AN ACCOMPLISH PROJECT MODULE

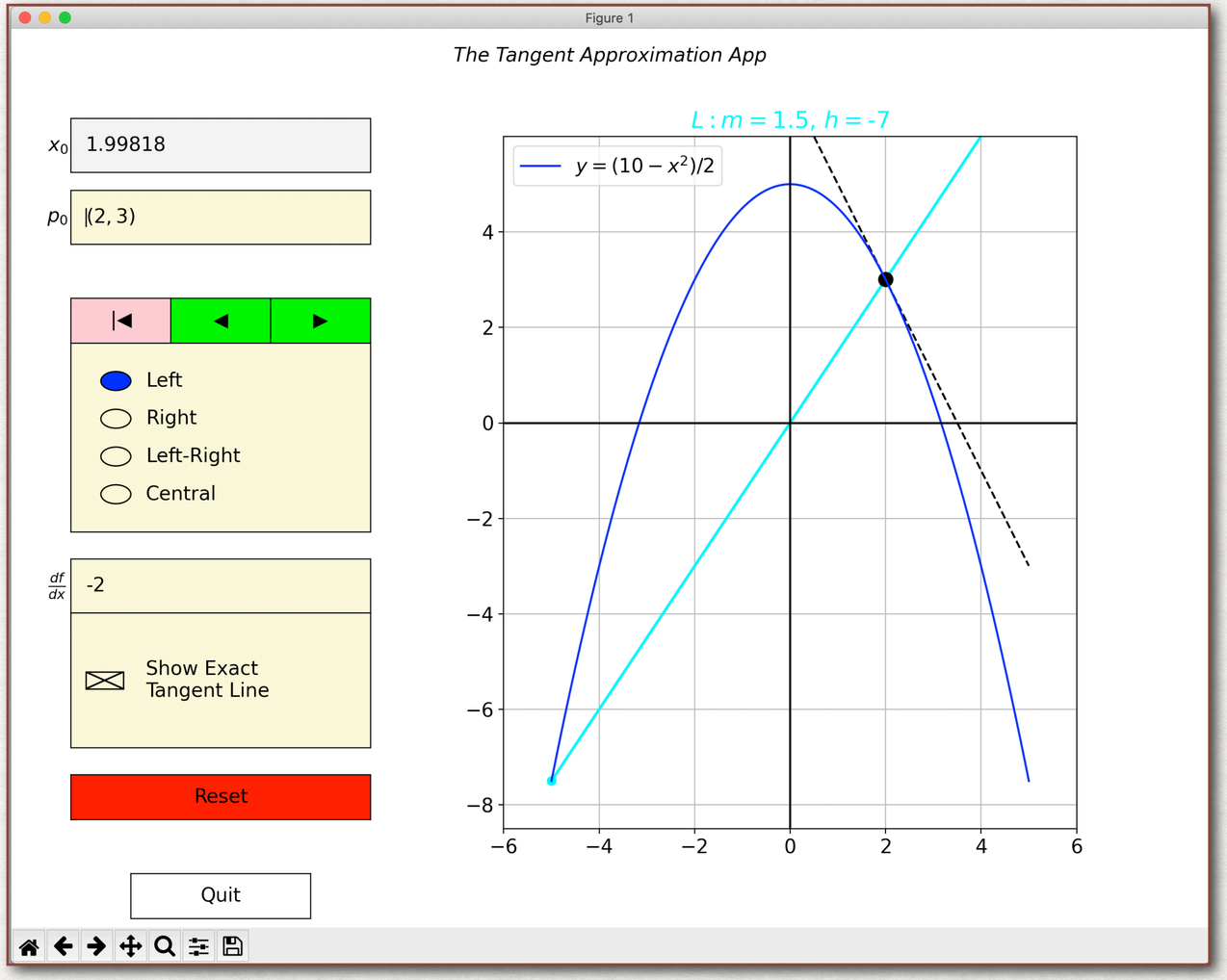
# THE TANGENT APPROXIMATION PYTHON APP: VISUALIZING THE NUMERICAL APPROXIMATION OF THE DERIVATIVE

Prof. Adam Hausknecht

# THE APPROXIMATION OF THE DERIVATIVE 2/25

## Goals:

- Learn how to numerically approximate the derivative of a function at a point
- Complete a Python app that uses secants to visually approximate the tangent line at a point on the graph of a function
- Learn about and practice using the Python mathematical packages ***matplotlib***, ***numpy***, and ***scipy***
- Learn how to create a Python package or library which can be imported by other Python programs




UMass | Dartmouth

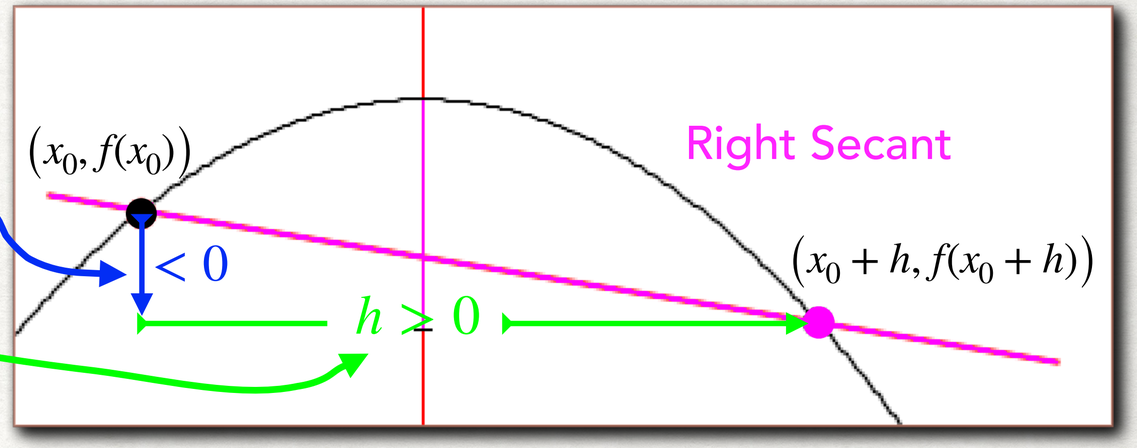
# THE APPROXIMATION OF THE DERIVATIVE 3/25

## Mathematical Background

- Recall that if $I = (a, b) \subset \mathbb{R}$ is an open interval and $f$ is a real-valued function defined on $I$, then the derivative of $f$ at $x = x_0 \in I$ is defined by

$$\frac{df}{dx}(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

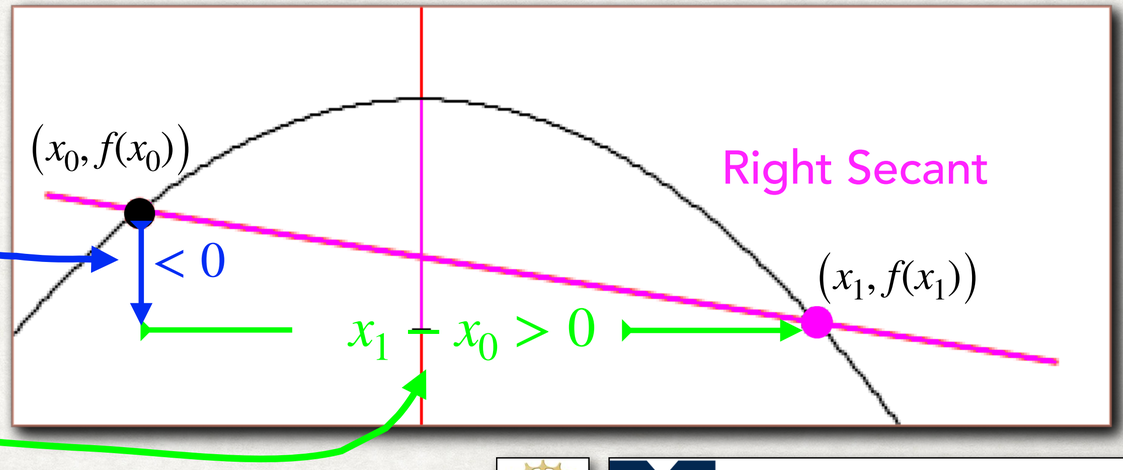
*provided* the limit exists.




