pi)
y = cos(4*x)
plot(x,y)

# (3) Your answer: the second plot
```

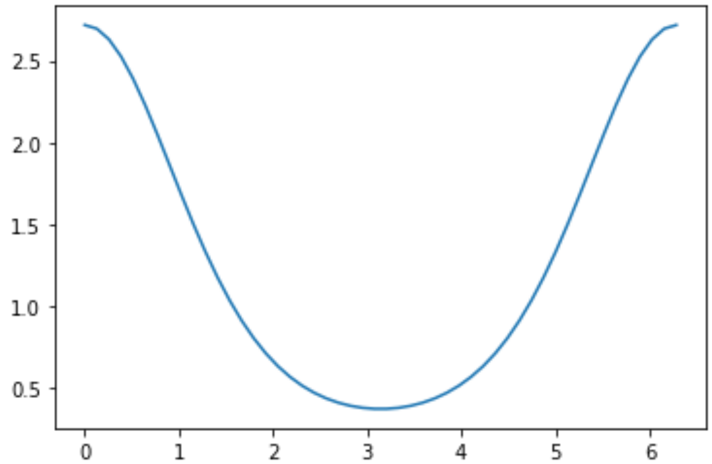
Out[7]: [<matplotlib.lines.Line2D at 0x7f9e42cba150>]



B7. **EXERCISE.** Plot the function $f(x) = e^{\cos x}$ over the interval $[0, 2\pi]$.

```
In [8]: # Type your answer below and press SHIFT+ENTER

        x = linspace(0,2*pi)
        y = exp(cos(x))
        plot(x,y)
```

```
Out[8]: [<matplotlib.lines.Line2D at 0x7f9e42f14950>]
```



Doing arthmetic on arrays

B8. **RUN** the following.
We make numpy arrays with `r_` or `linspace`
Numpy arrays "know" how to do "elementwise" arithmetic.
Warning: x^2 is written `x**2` .

```
In [9]: x = r_[1:5]

        x,  10 - x,  x + 10,  10*x,  x**2,  12/x,  x**x,  10**x
```

```
Out[9]: (array([1, 2, 3, 4]),
         array([9, 8, 7, 6]),
         array([11, 12, 13, 14]),
         array([10, 20, 30, 40]),
         array([ 1,  4,  9, 16]),
         array([12.,  6.,  4.,  3.]),
         array([  1,   4,  27, 256]),
         array([   10,   100,  1000, 10000]))
```

B9. **RUN** the following.

```
In [10]: # We can add arrays of the same shape (same length)

         x = r_[10, 20, 50, 100]
         y = r_[3, 0, 7, -1]
         x + y
```

```
Out[10]: array([13, 20, 57, 99])
```

In [11]:
```
# We can add an array (x) and a scalar (y)

x = r_[10, 20, 50, 100]
y = 100
x + y
```

Out[11]:
```
array([110, 120, 150, 200])
```

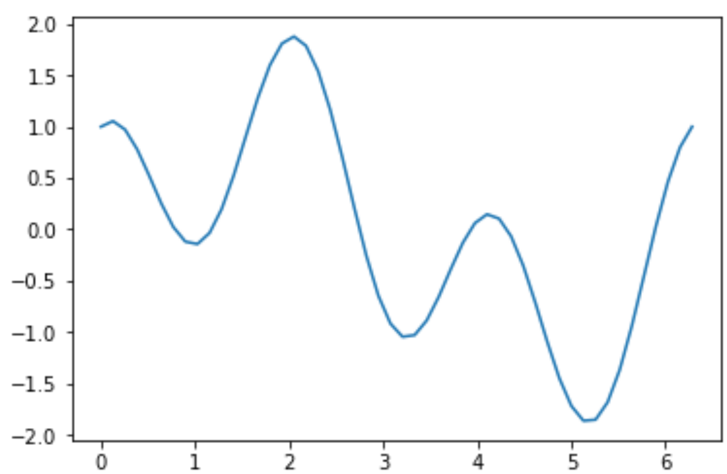
In [12]:
```
# We CANNOT add arrays of DIFFERENT shape

x = r_[10, 20, 50, 100]
y = r_[3, 0, 7]
x + y
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-ab56767c8fea> in <module>
      3 x = r_[10, 20, 50, 100]
      4 y = r_[3, 0, 7]
----> 5 x + y

ValueError: operands could not be broadcast together with shapes (4,) (3,)
```

B10. **RUN** the following.

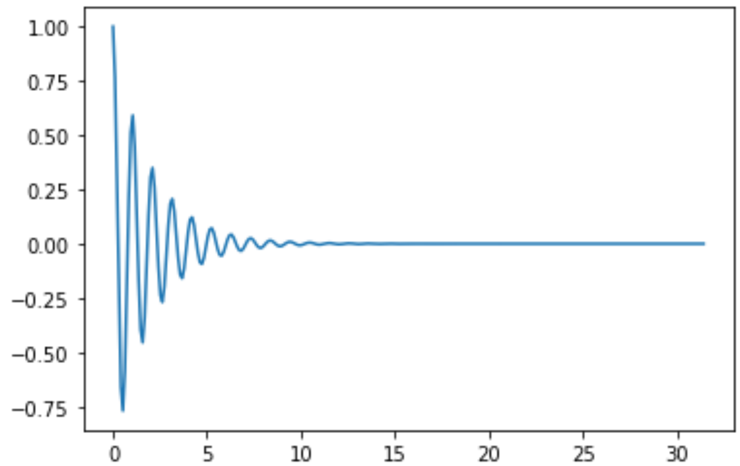
In [13]:
```
# y = sin x + cos 3x over the domain [0,2pi]

