

## 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.**

- What does each of the 5 arithmetic operations do?
- 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 around + - * / ** 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  $3 - \frac{3^2-2\cdot 3}{2\cdot 3-2}$

```
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  $\frac{a^b - c/b}{c - a}, \frac{a^{c-b}}{c - b}, \frac{a^{3/2}}{b}, \frac{a - b(c - a)}{c - a}$

```
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
<code>abs(x)</code>	$ x $	absolute value
<code>sqrt(x)</code>	$\sqrt{x}$	square root
<code>exp(x)</code>	$e^x$	exponential function
<code>log(x)</code>	$\ln x$	natural logarithm
<code>sin(x)</code>	$\sin x$	sine
<code>arcsin(x)</code>	$\sin^{-1} x$	inverse sine
<code>radians(x)</code>		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,...,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, ..., 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.

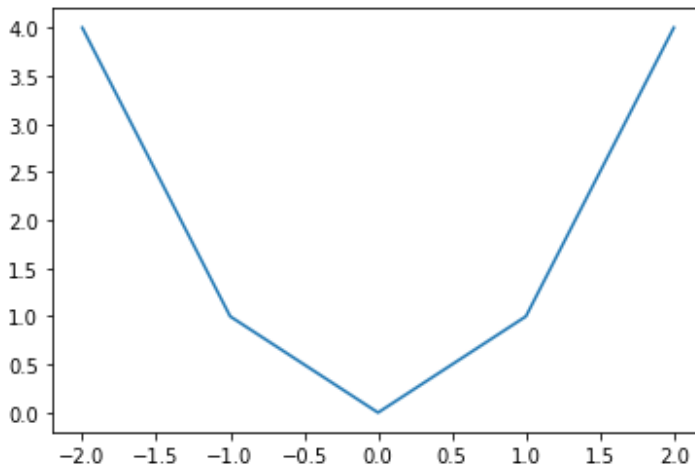
Note to instructor: you may remember that in A18 we wrote `%pylab` and here we write `%pylab inline`; the "inline" tells Jupyter to display images inline instead of as a pop-up.

```
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>]
```



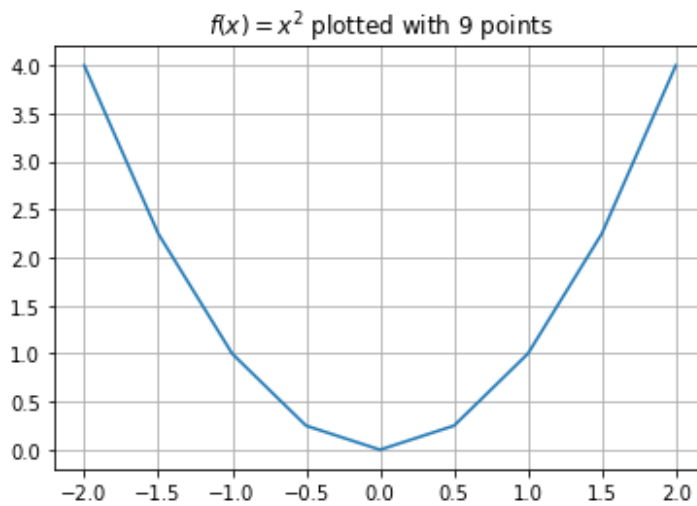
B3. **RUN** the following. Notice that we save time by making the  $x$  array using `linspace` (see A27) and making the  $y$  array by doing arithmetic on  $x$  (see A29). For illustrative purposes, we use `c_[x,y]` to make a table out of the arrays  $x$  and  $y$  (see A29).

```
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]
```

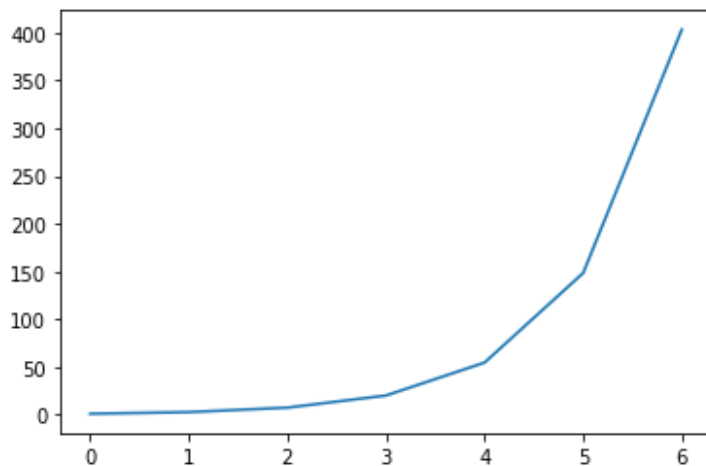
```
Out[3]: array([[ -2.   ,  4.   ],
               [ -1.5   ,  2.25 ],
               [ -1.   ,  1.   ],
               [ -0.5   ,  0.25 ],
               [  0.   ,  0.   ],
               [  0.5   ,  0.25 ],
               [  1.   ,  1.   ],
               [  1.5   ,  2.25 ],
               [  2.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

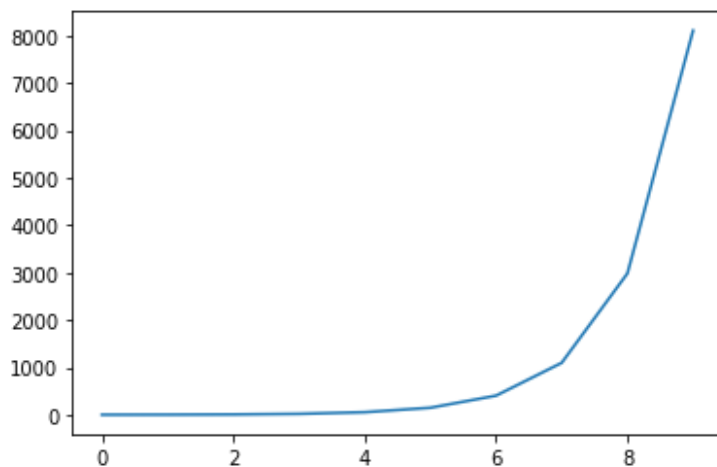
```
Out[4]: [<matplotlib.lines.Line2D at 0x7f9e42a15a90>]
```



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>]
```



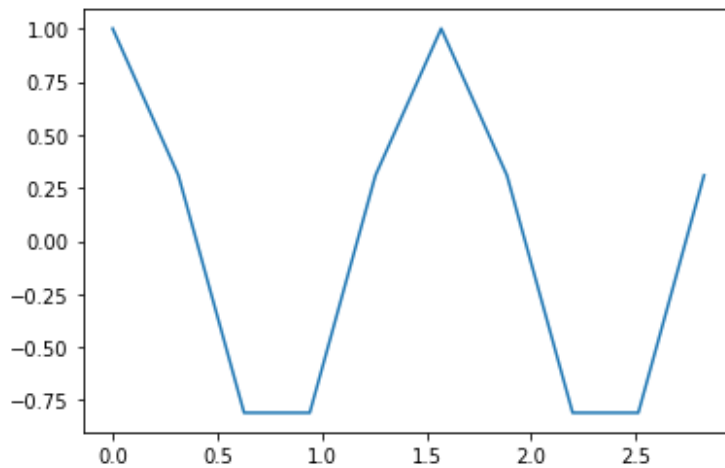
#### B6. EXERCISE.