- Notice that if $x_1 = x_0 + h$, then $h = x_1 - x_0$ and $f(x_0 + h) = f(x_1)$. Thus, this limit can be expressed as

$$\frac{df}{dx}(x_0) = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

*provided* the limit exists.



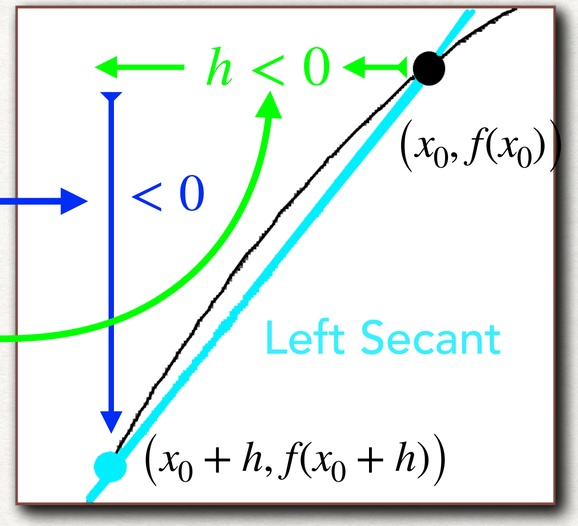
# THE APPROXIMATION OF THE DERIVATIVE 4/25

**Remarks**

- You shouldn't assume $h$ in limit *is always* ***positive.***

$$\frac{df}{dx}(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

$$\frac{f(x_0 + h) - f(x_0)}{h} = \text{slope of the secant} > 0$$

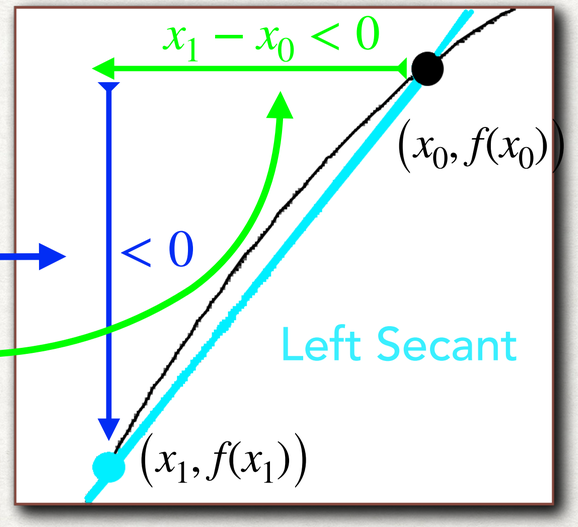



- Similarly, you shouldn't assume $x_1$ in the limit *is always to the* ***right*** *of* $x_0$

$$\frac{df}{dx}(x_0) = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

.

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = \text{slope of the secant} > 0$$



# THE APPROXIMATION OF THE DERIVATIVE 5/25

## Remarks

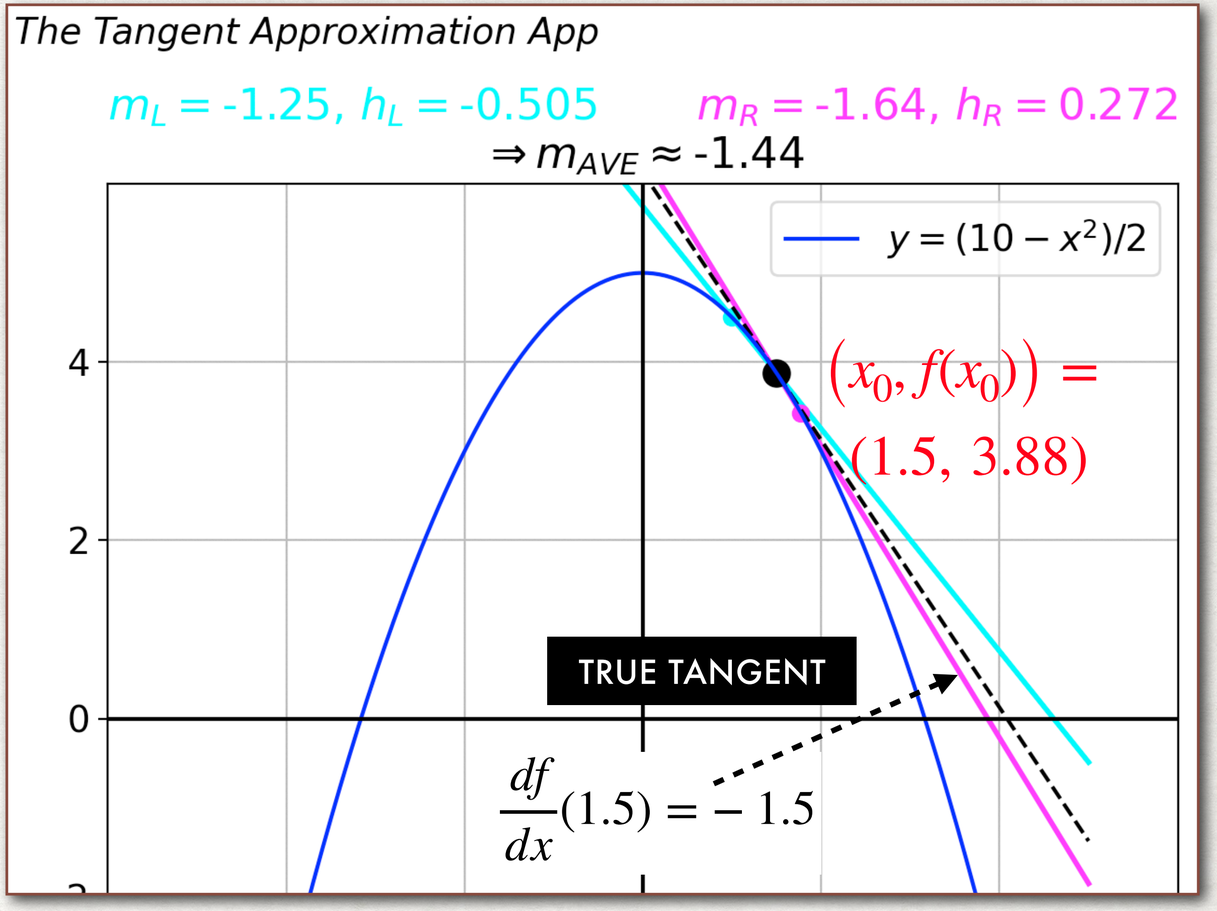
- If $h$ is **small** negative or positive, then

$$\frac{df}{dx}(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h} = \text{slope of secant}$$

Thus, we can use the ***left*** and ***right*** secants to approximate the tangent line at $(x_0, f(x_0))$ and their slopes to approximate $\frac{df}{dx}(x_0)$.