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

Out[13]:
```
[<matplotlib.lines.Line2D at 0x7f9e43060090>]
```



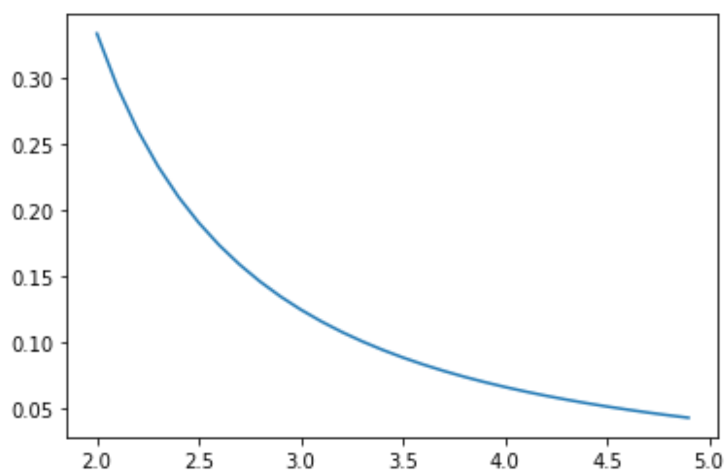
In [14]:
```
# y = e^(-x/2) cos 6x over the domain [0,10pi]

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)
```

Out[14]: [<matplotlib.lines.Line2D at 0x7f9e43144b90>]



In [15]:
```
# y = 1/(x^2 - 1) over the domain [2,5]

x = r_[2:5:0.1]
y = 1/(x**2 - 1)
plot(x,y)
```

Out[15]: [<matplotlib.lines.Line2D at 0x7f9e432306d0>]



B11. **EXERCISE.** First **RUN** the following.

In [16]:
```
a,b,c = r_[:5], r_[:50:10], r_[:10]
a,b,c
```

Out[16]:
```
(array([0, 1, 2, 3, 4]),
 array([ 0, 10, 20, 30, 40]),
 array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]))
```

Now, that we've defined a, b, c, which of the following are defined?

$a + b \qquad a + c \qquad a + 1 \qquad a * b \qquad c ** 2 \qquad c \ \text{^} \ 2$

In [17]:
```
# Type your answer below and press SHIFT+ENTER

a+b, a+1, a*b, c**2
```

Out[17]:
```
(array([ 0, 11, 22, 33, 44]),
 array([1, 2, 3, 4, 5]),
 array([  0,  10,  40,  90, 160]),
 array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81]))
```

B12. **RUN** the following example. Let x be the array 1,2,3. Write Python commands to compute x^3.
The output you get should be `array([ 1,  8, 27])` .

In [18]:
```
x = r_[1,2,3]
x**3
```

Out[18]:
```
array([ 1,  8, 27])
```

B13. **EXERCISE.** Using the same array `x = r_[1,2,3]` , find:

$\cos x \sin x \qquad \sin^2 x \qquad \sin x^2 \qquad 7x^2 \sin \frac{1}{7x^2}$

You should get

```
array([ 0.45464871, -0.37840125, -0.13970775])
array([0.70807342, 0.82682181, 0.01991486])
array([ 0.84147098, -0.7568025 ,  0.41211849])
array([0.99660211, 0.99978743, 0.99995801])
```

In [19]:
```
# Type your answer below and press SHIFT+ENTER

cos(x)*sin(x),  sin(x)**2,  sin(x**2),  7*x**2*sin(1/(7*x**2))
```

Out[19]:
```
(array([ 0.45464871, -0.37840125, -0.13970775]),
 array([0.70807342, 0.82682181, 0.01991486]),
 array([ 0.84147098, -0.7568025 ,  0.41211849]),
 array([0.99660211, 0.99978743, 0.99995801]))
```

B14. **EXERCISE.** Using the same array `x = r_[1,2,3]` , find:

$x - \frac{\cos x - \sin x}{\sin x + \cos x}$ $\qquad$ $\frac{1}{10}(x - \frac{x^{3/2}}{10})^2$

You should get

```
array([1.2179581 , 4.68770694, 1.66751188])
array([0.081      , 0.29486292, 0.61523085])
```

In [20]:
```
# Type your answer below and press SHIFT+ENTER

x - (cos(x)-sin(x))/(sin(x)+cos(x)),   1/10*(x - x**(3/2)/10)**2
```

Out[20]:
```
(array([1.2179581 , 4.68770694, 1.66751188]),
 array([0.081      , 0.29486292, 0.61523085]))
```