- (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>]

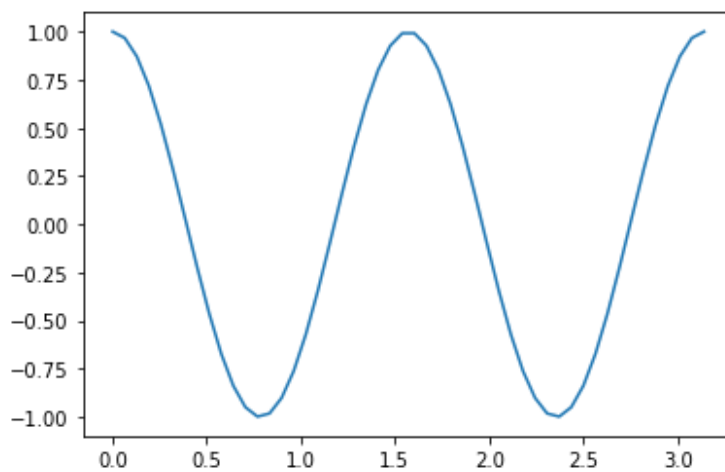


In [7]: # (2) Type your answer below and press SHIFT+ENTER

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

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

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

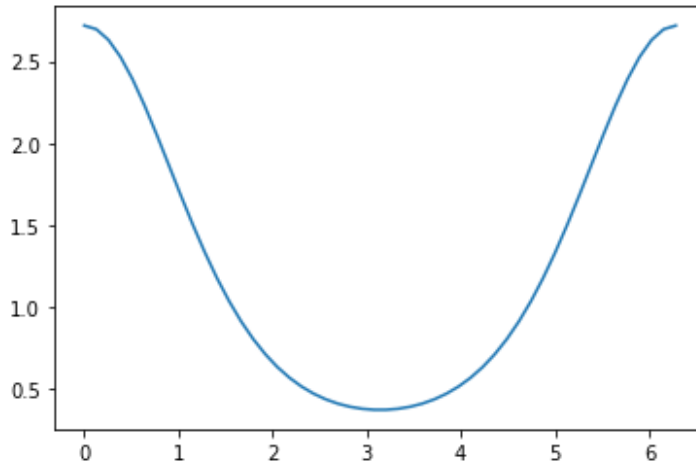


B7. **EXERCISE.** Plot the function  $f(x) = e^{\cos x}$  over the interval  $[0, 2\pi]$ .

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

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

Out[8]: [



## Doing arithmetic 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

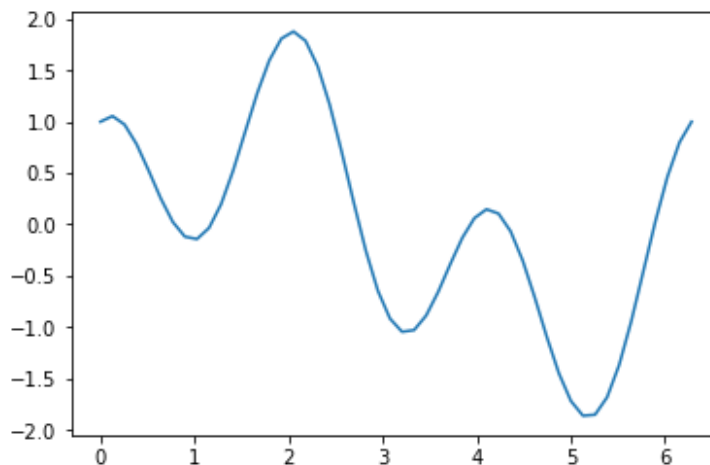
ValueError: operands could not be broadcast together with shapes (4,) (3,)
```

B10. **RUN** the following.

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

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

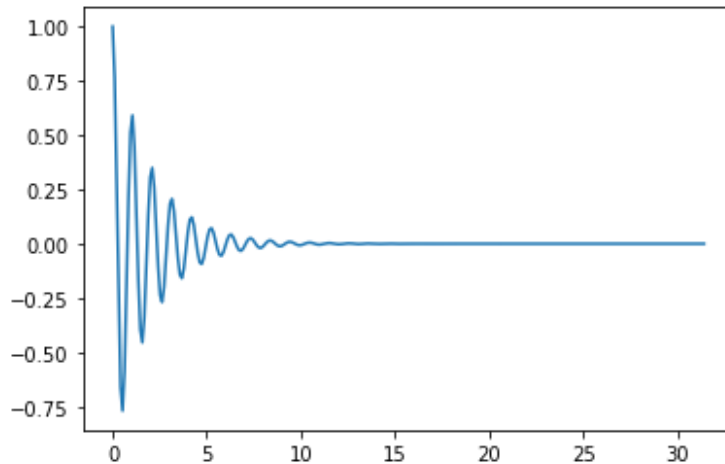
```
Out[13]: [matplotlib.lines.Line2D at 0x7f9e43060090]
```



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

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)
```

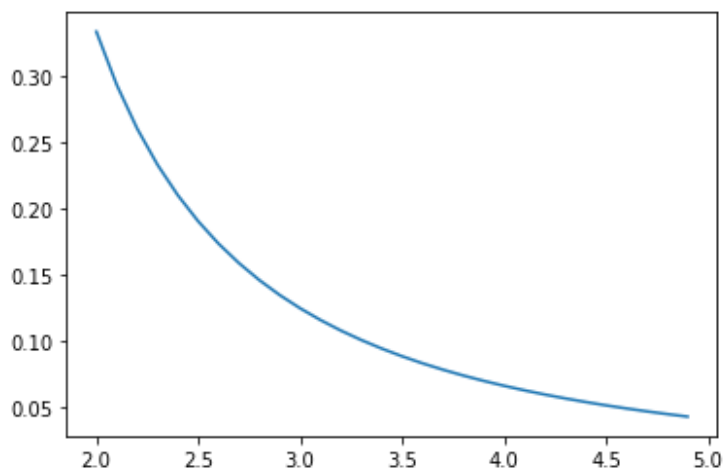
Out[14]: [



```
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]: [



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?

$a + b$        $a + c$        $a + 1$        $a * b$        $c ** 2$        $c ^ 2$

```
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:

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

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:

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

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.**

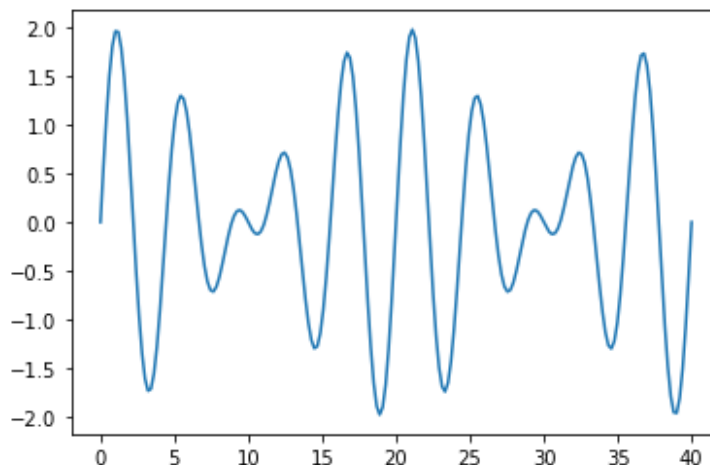
- (1) Graph the function  $f(x) = \sin(\frac{\pi}{2}x) + \sin(\frac{2}{5}\pi x)$  over the interval  $[0, 40]$ .
- (2) How many peaks (relative maxima) does your graph have?
- (3) This function is periodic; how many periods are graphed in  $[0, 40]$ ?
- (4) Estimate from your graph the value of  $f(10)$  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.