- If we ***average*** the left and right slopes, we get a better approximation of $\frac{df}{dx}(1.5)$:

$$m_{AVE} = \frac{m_L + m_R}{2} = \frac{-1.25 + (-1.64)}{2} \approx -1.44$$




UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 6/25

**Remarks**

- If we assume $h > 0$ is small, and $h_L = -h$, $h_R = h$

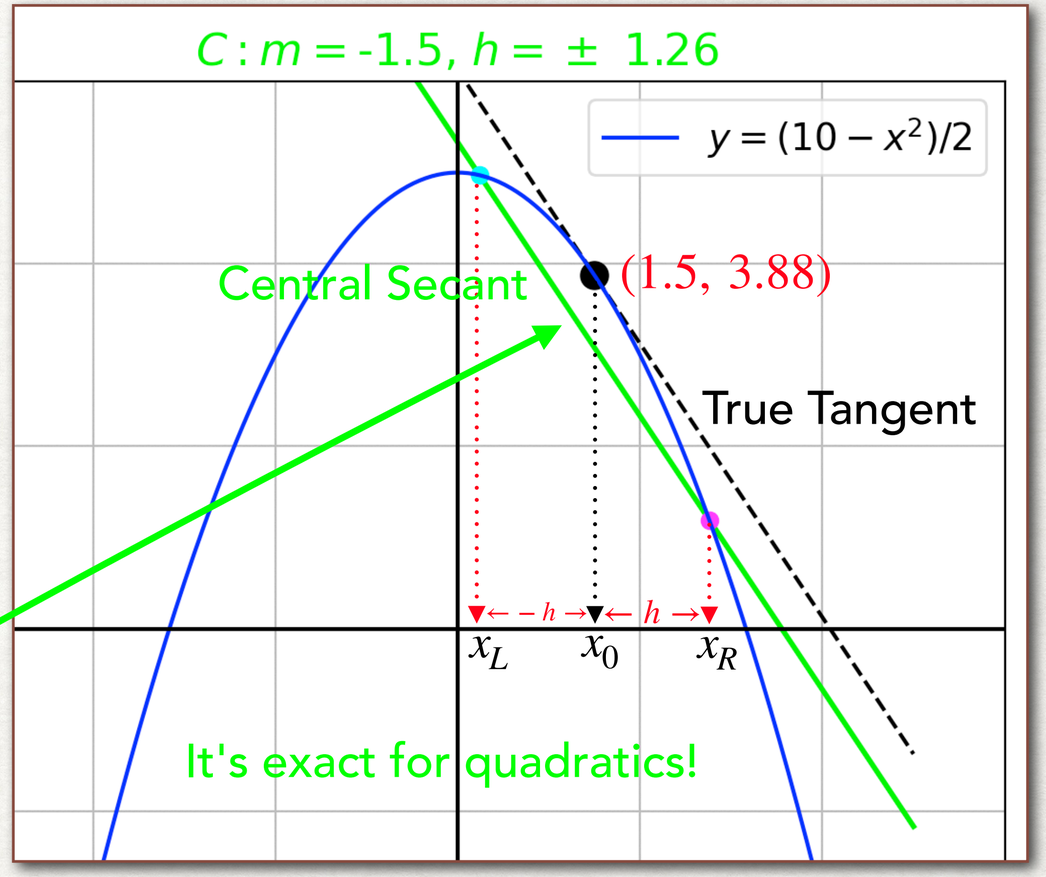
$$\frac{df}{dx}(x_0) \approx \frac{1}{2}\left(\frac{f(x_0+h_L)-f(x_0)}{h_L} + \frac{f(x_0+h_R)-f(x_0)}{h_R}\right)$$

$$\approx \frac{1}{2}\left(\frac{f(x_0-h)-f(x_0)}{-h} + \frac{f(x_0+h)-f(x_0)}{h}\right)$$

$$\approx \frac{1}{2h}\left(\cancel{f(x_0)} - f(x_0-h) + f(x_0+h) - \cancel{f(x_0)}\right)$$

$$\approx \frac{f(x_0+h)-f(x_0-h)}{2h}$$

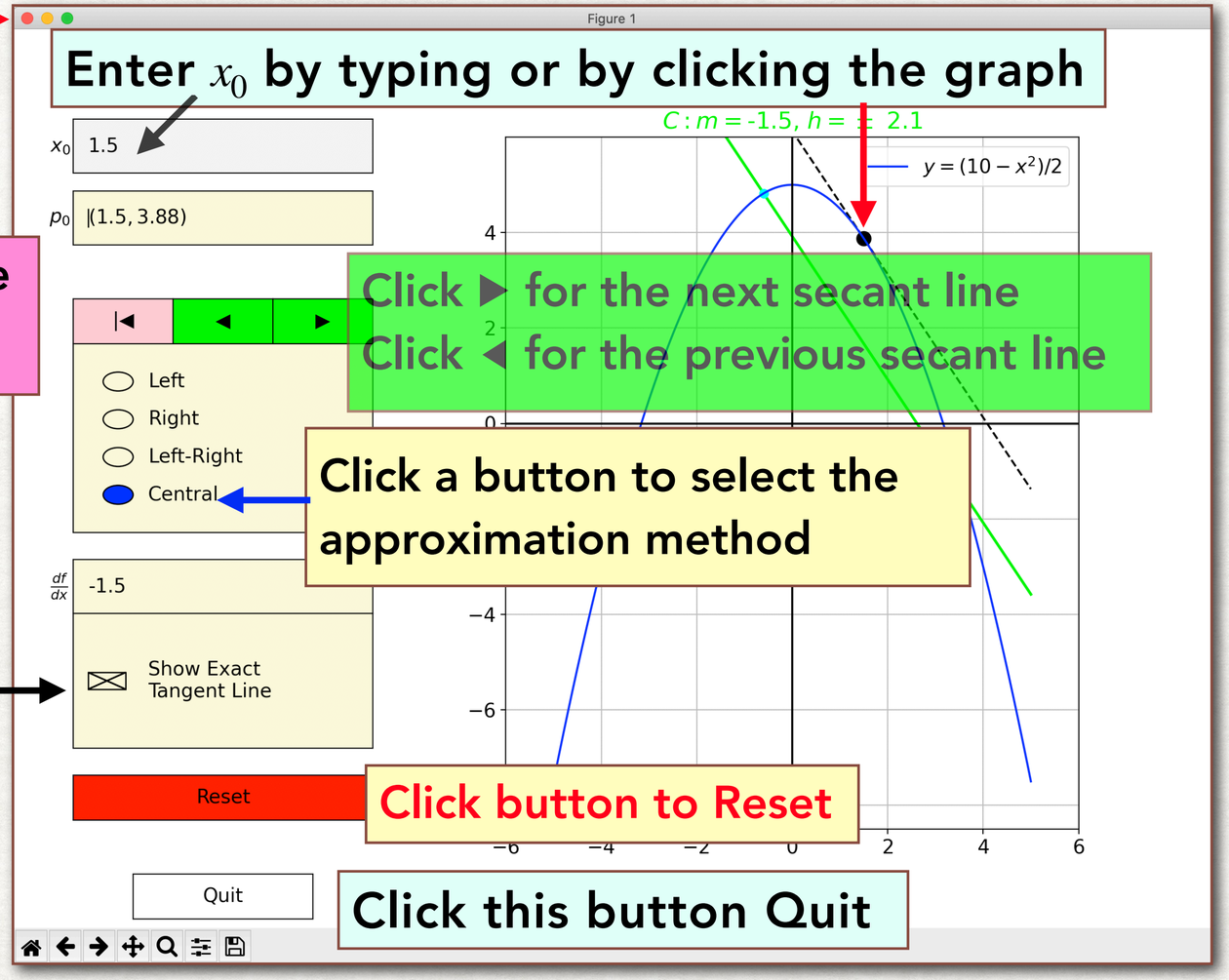
- $\frac{df}{dx}(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$ is called the **Central Difference** or **Three-Endpoint** derivative approximation for $\frac{df}{dx}(x_0)$.