Graphing practice

B15 **EXERCISE.**

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

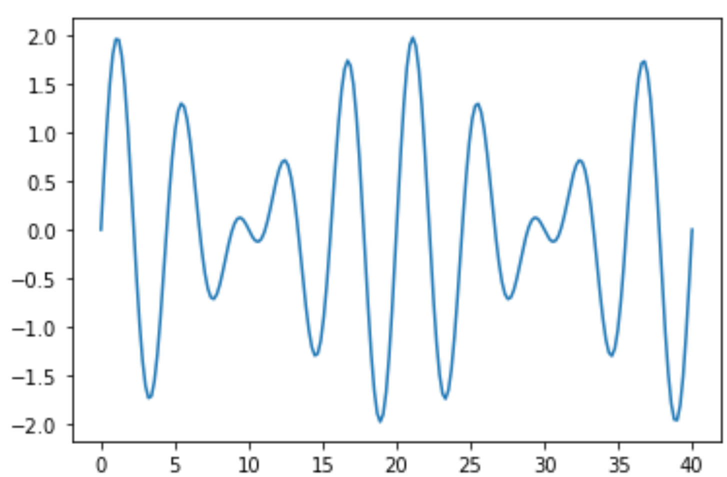
(2) How many peaks (relative maxima) does your graph have?

(3) This function is periodic; how many periods are graphed in $[0, 40]$?

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

In [21]:
```
# (1) Type your answer below and press SHIFT+ENTER

x = linspace(0,40,200)
y = sin(pi/2*x) + sin(2/5*pi*x)
plot(x,y)

# (2) Your answer: 10
# (3) Your answer: 2
# (4) Your answer: 0.0
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x7f9e4296a5d0>]
```



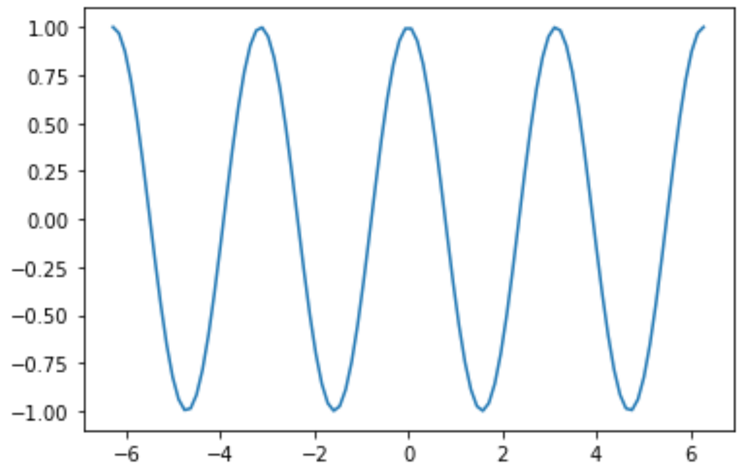
B16. **EXERCISE.**

(1) Graph $f(x) = \cos^2 x - \sin^2 x$ over the interval $[-2\pi, 2\pi]$ using 100 points.

(2) Does the resemble any of the following? $\cos 2x$ $\cos x/2$ $\cos x$

In [22]:
```
# (1) Type your answer below and press SHIFT+ENTER

x = linspace(-2*pi, 2*pi, 100)
y = cos(x)**2 - sin(x)**2
plot(x,y)

# (2) Your answer:  cos(2x)
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x7f9e428f5810>]
```



B17. **EXERCISE.**

(1) Plot the polynomial function $f(x) = x^3 - 20x^2 + 10x - 1$ over the interval $[-10, 10]$.

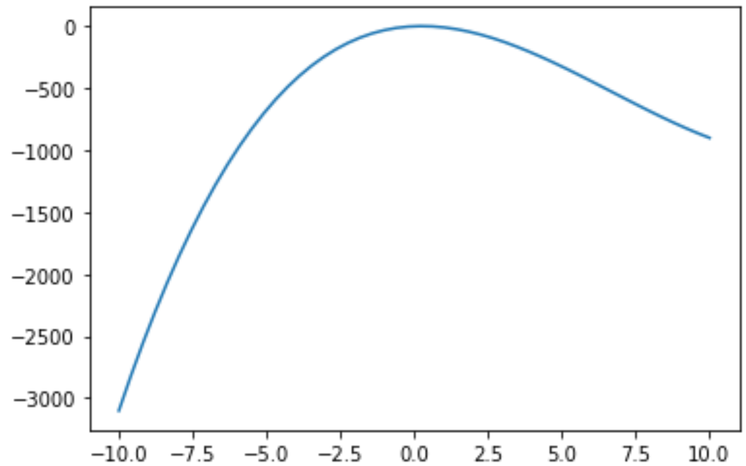
(2) Which is the approximate range for the y-axis?

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

In [23]:

```
# (1) Type your answer below and press SHIFT+ENTER

x = linspace(-10,10)
y = x**3 - 20*x**2 + 10*x - 1
plot(x,y)

# (2) Your answer:  [-3100,0]
```

Out[23]: [<matplotlib.lines.Line2D at 0x7f9e433ff3d0>]



B18. **EXERCISE.** We wish to investigate when (if) the function in B17 is positive. We can't readily tell from our graph in B17 so we will replot over a smaller domain.

(1). Which of these domains seems appropriate for this task?