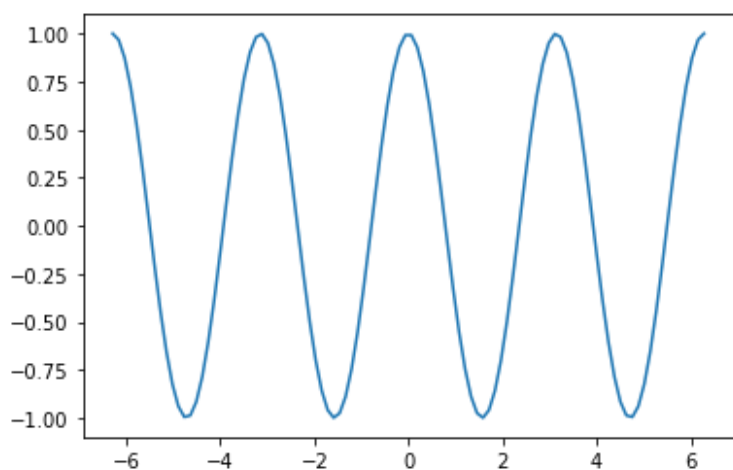
(2) Does the resemble any of the following?       $\cos 2x$        $\cos x/2$        $\cos x$

```
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?

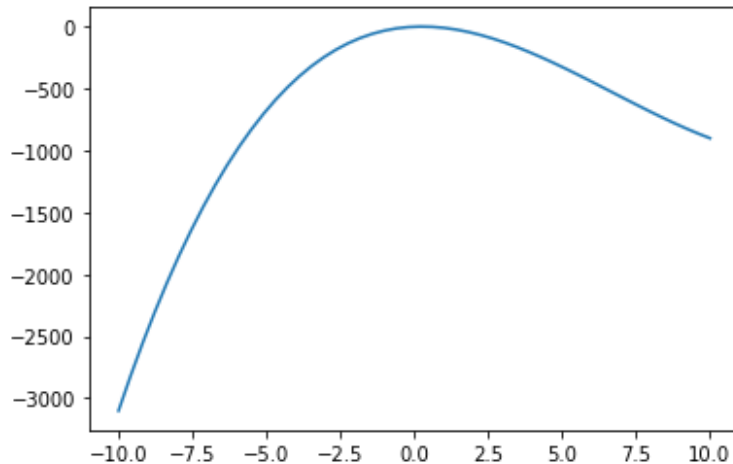
$[-10, 10]$        $(-10, 10)$        $[-3100, 0]$        $[0, 2\pi]$

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

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

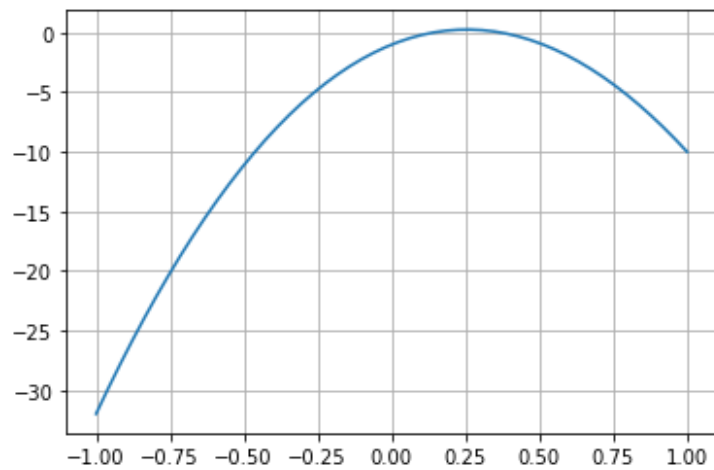
(3) From your graph, which of these  $x$  values have  $f(x) > 0$ ? Indicate all that apply:

0      0.25      0.50      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

C1. *RUN the following.*

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

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

The magic word `%pylab` loads a bunch of useful functions (including `pi` and `sin`).

The option `inline` tells Jupyter to display any graphs we make on the page instead of in a popup.

C2. Example. Let's say we want to find  $\lim_{x \rightarrow \pi/2} \frac{\sin x}{x}$ . The first thing we try is plugging in.

*RUN the following.*

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

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

C3. **DO EXERCISE.** Based on the result in C2, what is the value of the limit  $\lim_{x \rightarrow \pi/2} \frac{\sin x}{x}$ ?

```
In [3]: # RECORD YOUR ANSWER: the limit is 0.6366197723675814
```

C4. Example. Let's say we want to find  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$  instead.

*RUN the following.*

```
In [4]: x = 0  
sin(x)/x
```

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

```
Out[4]: nan
```

Notice that plugging in doesn't work, because  $\sin(0)/0$  is undefined.  
(Instead of "undefined", Python says "nan," which means "not a number.")

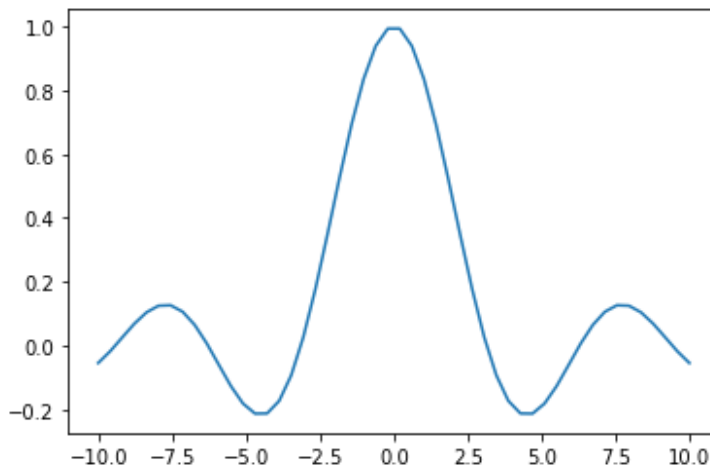
## Find the limit with a graph

C5. When plugging in doesn't work, one thing we can try is graphing.

*RUN the following.*

```
In [5]: x = linspace(-10,10)
        y = sin(x)/x
        plot(x,y)
```

```
Out[5]: [<matplotlib.lines.Line2D at 0x7fed1379aa90>]
```



C6. **DO EXERCISE.** Based on the graph in C5, what is the value of the limit  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$ ?

```
In [6]: # RECORD YOUR ANSWER: the limit is 1
```

## Find the limit with a table

C7. The other thing we can try is making a table.

*RUN the following.*

```
In [7]: x = r_[1, .1, .01, .001] # Make array of numbers approaching 0 from right
        y = sin(x)/x             # Plug the array into our function
        c_[x,y]                  # Make a table
```

```
Out[7]: array([[1.          , 0.84147098],
               [0.1        , 0.99833417],
               [0.01       , 0.99998333],
               [0.001      , 0.99999983]])
```

```
In [8]: x = r_[-1, -.1, -.01, -.001] # Same but approaching from left
        y = sin(x)/x
        c_[x,y]
```

```
Out[8]: array([[ -1.          , 0.84147098],
               [-0.1        , 0.99833417],
               [-0.01       , 0.99998333],
               [-0.001      , 0.99999983]])
```

Note that to display two tables, we use two code cells, because Jupyter only displays one output per cell.

C8. **DO EXERCISE.** Based on the table in C7, what is the value of the limit  $\lim_{x \rightarrow 0} \frac{\sin x}{x}$ ?