# THE APPROXIMATION OF THE DERIVATIVE 7/25

## Tangent Approximation App GUI

Click to Quit




Click |◀ to restart at the first secant line

Click ▶ for the next secant line

Click ◀ for the previous secant line

Click a button to select the approximation method

Check to display an exact or accurate tangent line

Click button to Reset

Click this button Quit

UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 8/25

## Completing the Tangent Approximation App

- To complete this project, you will need learn

1. How to use Python functions including **lambda** functions.
2. How to use Python **lists** and **tuples.**
3. How to create numpy **arrays** from lists and from numpy's **linspace** function.
4. How to evaluate mathematical expressions using numpy's **arrays** and numpy's. mathematical function library the includes special versions of **abs**, **trig** and **other** functions.
5. How to use matplotlib's **axis** object to plot function including how to label the function.
6. How to use Python's **try-except** statement used to write test functions for the app.

# THE APPROXIMATION OF THE DERIVATIVE 9/25

## Completing the Tangent Approximation App: Documentation & References

- The code was developed using the **Anaconda Python 3 Distribution**
  https://www.anaconda.com/products/individual
  the includes the packages **matplotlib**, **numpy**, **scipy**
  https://scipy.org
- There are numerous online sites **matplotlib**, **numpy**, and **scipy** documentation, examples, tutorials and help
- The Anaconda Python Distribution also includes the **Spyder IDE**
  https://www.spyder-ide.org
- There are online **Tutorials Videos** for **Spyder**

# THE APPROXIMATION OF THE DERIVATIVE 10/25

## Completing the Tangent Approximation App

- You will need to complete the following functions in **tangentApproximationStudentLib.py**

1. secantline(x:float, m:float, x0:float, y0:float) ->:float
2. nextH(h:float)->float:
3. generateLeftOrRightSecantSlopeAndPts(func, x0:float, y0:float,
   h:float, xMin:float, xMax:float)->Tuple:
4. generateCentralSecantSlopeAndPts(func, x0: float,
   h:float, xMin:float, xMax:float)->tuple:
5. plotTheFunction(axes, func, xMin:float, xMax:float, labelStr:str)-> tuple:

# THE APPROXIMATION OF THE DERIVATIVE 11/25

## Completing the Tangent Approximation App: The Library Part 1

```
1   import numpy as np
2
3   def secantline(x:float, m:float, x0:float, y0:float)->float:
4       """For x, the correspoding y-value of the secant line passing through
5          (x0, y0) of slope m is calculated and returned."""
6       
7       #Add code here!
8       return # The y-value corresponding to x
9       
10  def nextH(h:float)->float:
11      """The next value of h = Δx is 60% of its previous value."""
12      
13      #Add your code here!
14      return # the next value of h
15
16  def generateLeftOrRightSecantSlopeAndPts(func, x0:float, y0:float,
17                                           h:float, xMin:float, xMax:float)->tuple:
18      """Using the test function func, x0, y0, and h = Δx, the values of x1, y1
19         and the slope m of the secant line through (x0,y0) and (x1,y1) are
20         calculated. Using func, m, x0 and y0, the xValues array and yValues
21         array needed to plot the the secant are also generated. The results
22         are returned as the tuple (m, x1, y1, xValues, yValues)."""
23      
24      #Add your code here!
25      return #(m, x1, y1, xValues, yValues)
```

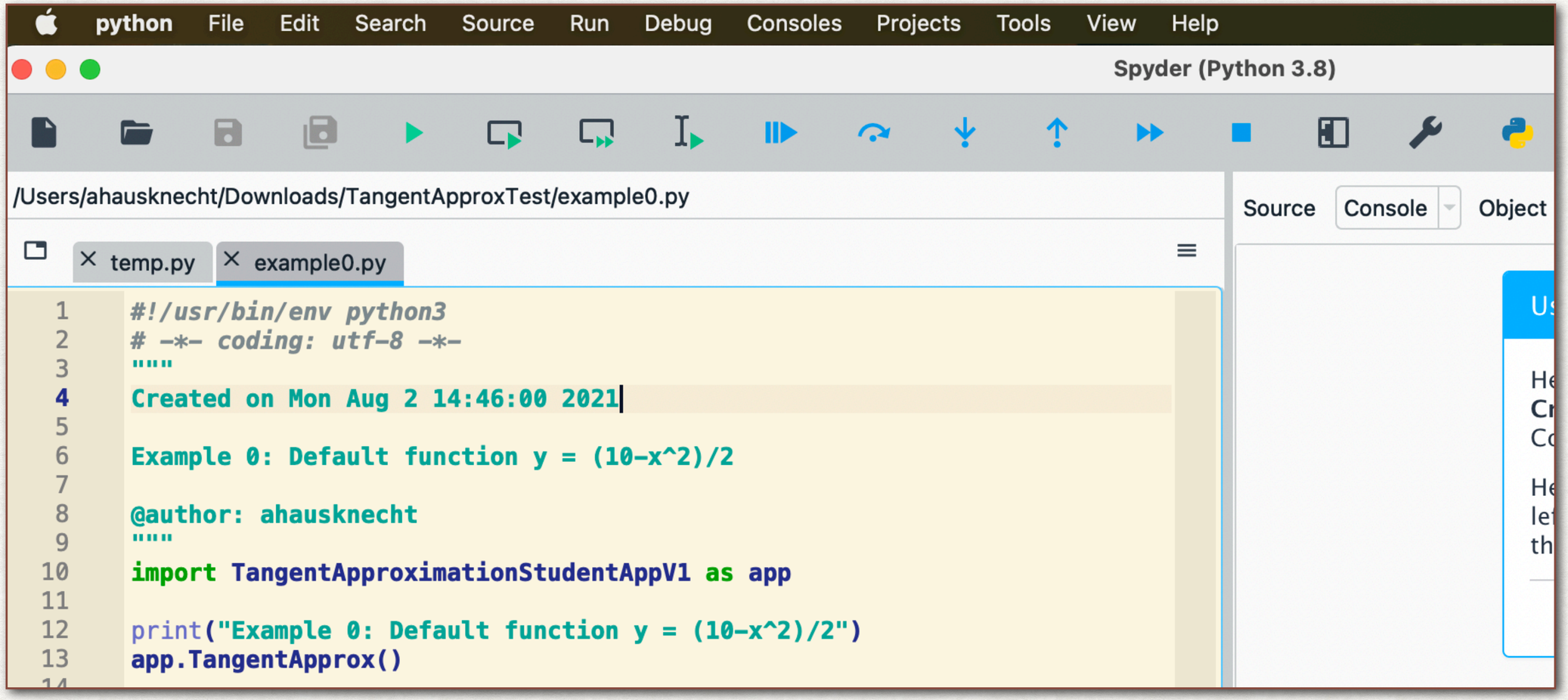
# THE APPROXIMATION OF THE DERIVATIVE 12/25