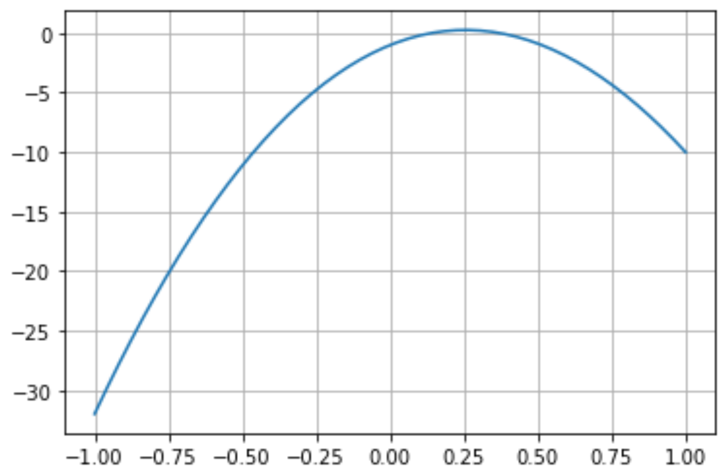
$[0, 500]$ $\quad$ $[0, 10]$ $\quad$ $[-1, 1]$ $\quad$ $[0, 2\pi]$

(2) Replot the graph over the selected domain. Turn on the grid using `grid()`

(3) From your graph, which of these x values have $f(x) > 0$? Indicate all that apply:

0 $\quad$ 0.25 $\quad$ 0.50 $\quad$ 0.75

In [24]:
```
# (1) Your answer:  [-1,1]
# (2) Type your answer below and press SHIFT+ENTER

x = linspace(-1,1)
y = x**3 - 20*x**2 + 10*x - 1
plot(x,y)
grid()

# (3) Your answer:  0.25
```



Lab C: Finding Limits in Python
https://mybinder.org/v2/gh/anniebmcc/pycalclab/master?filepath=mat301c.ipynb
2020 Summer — Calculus 1
Dr Matthew H Sunderland

Note to instructor: Troubleshooting
(1) If during the lab a student gets a `NameError` *, tell them to try running the* `%pylab inline` *in section* *C1* *below.*
(2) If they get scientific notation, tell them to try running the `set_printoptions(precision=17, suppress=True)` *in* *C5* *below.*

C1. **RUN** the following. When we are asked to compute a limit, the first thing we try is plugging in. Sometimes this works, such as for $\lim_{x\to\pi/2} \sin(x)/x$:

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
x = pi/2
sin(x)/x
```

Out[2]:
```
0.6366197723675814
```

C2. **RUN** the following. However sometimes plugging in doesn't work, such as for $\lim_{x\to 0} \sin(x)/x$:

In [3]:
```
x = 0
sin(x)/x
```

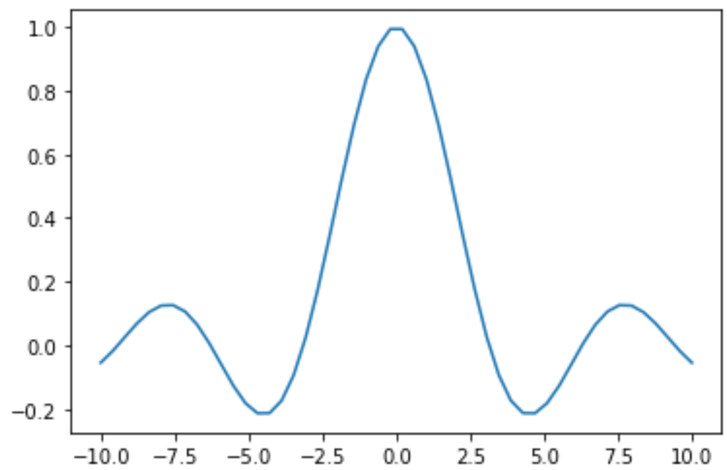
```
/Users/sunderland20a/opt/anaconda3/lib/python3.7/site-packages/ipykernel_l
auncher.py:2: RuntimeWarning: invalid value encountered in double_scalars
```

Out[3]:
```
nan
```

C3. **RUN** the following. When plugging in doesn't work, there are two things we can do in Python. First we can look at the graph. The graph tells us that $\lim_{x\to 0} \sin(x)/x = 1$.

In [4]:
```
x = linspace(-10,10)
y = sin(x)/x
plot(x,y)
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x7feb6fe99750>]
```



C4. **RUN** the following. The second thing we can do is make a table of x and $f(x)$ where we plug in numbers closer and closer to the given $x = a$. The table tells us that $\lim_{x \to 0} \sin(x)/x = 1$.

In [5]:
```
x = r_[1, .1, .01, .001]
y = sin(x)/x
c_[x,y]
```

Out[5]:
```
array([[1.        , 0.84147098],
       [0.1       , 0.99833417],
       [0.01      , 0.99998333],
       [0.001     , 0.99999983]])
```

In [6]:
```
x = r_[-1, -.1, -.01, -.001]
y = sin(x)/x
c_[x,y]
```

Out[6]:
```
array([[-1.        ,  0.84147098],
       [-0.1       ,  0.99833417],
       [-0.01      ,  0.99998333],
       [-0.001     ,  0.99999983]])
```

C5. **RUN** the following. Sometimes python will give us scientific notation. We use the command `set_printoptions` to turn off scientific notation and get more digits of precision.

In [7]:
```
x = r_[1, .1, .01, .001, .0001, .00001]
y = sin(x)/x
c_[x,y]
```

Out[7]:
```
array([[1.00000000e+00, 8.41470985e-01],
       [1.00000000e-01, 9.98334166e-01],
       [1.00000000e-02, 9.99983333e-01],
       [1.00000000e-03, 9.99999833e-01],
       [1.00000000e-04, 9.99999998e-01],
       [1.00000000e-05, 1.00000000e+00]])
```

In [8]:
```
set_printoptions(precision=17, suppress=True)