```
In [9]: # RECORD YOUR ANSWER: the limit is 1
```

## Simplify the code

C9. We can simplify the code for `x` using an array.

*RUN the following.*

```
In [10]: x = .1 ** r_[ :4]
         y = sin(x)/x
         c_[x,y]
```

```
Out[10]: array([[1.          , 0.84147098],
                [0.1        , 0.99833417],
                [0.01       , 0.99998333],
                [0.001      , 0.99999983]])
```

You should get the same table as in C7.

Why does this work? Because `.1 ** r_[ :4]` means "0.1 raised to the 0, 1, 2, 3," which is going to be 1, 0.1, 0.01, 0.001.



## Increase precision

C10. The arrays we have made so far show (up to) 8 decimal places.

As an example, let's change the number of decimal places displayed to 17 decimal places:

*RUN the following.*

```
In [11]: set_printoptions(precision=17)
```

```
x = .1 ** r_[:4]
y = sin(x)/x
c_[x,y]
```

```
Out[11]: array([[1.          , 0.8414709848078965],
                [0.1        , 0.9983341664682815],
                [0.01       , 0.999833334166665 ],
                [0.001      , 0.9999983333333416]])
```

After you run this cell, from now on, arrays will display (up to) 17 decimal places.

## Turn off scientific notation

C11. Arrays with very small numbers (less than 0.001) will automatically change to scientific notation.

*RUN the following.*

```
In [12]: x = .1 ** r_[:5]
y = sin(x)/x
c_[x,y]
```

```
Out[12]: array([[1.0000000000000000e+00, 8.4147098480789650e-01],
                [1.0000000000000001e-01, 9.9833416646828155e-01],
                [1.0000000000000002e-02, 9.9998333341666645e-01],
                [1.0000000000000002e-03, 9.9999983333333416e-01],
                [1.0000000000000002e-04, 9.9999999833333342e-01]])
```

C12. We can turn off scientific notation for arrays using `set_printoptions`

*RUN the following.*

```
In [13]: set_printoptions(suppress=True)
```

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

```
Out[13]: array([[1.          , 0.8414709848078965],
                [0.1        , 0.9983341664682815],
                [0.01       , 0.9999833334166665],
                [0.001      , 0.9999998333333416],
                [0.0001     , 0.9999999983333334]])
```

### C13. DO EXERCISE.

(1) Use the graphical approach to find  $\lim_{x \rightarrow 0^+} x^x$ .

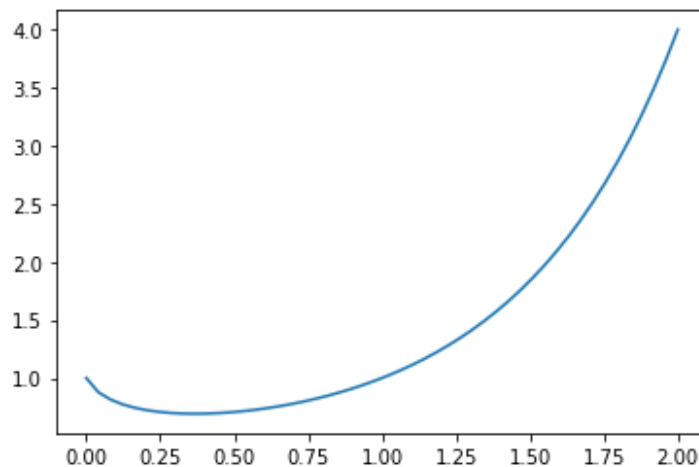
(2) Use the numerical approach (make a table) for the same limit.

```
In [14]: # (1) TYPE YOUR CODE:
```

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

# RECORD YOUR ANSWER: the limit is 1
```

```
Out[14]: [<matplotlib.lines.Line2D at 0x7fed13905d10>]
```



```
In [15]: # (2) TYPE YOUR CODE:
```

```
x = 0.1 ** r_[:10]
y = x**x
c_[x,y]
```

```
# RECORD YOUR ANSWER: the limit is 1
```

```
Out[15]: array([[1.          , 1.          ],
 [0.1          , 0.7943282347242815],
 [0.01         , 0.954992586021436 ],
 [0.001        , 0.9931160484209338],
 [0.0001       , 0.9990793899844618],
 [0.00001      , 0.9998848773724686],
 [0.000001     , 0.9999861845848758],
 [0.0000001    , 0.9999983881917339],
 [0.00000001   , 0.9999998157932095],
 [0.000000001  , 0.9999999792767343]])
```

## Define a function

C14. Example. Let's make a graph and make tables to find  $\lim_{x \rightarrow 0} \frac{1 - \cos x}{x^2}$

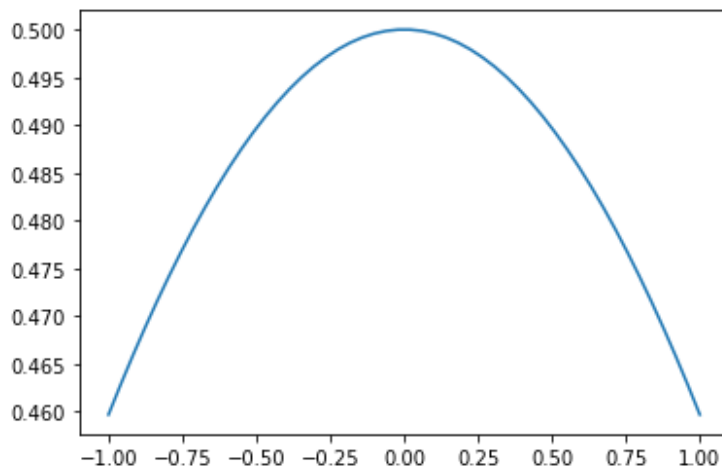
We will use `def` to store the function so we don't have to type the function over and over.

*RUN the following.*

```
In [16]: def f(x):
         return (1 - cos(x))/x**2
```

```
In [17]: x = linspace(-1,1)
         y = f(x)
         plot(x,y)
```

```
Out[17]: [<matplotlib.lines.Line2D at 0x7fed139f2e90>]
```



```
In [18]: x = 0.1 ** r_[:8]
y = f(x)
c_[x,y]
```

```
Out[18]: 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 ]])
```

```
In [19]: x = -0.1 ** r_[:8]
y = f(x)
c_[x,y]
```

```
Out[19]: 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 ]])
```

So, the graph and the tables tell us that  $\lim_{x \rightarrow 0} \frac{1 - \cos x}{x^2} = 0.5$

## Taking the limit as $x$ approaches a number other than zero

C15. Example. Let's investigate  $\lim_{x \rightarrow \pi/2} \left( \frac{\pi}{2} - x \right) \tan x$

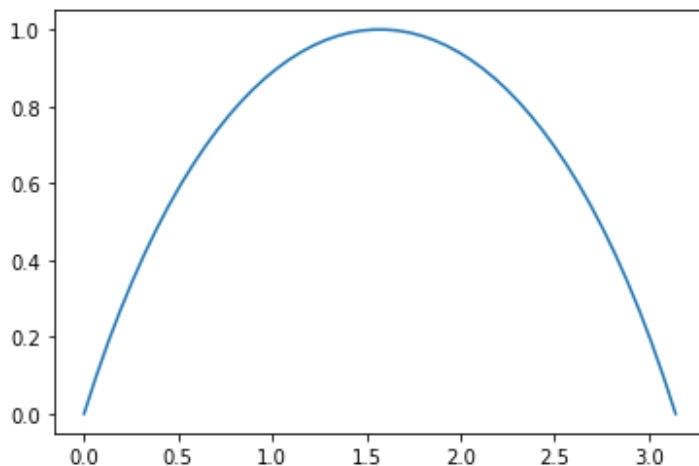
Notice that in this example,  $x$  is not approaching 0.