## Completing the Tangent Approximation App: The Library Part 2

```
28  def generateCentralSecantSlopeAndPts(func, x0: float,
29                                       h:float, xMin:float, xMax:float)->tuple:
30      """Using the test function func, x0, and h = Δx, the left (xL,yL)
31         and right (xR,yR) central-secant points are calculated. The slope
32         m of the central-secant throught these points is also found. Using
33         m and the points, the xValues array and yValues array needed to plot
34         the central-secant are also generated. The results are returned
35         as the tuple (m, xL, yL, xR, yR, xValues, yValues)."""
36
37      #Add your code here!
38      return #(m, xL, yL, xR, yR, xValues, yValues)
39
40
41  def plotTheFunction(axes, func, xMin:float, xMax:float, labelStr:str)-> tuple:
42      """Use the axes to plot the function func over the domain [xMin, xMax].
43         To do this, 300 x-values and y-values must be generated for the
44         plot. The labelStr is used to label the plot."""
45
46      #Add your code here!
47      return # the minimum and the maximum of the calculated y-values
48
```


# THE APPROXIMATION OF THE DERIVATIVE 13/25

## Running the Completed App from Spyder:




- Spyder 5.0.5 with example0.py

# THE APPROXIMATION OF THE DERIVATIVE 14/25

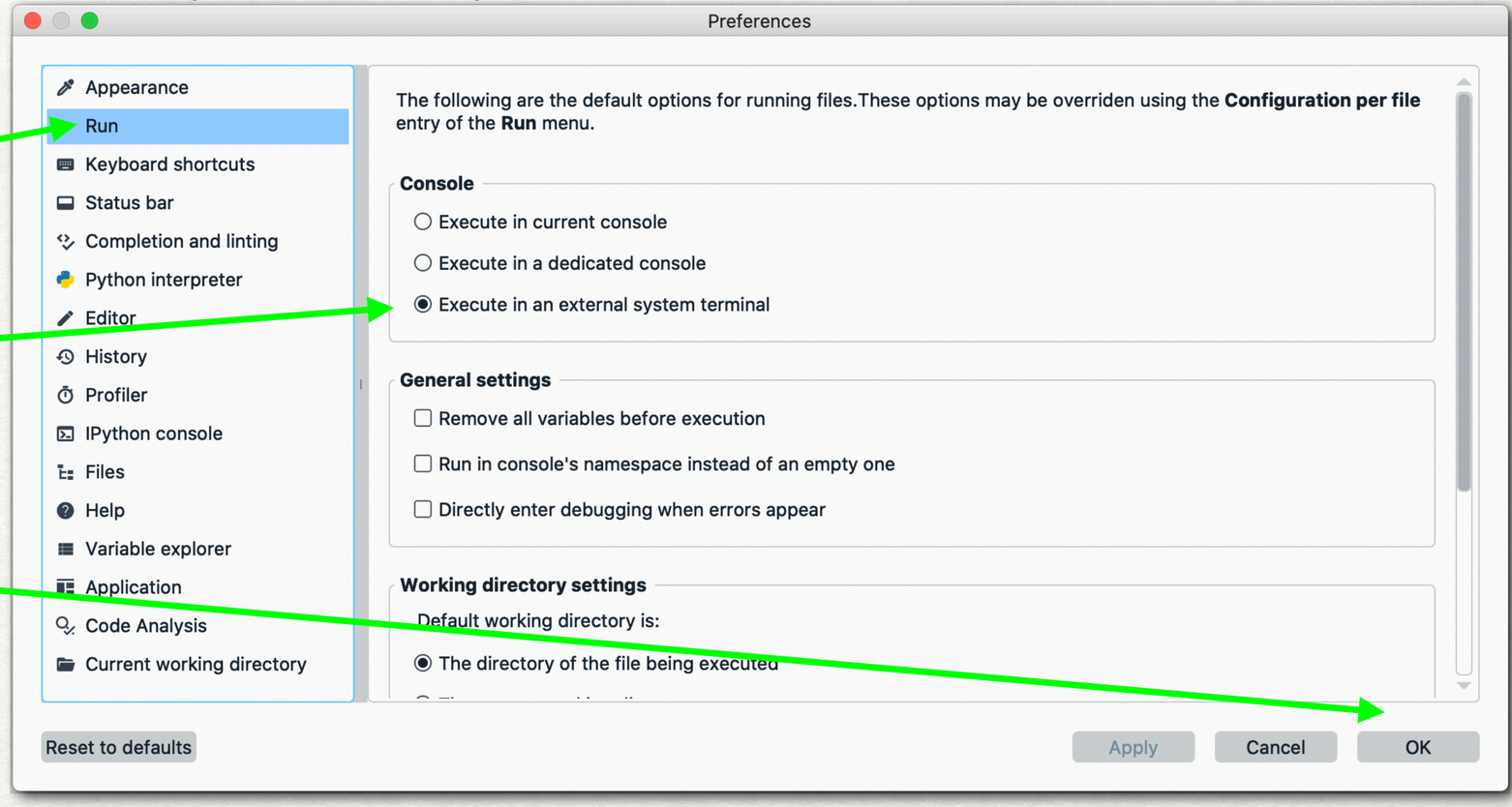
## Running the Completed App from Spyder:

- To avoid a Spyder bug, we need to change **Spyder's** default **Run** setting. To do this click the Wrench Preferences icon  at the top which will open the **Preferences** window shown below:

- Click the **Run** button to display the run settings

- Select *"Execute in an external system terminal"* option.

- Click the *"Apply"* and *"OK"* buttons to change to the new setting and close the preferences window.