x = r_[1, .1, .01, .001, .0001, .00001]
y = sin(x)/x
c_[x,y]
```

Out[8]:
```
array([[1.                 , 0.8414709848078965],
       [0.1                , 0.9983341664682815],
       [0.01               , 0.9999833334166665],
       [0.001              , 0.9999998333333416],
       [0.0001             , 0.9999999983333334],
       [0.00001            , 0.9999999999833332]])
```

C6. **RUN** the following. See that we get the same output in C6 and C5, but the code is cleaner in C6 because we make x in a more clever way.

In [9]:
```
x = .1 ** r_[:6]
y = sin(x)/x
c_[x,y]
```

Out[9]:
```
array([[1.                 , 0.8414709848078965],
       [0.1                , 0.9983341664682815],
       [0.01               , 0.9999833334166665],
       [0.001              , 0.9999998333333416],
       [0.0001             , 0.9999999983333334],
       [0.00001            , 0.9999999999833332]])
```

C7. **EXERCISE.**

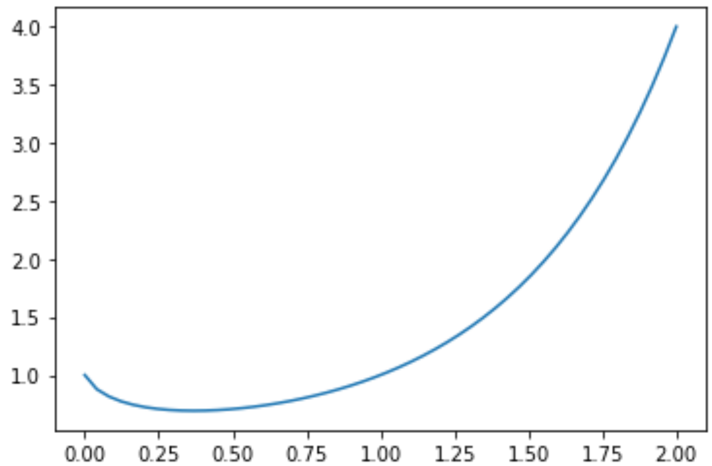
(1) Use the graphical approach to find the following right limit of $f(x) = x^x$, $x > 0$,

$$\lim_{x \to 0^+} x^x.$$

(2) What is the value of the limit? (Enter no limit as DNE)

In [10]:
```
# (1) Type your code below and press SHIFT+ENTER

x = linspace(0,2) # Or any interval starting at 0
y = x ** x
plot(x,y)

# (2) YOUR ANSWER: the limit is 1
```

Out[10]: `[<matplotlib.lines.Line2D at 0x7feb701200d0>]`



C8. **EXERCISE.**

(1) Use the numeric (table) approach to find the following (two sided) limit

$$\lim_{x \to 0} \frac{1 - \cos x}{x^2}.$$

You will need a table for the right-hand limit and a table for the left-hand limit.

(2) What is the value of the limit?

(3) How does the numerical instability of the problem show up in the output?

In [11]:
```
# (1) Type your code below and press SHIFT+ENTER

x = 0.1 ** r_[:10]
y = (1 - cos(x))/x**2
c_[x,y]
```

Out[11]:
```
array([[1.          , 0.45969769413186023],
       [0.1         , 0.49958347219742893],
       [0.01        , 0.4999958333473662 ],
       [0.001       , 0.4999999583255031 ],
       [0.0001      , 0.49999999696126435],
       [0.00001     , 0.5000000413701853 ],
       [0.000001    , 0.5000444502911701 ],
       [0.0000001   , 0.4996003610813201 ],
       [0.00000001  , 0.                 ],
       [0.000000001 , 0.                 ]])
```

In [12]:
```
x = -0.1 ** r_[:10]
y = (1 - cos(x))/x**2
c_[x,y]
```

Out[12]:
```
array([[-1.                , 0.45969769413186023],
       [-0.1               , 0.49958347219742893],
       [-0.01              , 0.4999958333473662 ],
       [-0.001             , 0.4999999583255031 ],
       [-0.0001            , 0.49999999696126435],
       [-0.00001           , 0.5000000413701853 ],
       [-0.000001          , 0.5000444502911701 ],
       [-0.0000001         , 0.4996003610813201 ],
       [-0.00000001        , 0.                 ],
       [-0.000000001       , 0.                 ]])
```

In [13]:
```
# (2) YOUR ANSWER: the limit is 0.5