*RUN the following.*

```
In [20]: def f(x):
          return (pi/2 - x)*tan(x)
```

```
In [21]: x = linspace(0,pi)
y = f(x)
plot(x,y)
```

```
Out[21]: [<matplotlib.lines.Line2D at 0x7fed13adead0>]
```



```
In [22]: x = pi/2 + 0.1 ** r_[:8]
y = f(x)
c_[x,y]
```

```
Out[22]: array([[2.5707963267948966, 0.6420926159343308],
 [1.6707963267948966, 0.9966644423259243],
 [1.5807963267948966, 0.9999666664444484],
 [1.5717963267948964, 0.9999996666667056],
 [1.5708963267948965, 0.999999996667279 ],
 [1.5708063267948966, 0.999999999727899],
 [1.5707973267948965, 1.000000000060899 ],
 [1.5707964267948966, 1.00000000006123202]])
```

```
In [23]: x = pi/2 - 0.1 ** r_[:8]
y = f(x)
c_[x,y]
```

```
Out[23]: array([[0.5707963267948966, 0.6420926159343306],
 [1.4707963267948965, 0.9966644423259232],
 [1.5607963267948965, 0.9999666664444362],
 [1.5697963267948967, 0.9999996666665832],
 [1.5706963267948966, 0.9999999966660543],
 [1.5707863267948965, 0.999999999605436],
 [1.5707953267948966, 0.999999999384342],
 [1.5707962267948965, 0.9999999993876733]])
```

The graph and the tables tell us that  $\lim_{x \rightarrow \pi/2} \left( \frac{\pi}{2} - x \right) \tan x = 1$

## More Exercises

### C16. DO EXERCISE.

Let  $f(x) = x \ln x$ .

(1) Plot a graph of  $f(x)$  over the interval  $(0, 1]$  and based on your graph, estimate

$$\lim_{x \rightarrow 0^+} f(x).$$

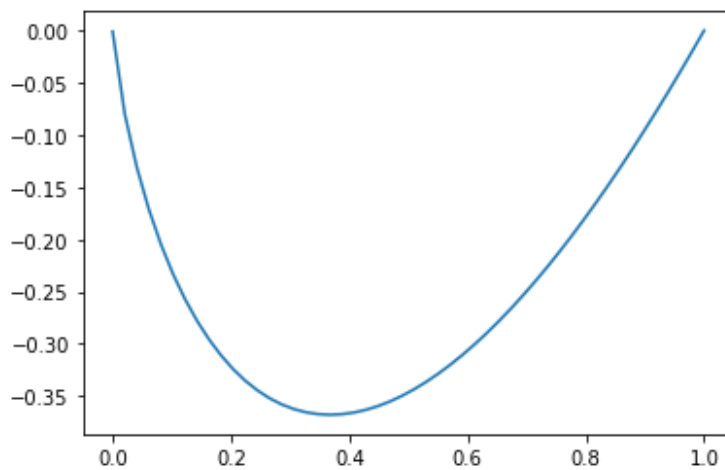
(2) Use the numeric approach to find the same limit.

In [24]: *# (1) TYPE YOUR CODE:*

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

*# RECORD YOUR ANSWER: the limit is 0*

Out[24]: [



```
In [25]: # (2) TYPE YOUR CODE:
```

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

```
# RECORD YOUR ANSWER: the limit is 0
```

```
Out[25]: array([[ 1.          ,  0.          ],
 [ 0.1          , -0.23025850929940456],
 [ 0.01         , -0.04605170185988092],
 [ 0.001        , -0.00690775527898214],
 [ 0.0001       , -0.00092103403719762],
 [ 0.00001      , -0.0001151292546497 ],
 [ 0.000001     , -0.00001381551055796],
 [ 0.0000001    , -0.0000016118095651 ],
 [ 0.00000001   , -0.00000018420680744],
 [ 0.000000001  , -0.00000002072326584],
 [ 0.0000000001 , -0.00000000230258509],
 [ 0.00000000001, -0.00000000025328436],
 [ 0.000000000001, -0.00000000002763102],
 [ 0.0000000000001, -0.00000000000299336],
 [ 0.00000000000001, -0.00000000000032236],
 [ 0.000000000000001, -0.00000000000003454],
 [ 0.0000000000000001, -0.00000000000000368]])
```

#### C17. DO EXERCISE.

(1) Define the python function  $f(x) = x^{-1}e^{-1/x}$

(2) Plot the function  $f(x)$  over the interval  $(0, 1]$  and estimate.

$$\lim_{x \rightarrow 0^+} f(x)$$

(3) Use the numeric approach to find the same limit.

```
In [26]: # (1) TYPE YOUR CODE:
```

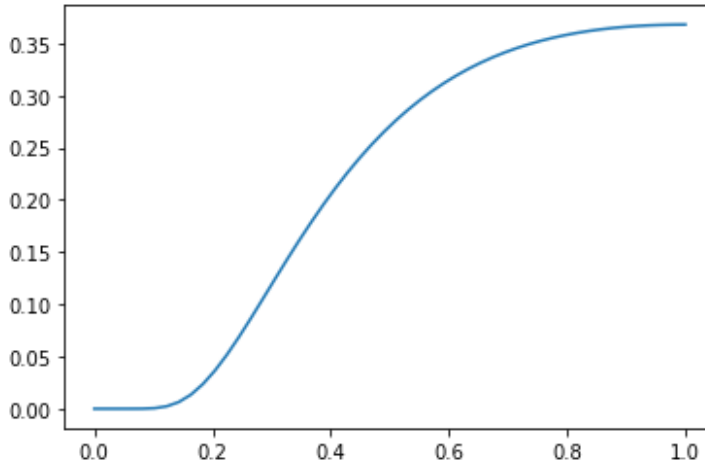
```
def f(x): return x**-1 * exp(-1/x)
```

In [27]: # (2) TYPE YOUR CODE:

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

# RECORD YOUR ANSWER: the limit is 0

Out[27]: [<matplotlib.lines.Line2D at 0x7fed13c21810>]



In [28]: # (3) TYPE YOUR CODE:

```
x = 0.1 ** r_[:5]
c_[x,f(x)]
```

# RECORD YOUR ANSWER: the limit is 0

Out[28]: array([[1. , 0.36787944117144233],  
 [0.1 , 0.00045399929762485],  
 [0.01 , 0. ],  
 [0.001 , 0. ],  
 [0.0001 , 0. ]])

### C18. DO 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  $x \rightarrow 0$ .
- (5) How did graphing the absolute value help you find the limit in (3)? (Choose one)

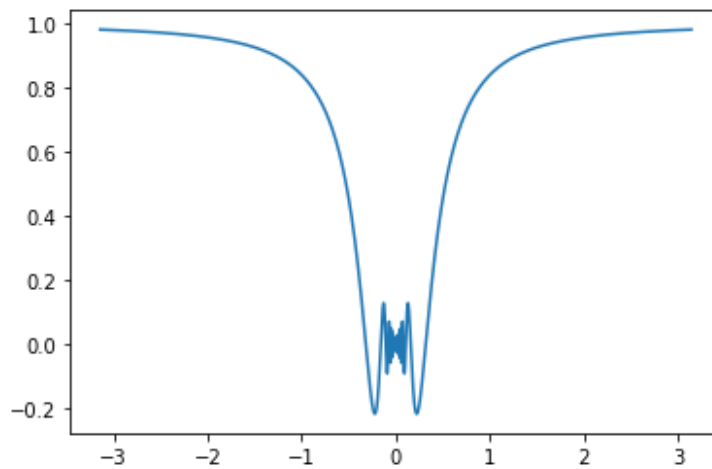
*The mean-value theorem    The function is continuous    The squeeze theorem    They didnt; I just guessed*



In [29]: # (1) TYPE YOUR CODE:

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

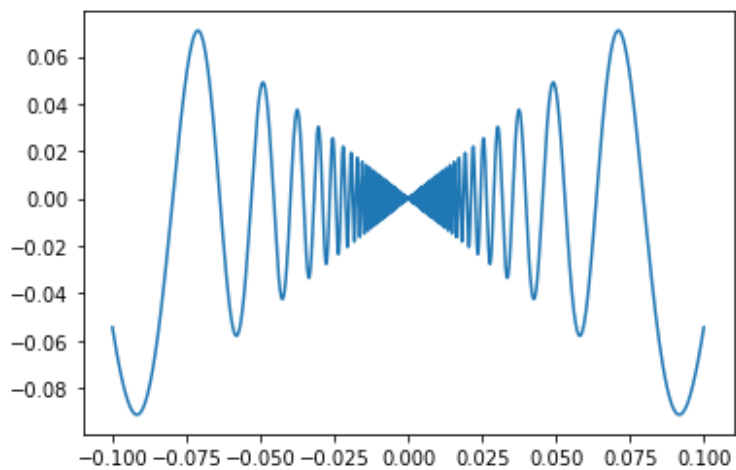
Out[29]: [<matplotlib.lines.Line2D at 0x7fed13d66b50>]