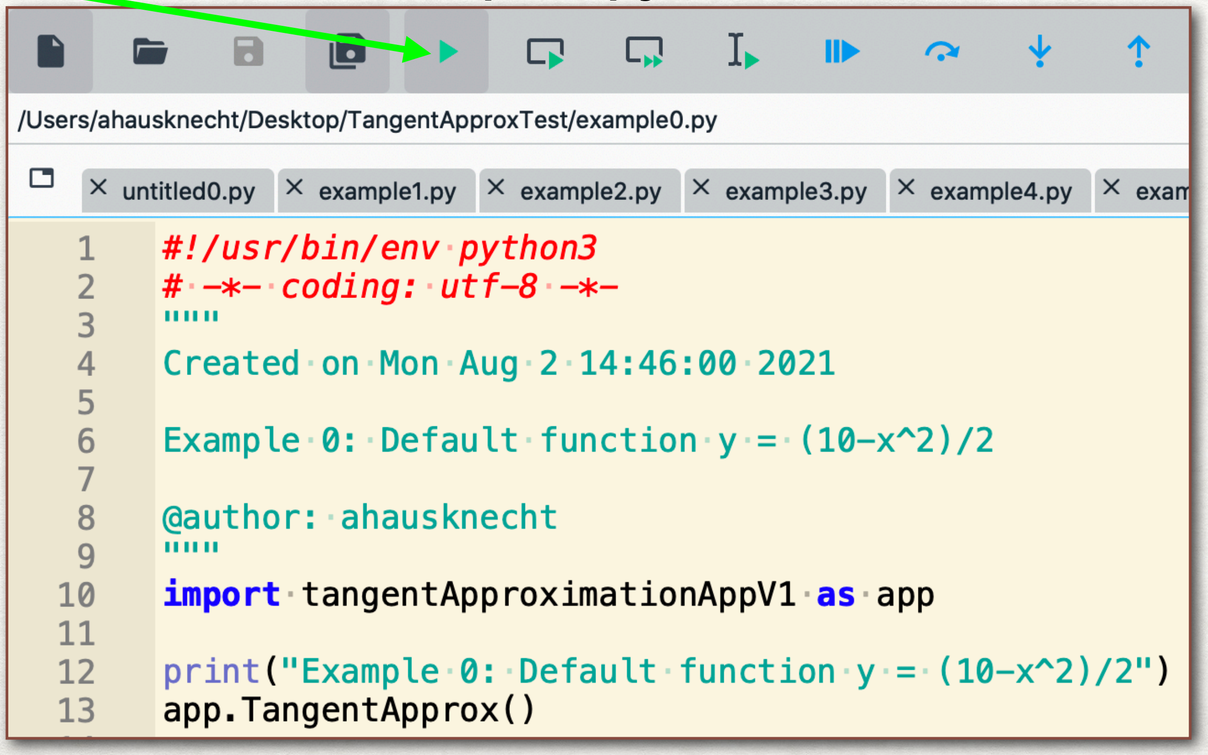


# THE APPROXIMATION OF THE DERIVATIVE 15/25

## Running the Completed App from Spyder

- Both the **tangentApproximationAppV1.py** and **tangentApproximationStudentLib.py** must be in the same directory.

1. Create the following short program in the file **example0.py** the same directory as **tangentApproximationAppV1.py** and **tangentApproximationStudentLib.py.**
2. Click the **Run** ▶ icon to run **example0.py** in external terminal.

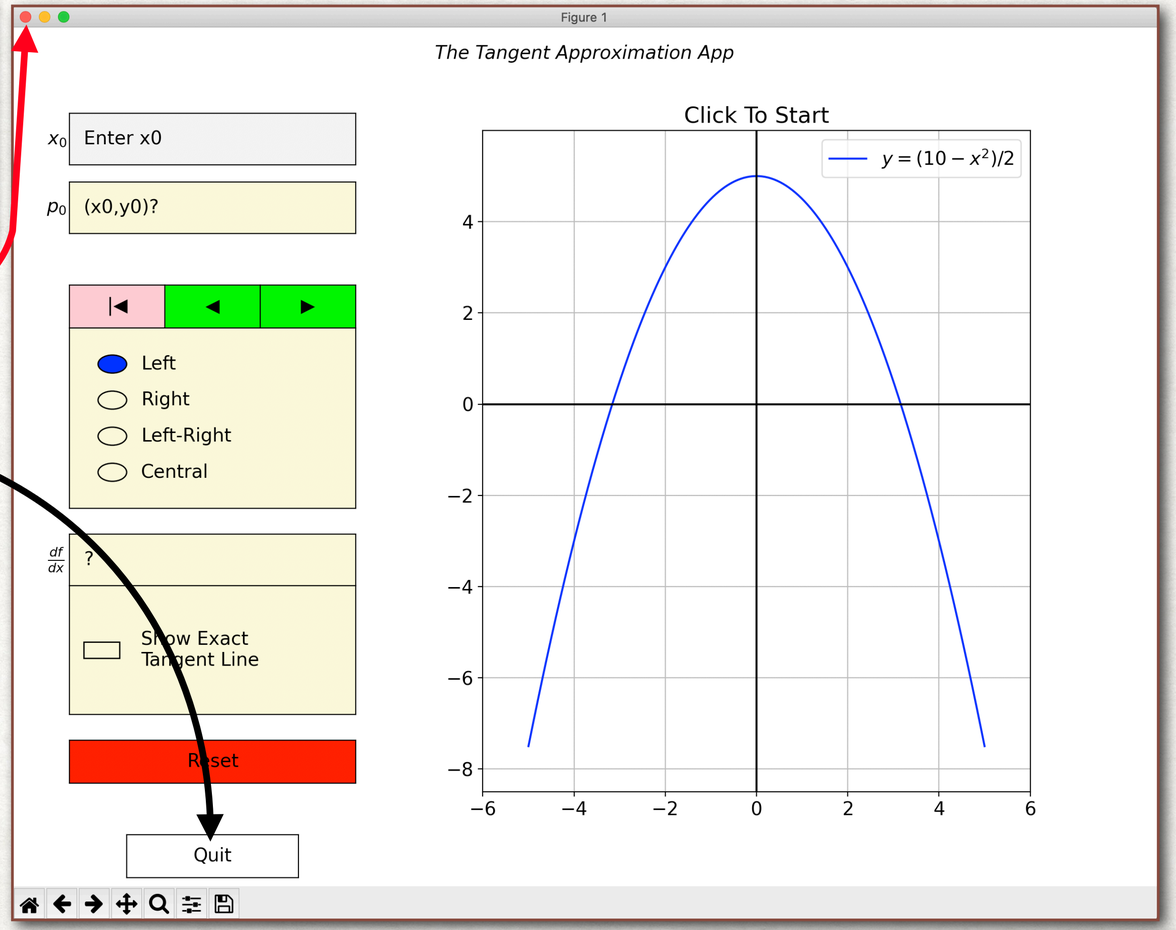


# THE APPROXIMATION OF THE DERIVATIVE 16/25

## Running the Completed App from Spyder

- **Spyder** then opens a **Terminal** window and runs a script the runs **example0.py**:

```
ahausknecht — run_spyder_vl02p4x5.sh — python ◂ run_spyder_vl02p4x5.sh — 80×24
Last login: Mon Aug  2 13:57:20 on ttys001
/var/folders/31/r_2f8ryd5ks7xb80nyrbj2cc0000gn/T/spyder-ahausknecht/run_spyder_v
l02p4x5.sh ; exit;
Enter GCC for gcc
zsh_profile
(base) ahausknecht@Adam314 ~ % /var/folders/31/r_2f8ryd5ks7xb80nyrbj2cc0000gn/T/
spyder-ahausknecht/run_spyder_vl02p4x5.sh ; exit;
Example 0: Default function y = (10-x^2)/2
```




- Click the small red button or the **Quit** button to exit the app.