# (3) YOUR ANSWER: when x gets too close to 0
#                  the computer starts "running out of digits"
#                  and starts giving incorrect f(x)
```

C9. **EXERCISE.**

(1) Use the numeric approach to find the following limit

$$\lim_{x \to \pi/2} \left(\frac{\pi}{2} - x \right) \tan x.$$

(2) What is the value of the limit?

In [14]:
```
# (1) Type your code below and press SHIFT+ENTER

x = pi/2 + 0.1 ** r_[:10]
y = (pi/2 - x)*tan(x)
c_[x,y]
```

Out[14]:
```
array([[2.5707963267948966, 0.6420926159343308],
       [1.6707963267948966, 0.9966644423259243],
       [1.5807963267948966, 0.9999666664444484],
       [1.5717963267948964, 0.9999996666667056],
       [1.5708963267948965, 0.999999996667279 ],
       [1.5708063267948966, 0.9999999999727899],
       [1.5707973267948965, 1.000000000060899 ],
       [1.5707964267948966, 1.0000000006123202],
       [1.5707963367948965, 1.0000000061232341],
       [1.5707963277948966, 1.0000000612323385]])
```

In [15]:
```
x = pi/2 - 0.1 ** r_[:10]
y = (pi/2 - x)*tan(x)
c_[x,y]
```

Out[15]:
```
array([[0.5707963267948966, 0.6420926159343306],
       [1.4707963267948965, 0.9966644423259232],
       [1.5607963267948965, 0.9999666664444362],
       [1.5697963267948967, 0.9999996666665832],
       [1.5706963267948966, 0.9999999966660543],
       [1.5707863267948965, 0.9999999999605436],
       [1.5707953267948966, 0.9999999999384342],
       [1.5707962267948965, 0.9999999993876733],
       [1.5707963167948966, 0.9999999938767661],
       [1.5707963257948965, 0.9999999387676689]])
```

In [16]:
```
# (2) YOUR ANSWER: the limit is 1
```

C10. **EXERCISE.**

Let $f(x) = x \ln x$ for $x > 0$. We will investigate the right-hand limit $\lim_{x \to 0^+} f(x)$ as follows.
(1) Use the numeric approach to find the limit.
(2) What is the value of the limit?
(3) Plot a graph of $f(x)$ over the interval $(0, 1)$.
(4) Does the graph in (3) confirm your answer in (2)?

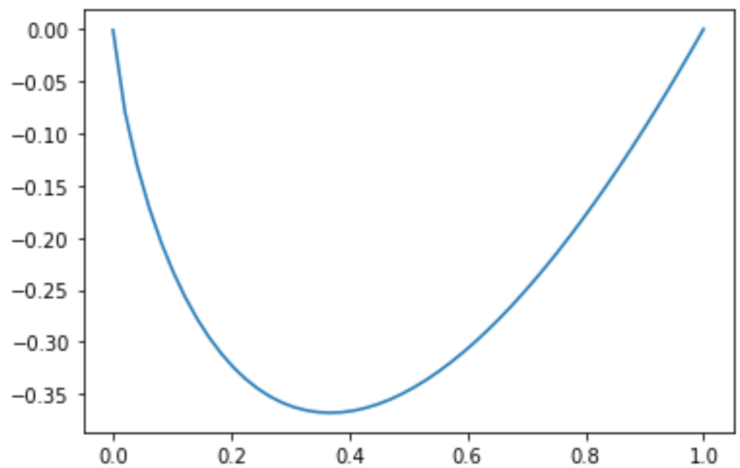
In [17]:
```
# (1) Type your code below and press SHIFT+ENTER

x = 0.1 ** r_[:17]
y = x*log(x)
c_[x,y]
```

Out[17]:
```
array([[ 1.                ,  0.                ],
       [ 0.1               , -0.2302585092994046],
       [ 0.01              , -0.0460517018598809],
       [ 0.001             , -0.0069077552789821],
       [ 0.0001            , -0.0009210340371976],
       [ 0.00001           , -0.000115129254649 ],
       [ 0.000001          , -0.0000138155105580],
       [ 0.0000001         , -0.000001611809565 ],
       [ 0.00000001        , -0.0000001842068074],
       [ 0.000000001       , -0.0000000207232658],
       [ 0.0000000001      , -0.000000002302585 ],
       [ 0.00000000001     , -0.0000000002532844],
       [ 0.000000000001    , -0.0000000000276310],
       [ 0.0000000000001   , -0.0000000000029934],
       [ 0.00000000000001  , -0.0000000000003224],
       [ 0.000000000000001 , -0.0000000000000345],
       [ 0.0000000000000001, -0.0000000000000037]])
```

In [18]:
```
# (2) YOUR ANSWER: the limit is 0
```

In [19]:
```
# (3) Type your code below and press SHIFT+ENTER

x = linspace(0.0001,1)
y = x*log(x)
plot(x,y)