In [30]: # (2) TYPE YOUR CODE:

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

Out[30]: [<matplotlib.lines.Line2D at 0x7fed13e92b90>]



In [31]: # (3) TYPE YOUR CODE:

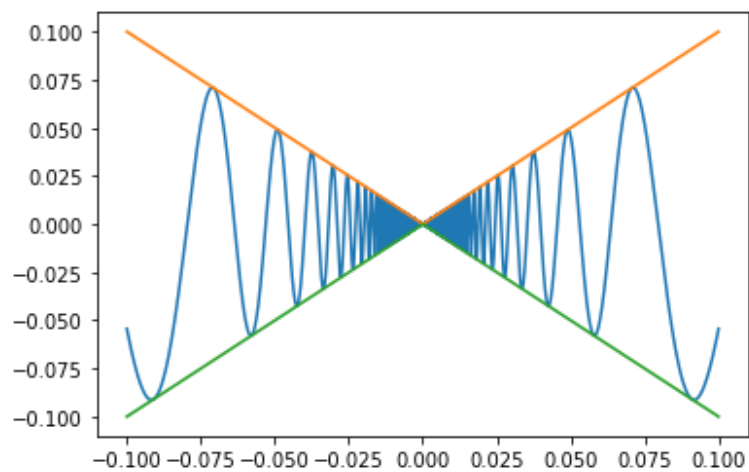
```
x = linspace(-0.1,0.1,10000)
y = x*sin(1/x)
plot(x,y)

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

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

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

Out[31]: [<matplotlib.lines.Line2D at 0x7fed140a3690>]



## Lab D: Finding First Derivatives in Python

<https://mybinder.org/v2/gh/anniebmcc/pycalclab/master?filepath=mat301d.ipynb>

2020 Summer — Calculus 1

Dr Matthew H Sunderland

D1. RUN the following.

```
In [1]: %pylab inline
        set_printoptions(precision=15)
        set_printoptions(suppress=True)
```

Populating the interactive namespace from numpy and matplotlib

D2. Warmup (1)–(3).

$$\text{Let } f(x) = \frac{x\sqrt{(x-1)^2}}{x-1} + \frac{\sqrt{x^2}}{x}$$

(1) Find  $\lim_{x \rightarrow 1^+} f(x)$

```
In [2]: # (1) TYPE YOUR CODE:

def f(x):
    return x*sqrt((x-1)**2)/(x-1) + sqrt(x**2)/x

x = 1 + 0.1**r_[1:10]
c_[x,f(x)]

# RECORD YOUR ANSWER: the limit is 2
```

```
Out[2]: array([[1.1      , 2.1      ],
               [1.01     , 2.01     ],
               [1.001    , 2.001    ],
               [1.0001   , 2.0001   ],
               [1.00001  , 2.00001  ],
               [1.000001 , 2.000001 ],
               [1.0000001, 2.0000001 ],
               [1.00000001, 2.00000001 ],
               [1.000000001, 2.000000001 ],
               [1.0000000001, 2.0000000001 ]])
```

(2) Find  $\lim_{x \rightarrow 1^-} f(x)$

```
In [3]: # (2) TYPE YOUR CODE:

x = 1 - 0.1**r_[1:10]
c_[x,f(x)]

# RECORD YOUR ANSWER: the limit is 0
```

```
Out[3]: array([[0.9          , 0.1          ],
               [0.99         , 0.01         ],
               [0.999        , 0.001        ],
               [0.9999       , 0.0001       ],
               [0.99999      , 0.00001      ],
               [0.999999     , 0.000001     ],
               [0.9999999    , 0.0000001    ],
               [0.99999999   , 0.00000001   ],
               [0.999999999  , 0.000000001  ],
               [0.9999999999 , 0.0000000001 ]])
```

(3) Based on your answer to (1) and (2), find  $\lim_{x \rightarrow 1} f(x)$

```
In [4]: # (3) RECORD YOUR ANSWER: the limit does not exist
```

## Computing the derivative at a point using the limit definition of derivative

D3. EXAMPLE. Let us approximate  $\left. \frac{d}{dx} x^x \right|_{0.6}$  to 4 decimal places.

In lecture we learned that limit definition of derivative is  $f'(x) := \lim_{h \rightarrow 0} \frac{f(x+h)-f(x)}{h}$

RUN the following.

```
In [5]: def f(x): return x**x

x = 0.6
h = 0.1 ** r_[:15] # = array of numbers going to 0
m = (f(x+h)-f(x))/h
c_[h,m] # make a table
```

```
Out[5]: array([[1.          , 1.385228648279758],
               [0.1        , 0.430339898526158],
               [0.01       , 0.367055009340134],
               [0.001      , 0.360744452604034],
               [0.0001     , 0.360113206078205],
               [0.00001    , 0.360050079117879],
               [0.000001   , 0.360043766467477],
               [0.0000001  , 0.360043135305688],
               [0.00000001 , 0.360043073133198],
               [0.000000001, 0.360043106439889],
               [0.0000000001, 0.360043106439889],
               [0.00000000001, 0.360045326885938],
               [0.000000000001, 0.360045326885938],
               [0.0000000000001, 0.35971225997855 ],
               [0.00000000000001, 0.35527136788005 ]])
```

⇒ The limit to 4 decimal places is 0.3600

D4. ASSIDE. Later on in calculus you will learn how to compute  $\frac{d}{dx}x^x \Big|_{0.6}$  algebraically. It turns out that

$$(x^x)' = (e^{x \ln x})' = e^{x \ln x} (\ln x + x/x) = x^x (\ln x + 1)$$

and so  $(x^x)'(0.6) = 0.6^{0.6}(\ln 0.6 + 1)$ .

RUN the following

```
In [6]: 0.6**0.6*(log(0.6) + 1)
```

```
Out[6]: 0.3600430649889697
```

So we see that  $(x^x)'(0.6) = 0.6^{0.6}(\ln 0.6 + 1) = 0.3600430649889697$ , which confirms our approximation in D3.

D5. **EXERCISE.** Approximate  $\frac{d}{dx}(\sin x) \Big|_{x = \pi/4}$  to 4 decimal places.