- **The Terminal** window will show the following message:

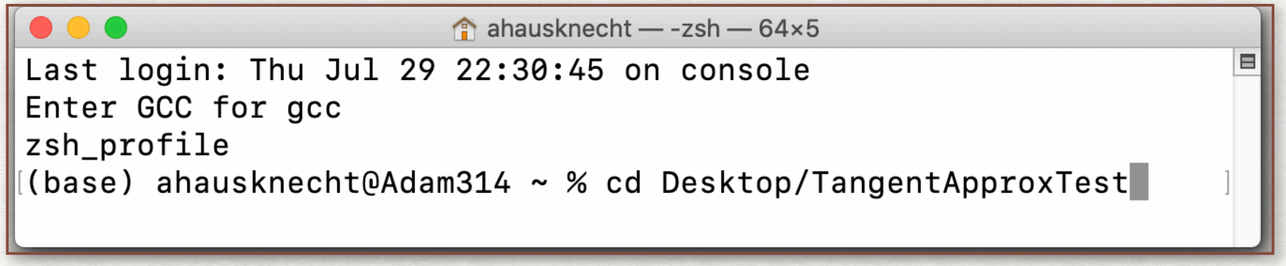
```
Tangent Approximation App ending! Bye-Bye...

[Process completed]
```

UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 17/25

## Running the Completed App Directly via the Terminal

1. Open a **Terminal** window.
2. Use the **cd** command to change directories to the one containing both **tangentApproximationAppV1.py** and **example0.py**.




3. At the prompt, enter **% python example0.py**

```
(base) ahausknecht@Adam314 TangentApproxTest % ls
__pycache__/                    example5.py
example0.py                     example6.py
example1.py                     tangentApproxLib.py
example2.py                     tangentApproxLibStudent.py
example3.py                     tangentApproximationAppV1.py*
example4.py
(base) ahausknecht@Adam314 TangentApproxTest % python example0.py
Example 0: Default function y = (10-x^2)/2
```

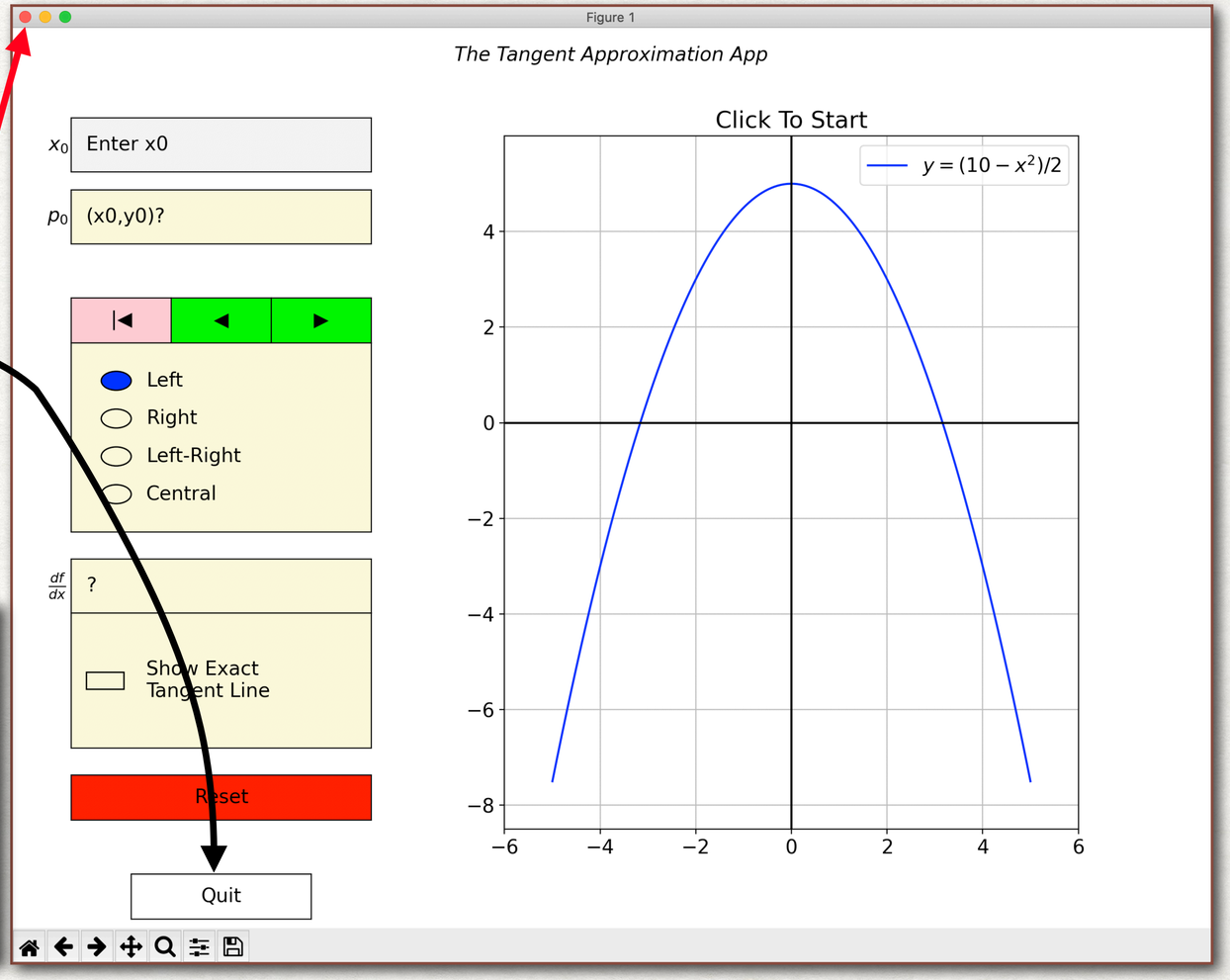
# THE APPROXIMATION OF THE DERIVATIVE 18/25

## Running the Completed App from the Terminal

- Note that the default built-in function $f(x) = (10 - x^2)/2$ is used.
- Again, click the small red button or the **Quit** button to **exit** the app.
- Note from the **Terminal**, you can also run the Tangent app by entering