# (4) YOUR ANSWER: yes, the graph also shows that the limit at 0 is 0
```

Out[19]: [<matplotlib.lines.Line2D at 0x7feb701f7b10>]



C11. **EXERCISE.**
We wish to find the limit of the oscillating function

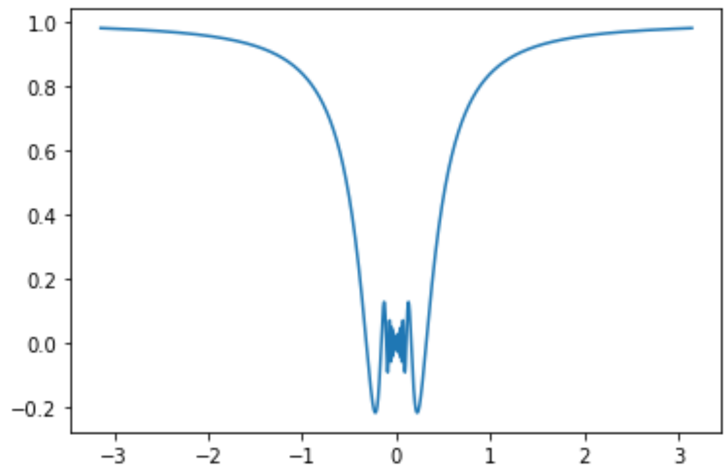
$$f(x) = x \sin \frac{1}{x}$$

as x approaches 0.

(1) Plot the function f over the interval $[-pi, pi]$ using 10000 points for x.
(2) "Zoom in" by replotting the function over $[-0.1, 0.1]$.
(3) Motivated by the squeeze theorem, plot on the same axes the function f over $[-0.1, 0.1]$ as well as $y = |x|$ and $y = -|x|$.
(4) Estimate the limit as $x \to 0$.
(5) How did graphing the absolute value help you find the limit in (3)? (Choose one)
The mean-value theorem   *The function is continuous*   *The squeeze theorem*   *They didnt; I just guessed*

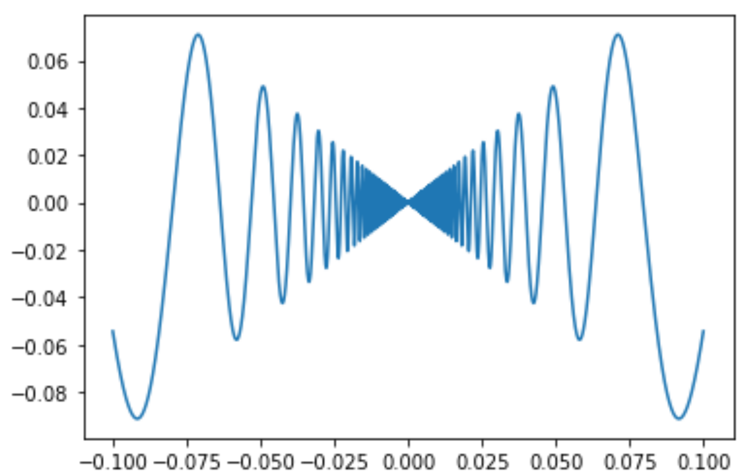
In [20]:
```
# (1) Type your code below and press SHIFT+ENTER

x = linspace(-pi,pi,10000)
y = x*sin(1/x)
plot(x,y)
```

Out[20]: [<matplotlib.lines.Line2D at 0x7feb702faf90>]



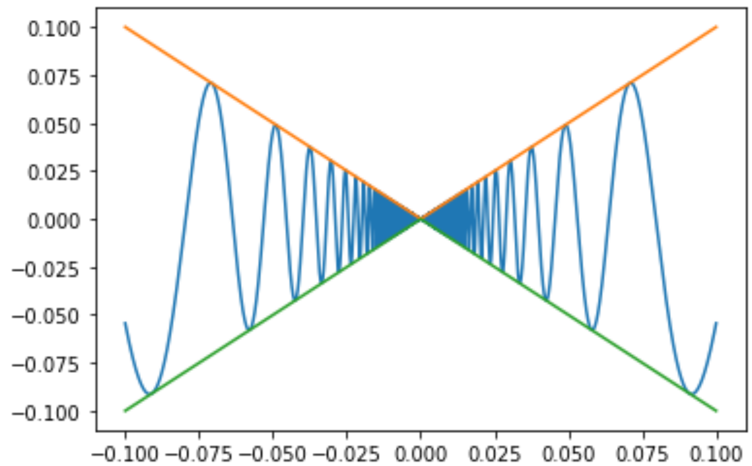
In [21]:
```
# (2) Type your code below and press SHIFT+ENTER

x = linspace(-0.1,0.1,10000)
y = x*sin(1/x)
plot(x,y)
```

Out[21]: [<matplotlib.lines.Line2D at 0x7feb70409350>]



In [22]:
```
# (3) Type your code below and press SHIFT+ENTER

x = linspace(-0.1,0.1,10000)
y = x*sin(1/x)
plot(x,y)

y = abs(x)
plot(x,y)

y = -abs(x)
plot(x,y)

# (4) YOUR ANSWER: the limit is 0
# (5) YOUR ANSWER: The squeeze theorem
```

Out[22]: [<matplotlib.lines.Line2D at 0x7feb70710a10>]



Functions

C12. **RUN** the following. We learned how to make python functions in A16. Here we define

$$f(x) = \frac{\sin x}{x}$$

$$g(x) = 7x^2 \sin \frac{1}{7x^2}$$

and then use our functions to find $f(1)$ and $g(1)$.

In [23]:
```
def f(x):
    return sin(x)/x

def g(x):
    a = 7*x**2
    return a * sin(1/a)