```
In [7]: # TYPE YOUR CODE:
```

```
def f(x): return sin(x)
```

```
x = pi/4
```

```
h = 0.1 ** r_[:15]
```

```
m = (f(x+h)-f(x))/h
```

```
c_[h,m]
```

```
# RECORD YOUR ANSWER:
```

```
# the limit to 4 decimal places is 0.7071
```

```
Out[7]: array([[1.          , 0.269954482712928],
               [0.1        , 0.67060297290399 ],
               [0.01       , 0.70355949168921 ],
               [0.001      , 0.706753109974256],
               [0.0001     , 0.707071424668193],
               [0.00001    , 0.707103245645157],
               [0.000001   , 0.707106427633164],
               [0.0000001  , 0.707106745378993],
               [0.00000001 , 0.707106773134569],
               [0.000000001, 0.70710681754349 ],
               [0.0000000001, 0.707106595498885],
               [0.00000000001, 0.707101044383762],
               [0.000000000001, 0.707101044383762],
               [0.0000000000001, 0.708322289710849],
               [0.00000000000001, 0.7105427357601 ]])
```

## Secant and tangent lines

The fraction inside the limit in the limit definition of derivative is called the "difference quotient"

$$\text{difference quotient} = \frac{f(x+h) - f(x)}{h}$$

The difference quotient is the slope of the secant line through the graph at  $x$  and  $x+h$ .

D6. EXAMPLE. Consider the same function as in [D3](#),  $f(x) = x^x$ . Let's plot  $f$  over  $[0, 2]$  and then plot the secant line for  $h = 1, 0.1, 0.01$ .

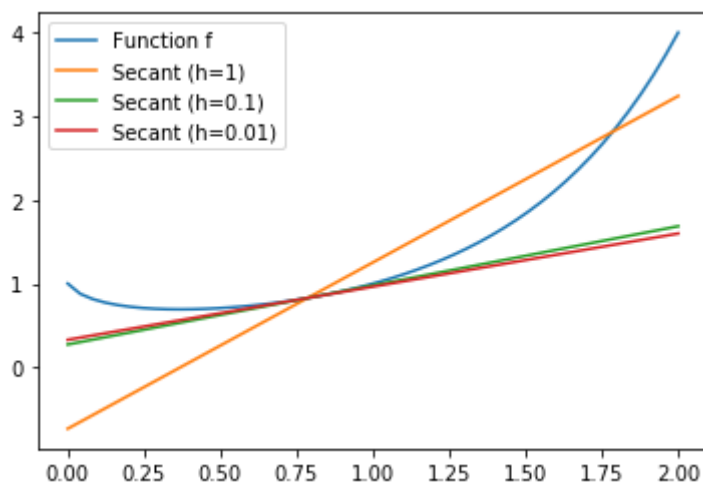
```
In [8]: def f(x): return x**x

x = linspace(0,2)
plot(x,f(x))

x0 = pi/4

for h in [1, 0.1, 0.01]:
    m = (f(x0+h) - f(x0))/h
    y = f(x0) + m*(x-x0)
    plot(x,y)

legend(['Function f', 'Secant (h=1)', 'Secant (h=0.1)', 'Secant (h=0.01)']);
```



We actually have already computed the slope of these 3 secant lines in [D3](#)!

Their slopes are:

1.385228648279758,  
0.430339898526158, and  
0.367055009340134.

As  $h$  approaches 0, the secant lines approach the tangent line, and the difference quotient (the slope of the secant line) approaches the derivative (the slope of the tangent line).

**D7. EXERCISE (1)–(2).** Consider the same function as in [D5](#),  $f(x) = \sin x$ .

(1) Plot  $f$  over  $[0, \pi]$ . and then plot the secant line for  $h = \pi/2, 1, 0.1$ .

(2) What is the slope of the three secant lines? Hint: look at the table you made in [D5](#).

In [9]: *# TYPE YOUR CODE:*

```
def f(x): return sin(x)

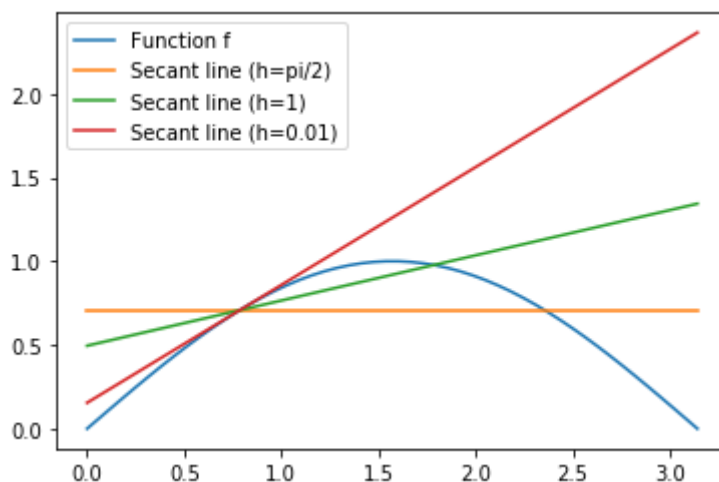
x = linspace(0,pi)
plot(x,f(x))

x0 = pi/4

for h in [pi/2, 1, 0.01]:
    m = (f(x0+h) - f(x0))/h
    y = f(x0) + m*(x-x0)
    plot(x,y)

legend([
    'Function f',
    'Secant line (h=pi/2)',
    'Secant line (h=1)',
    'Secant line (h=0.01)',
]);

# (2) RECORD YOUR ANSWER:
#     the slope of the 3 secant lines are
#     0,
#     0.269954482712928,
#     0.67060297290399
```



## Graphing the difference quotient

D8. EXAMPLE. Consider the same function as in [D3](#),  $f(x) = x^x$ .

Let's graph over the interval  $[0, 2]$  the difference quotient for  $h = 1, 0.1, 0.01, 0.001$ .

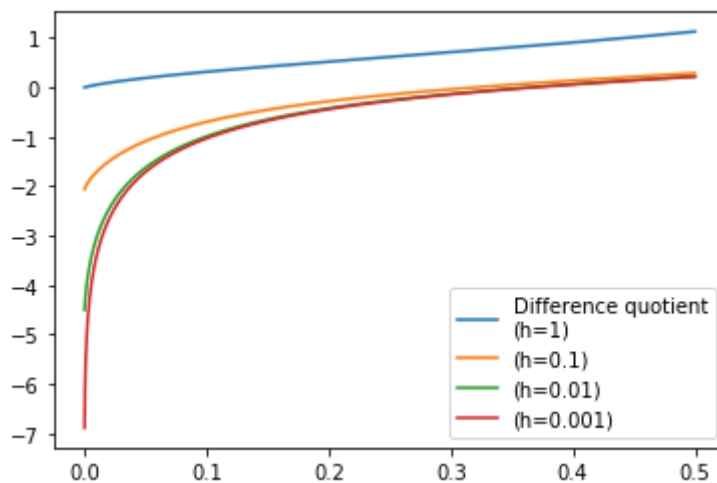


```
In [10]: def f(x): return x**x

x = linspace(0,0.5,10000)

for h in [1, 0.1, 0.01, 0.001]:
    m = (f(x+h) - f(x))/h
    plot(x,m)

legend([
    'Difference quotient\n(h=1)',
    '(h=0.1)',
    '(h=0.01)',
    '(h=0.001)',
]);
```



The graph of the difference quotient approaches the graph of the derivative function  $f'$ . For  $h$  very close to zero, the difference quotient approximates the derivative.

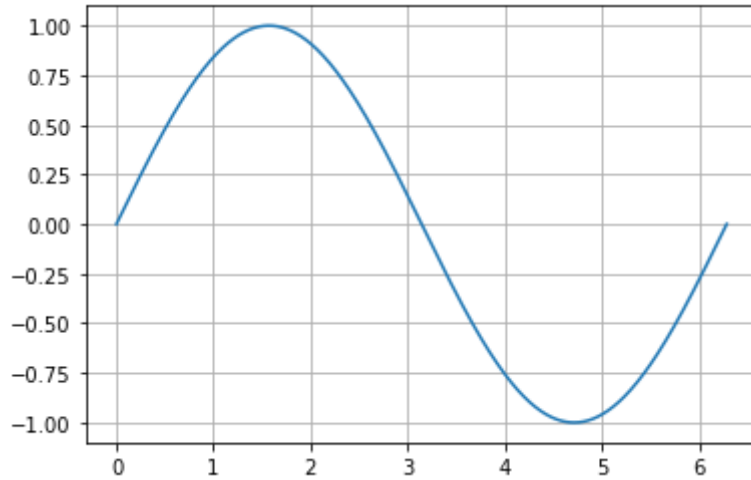
D9. **EXERCISE (1)–(4).** Consider the same function as in [D5](#),  $f(x) = \sin x$ .

(1) Plot  $f$  over  $[0, 2\pi]$ . Add a grid.

In [11]: # (1) TYPE YOUR CODE:

```
def f(x): return sin(x)

x = linspace(0,2*pi,10000)
plot(x,f(x))
grid();
```



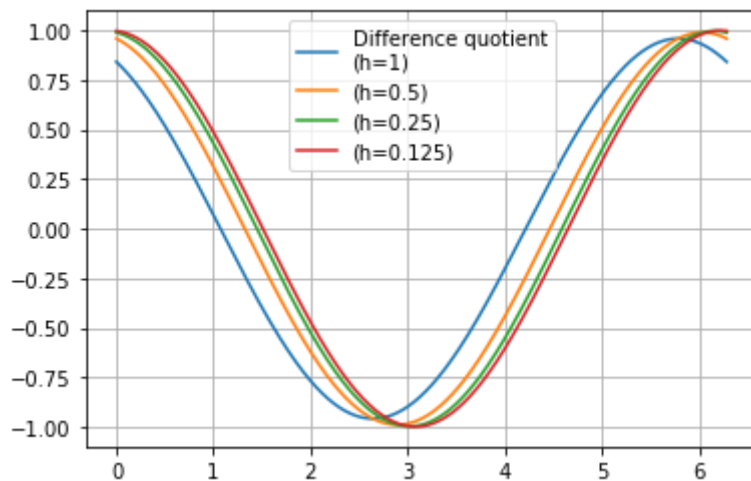
(2) Plot the difference quotient for  $h = 1, 0.5, 0.25, 0.125$ . Add a grid.

```
In [12]: # (2) TYPE YOUR CODE:

for h in [1, 0.5, 0.25, 0.125]:
    m = (f(x+h) - f(x))/h
    plot(x,m)

grid()

legend([
    'Difference quotient\n(h=1)',
    '(h=0.5)',
    '(h=0.25)',
    '(h=0.125)',
]);
```



(3) What graph do the difference quotients in (2) appear to be approaching?

```
In [13]: # (3) RECORD YOUR ANSWER: the graph of cos(x)
```

(4) What do the zeros of the graph in (2) tell you about the graph in (1)?

```
In [14]: # (4) RECORD YOUR ANSWER:
#       At the zeros in (2),
#       the slope in (1) is zero
```

#### D10. EXERCISE (1)–(5).

- (1) Let  $f(x) = e^x$ . Graph  $f$  over  $[-2, 2]$  and also graph the difference quotient for  $h = 0.01$ . The difference quotient for small  $h$  is approximately the derivative.
- (2) TRUE/FALSE. Since  $f$  is always increasing,  $f' > 0$
- (3) TRUE/FALSE. Tangent lines for  $f$  are never flat, so  $f$  has no critical points.
- (4) TRUE/FALSE. Since  $f$  increases faster and faster,  $f'$  should be increasing.
- (5) Take a guess as to what the derivative of  $f(x) = e^x$  is.

```
In [15]: # (1) TYPE YOUR CODE:

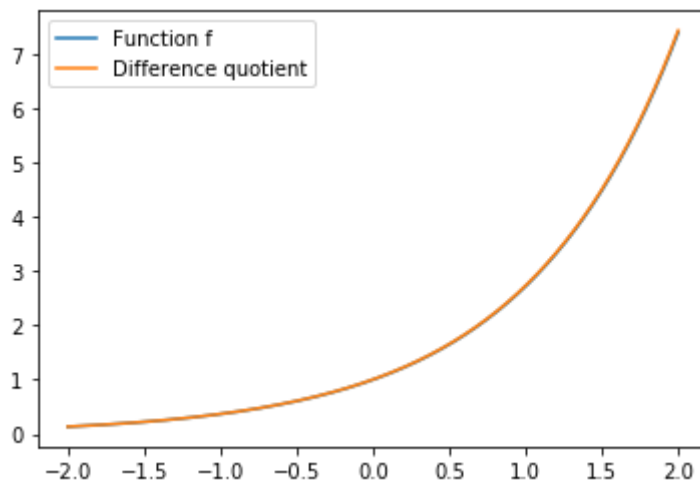
def f(x): return exp(x)

x = linspace(-2,2,10000)
plot(x,f(x))

h = 0.01
m = (f(x+h) - f(x))/h
plot(x,m)

legend(['Function f','Difference quotient']);

# (2) RECORD YOUR ANSWER: true
# (3) RECORD YOUR ANSWER: true
# (4) RECORD YOUR ANSWER: true
# (5) RECORD YOUR ANSWER: e^x
```



## Differentiability

D11. EXAMPLE. In lecture we learned that corners and cusps mean "continuous but not differentiable." Let's look at a function that has a cusp at 0.

Plot  $f(x) = x^{2/3}$  over  $[-2, 2]$ , and also plot the difference quotient for  $h = 1, 0.5, 0.25, 0.125$ .

(Numpy does not like taking the cube root of negative numbers, so we will graph  $(x^2)^{1/3}$  which is equivalent to  $x^{2/3}$ .)

```

In [16]: def f(x): return (x**2)**(1/3)

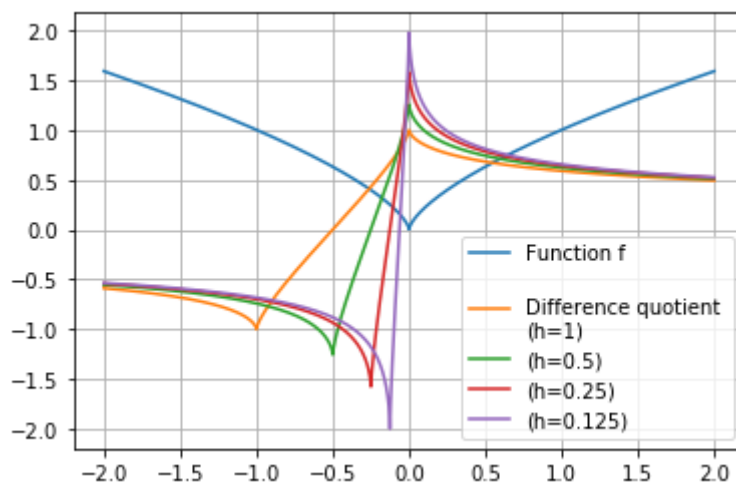
x = linspace(-2,2,10000)
plot(x,f(x))

for h in [1, 0.5, 0.25, 0.125]:
    m = (f(x+h) - f(x))/h
    plot(x,m)

grid()

legend([
    'Function f',
    '\nDifference quotient \n(h=1)',
    '(h=0.5)',
    '(h=0.25)',
    '(h=0.125)',
]);

```



The graph of the difference quotient approaches the graph of the derivative of  $x^{2/3}$ , which is  $(2/3)x^{-1/3}$ . Since  $f$  has a cusp at 0, the derivative has an asymptote at 0.

#### D12. EXERCISE (1)–(2).

(1) Plot  $f(x) = |x|$  over  $[-2, 2]$ , and also plot the difference quotient for  $h = 1, 0.5, 0.25, 0.125$ .

(2) Fill in the blank: Since  $f$  has a corner at 0, thus  $f'$  has a \_ at 0.

```

In [17]: # (1) TYPE YOUR CODE:

def f(x): return abs(x)

x = linspace(-2,2,10000)
plot(x,f(x))

for h in [1, 0.5, 0.25, 0.125]:
    m = (f(x+h) - f(x))/h
    plot(x,m)

grid()

legend([
    'Function f',
    '\nDifference quotient \n(h=1)',
    '(h=0.5)',
    '(h=0.25)',
    '(h=0.125)',
]);

# (2) RECORD YOUR ANSWER:
#     Since f has a corner at 0,
#     thus f' has a jump at 0.

```