f(1), g(1)
```

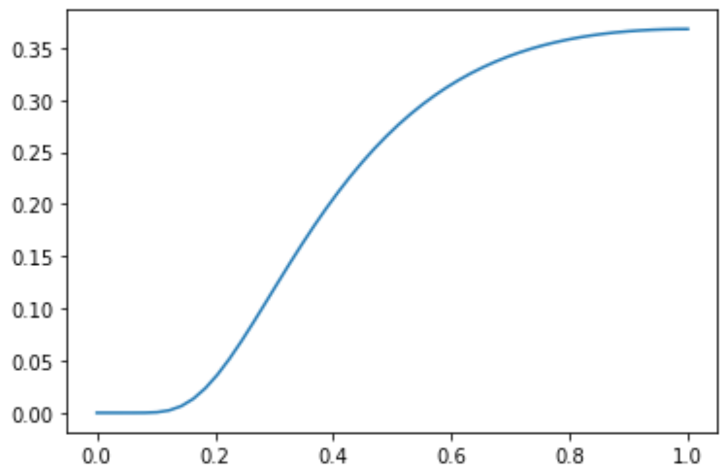
Out[23]: (0.8414709848078965, 0.9966021085458455)

C13. **EXERCISE.**

(1) Define the python function $f(x) = x^{-1}e^{-1/x}$
(2) Plot the function $f(x)$ over the interval $(0, 1]$ using your python function `f`
(3) Use your graph in (2) to estimate the right limit as x goes to 0.

In [24]:
```
# (1) Type your code below and press SHIFT+ENTER

def f(x):
    return x**-1 * exp(-1/x)

# (2) Type your code below and press SHIFT+ENTER

x = linspace(0.0001,1)
plot(x,f(x))

# (3) YOUR ANSWER: the limit is 0
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x7feb708a29d0>]
```



C14. **EXERCISE.**

(1) Create python functions with the following definitions.

$$f(x) = \frac{x-1}{\arccos x}$$

$$g(x) = x^x$$

(2) Make a table to find the limit

$$\lim_{x \to 1^-} g(f(x))$$

(3) What is the value of the limit in (2)?

(4) Does the limit in (1) satisfy the following limit law? If so, what are c, L, M?

If $\lim_{x \to c} f(x) = L$
and $\lim_{x \to L} g(x) = M$
then $\lim_{x \to c} g(f(x)) = M$

In [25]:
```
# (1) Type your code below and press SHIFT+ENTER

def f(x): return (x - 1)/arccos(x)
def g(x): return (1+x)**(1/x)

# (2) Type your code below and press SHIFT+ENTER

x = 1 - .1 ** r_[:10]
c_[x, g(f(x))]

# (3) YOUR ANSWER: the limit is e
# (4) YOUR ANSWER: c, L, M = 1, 0, e
```

Out[25]:
```
array([[0.                , 4.9043659284297805],
       [0.9               , 3.0973987898432522],
       [0.99              , 2.8209767308891007],
       [0.999             , 2.7493070339386976],
       [0.9999            , 2.7279550422197847],
       [0.99999           , 2.721327200112239 ],
       [0.999999          , 2.7192435094962595],
       [0.9999999         , 2.718585803911689 ],
       [0.99999999        , 2.7183779404648525],
       [0.999999999       , 2.718312220396477 ]])
```

C15. **EXERCISE.**
(1) Define $f(x) = x^x$.
(2) Make a table to find

$$f'(x) := \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

for $x = 1$.

(3) What value for $f'(1)$ do you get?
(4) Make the same table as in (2) but with $x = 0$.
(5) What value for $f'(0)$ do you get?
(6) Are you surprised that 43) worked? What part of (4) do we expect to fail, and what did python do instead?

In [26]:
```
# (1) Type your code below and press SHIFT+ENTER

def f(x):
    return x**x

# (2) Type your code below and press SHIFT+ENTER

h = .1 ** r_[:10]

c_[h, (f(1+h) - f(1))/h]

# (3) YOUR ANSWER: f'(1) = 1
```

Out[26]:
```
array([[1.                  , 3.                  ],
       [0.1                 , 1.1053424105457577],
       [0.01                , 1.0100503341741616],
       [0.001               , 1.0010005003333597],
       [0.0001              , 1.0001000049997264],
       [0.00001             , 1.0000100000517873],
       [0.000001            , 1.0000010000066335],
       [0.0000001           , 1.000000100503939 ],
       [0.00000001          , 0.9999999939225285],
       [0.000000001         , 1.0000000827403706]])
```

In [27]:
```
# (4) Type your code below and press SHIFT+ENTER

h = .1 ** r_[:10]

c_[h, (f(0+h) - f(0))/h]

# (5) YOUR ANSWER: f'(0) = -oo
```

Out[27]:
```
array([[  1.                ,   0.               ],
       [  0.1               ,  -2.056717652757185],
       [  0.01              ,  -4.5007413978564  ],
       [  0.001             ,  -6.883951579066181],
       [  0.0001            ,  -9.206100155382256],
       [  0.00001           , -11.512262753143872],
       [  0.000001          , -13.815415124240888],
       [  0.0000001         , -16.118082660776516],
       [  0.00000001        , -18.42067904878063 ],
       [  0.000000001       , -20.723265659050572]])
```

In [28]:
```
# (6) YOUR ANSWER: Yes, I am surprised that (3) worked,
#                  because f(0) = 0**0 should be indeterminant
#                  Instead, python computes 0**0 to be 1:

0**0
```

Out[28]:
```
1
```