  **% python tangentApproximationAppV1.py**

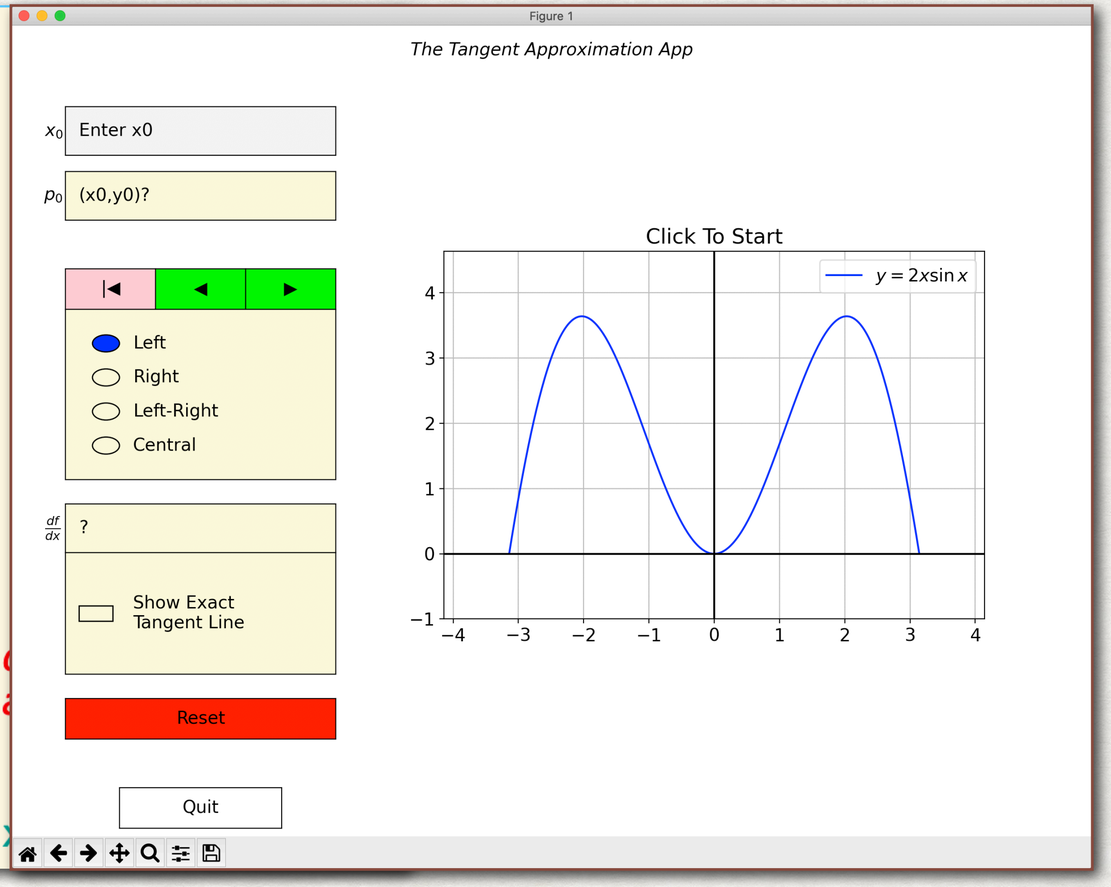
```
(base) ahausknecht@Adam314 TangentApproxTest % ls
__pycache__/
example0.py
example1.py
example2.py
example3.py
example4.py
example5.py
example6.py
tangentApproxLib.py
tangentApproxLibStudent.py
(base) ahausknecht@Adam314 TangentApproxTest % python tangentApproximationAppV1.py
```



# THE APPROXIMATION OF THE DERIVATIVE 19/25

## Testing the App: Example 1

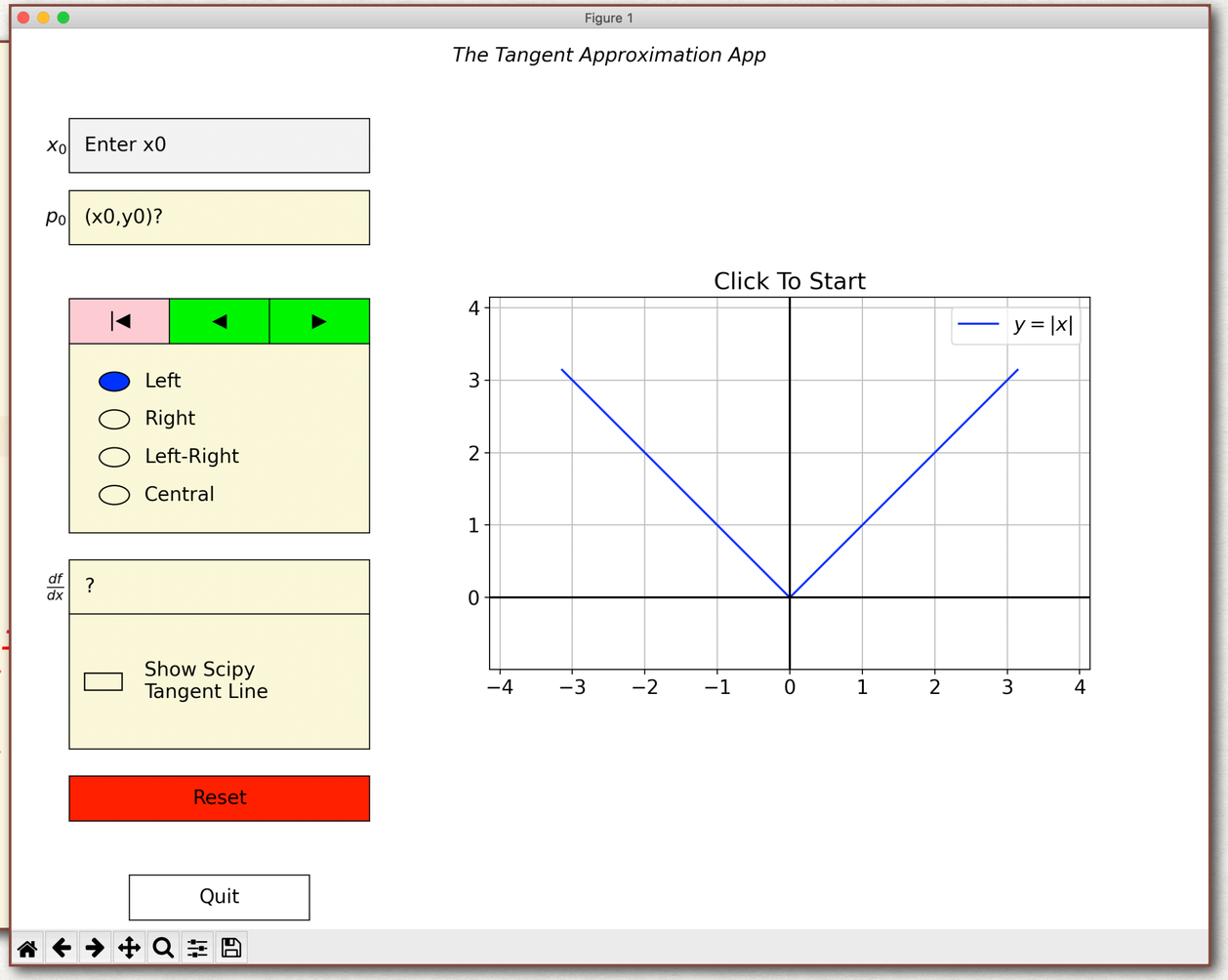
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

Example 1: y = 2x sin x

@author: ahausknecht
"""
import tangentApproximationAppV1 as app
import numpy as np

print("Example 1: y = 2x sin x")

# Use either a lambda function or use a normal functi
# if-statements which will cause a problem for numpy a
f = lambda x: 2*x*np.sin(x)
dfdx = lambda x: 2*(np.sin(x)+x*np.cos(x))
app.TangentApprox(f, dfdx, -np.pi, np.pi, '$y=2x\sin x
```



# THE APPROXIMATION OF THE DERIVATIVE 20/25

## Testing the App: Example 2

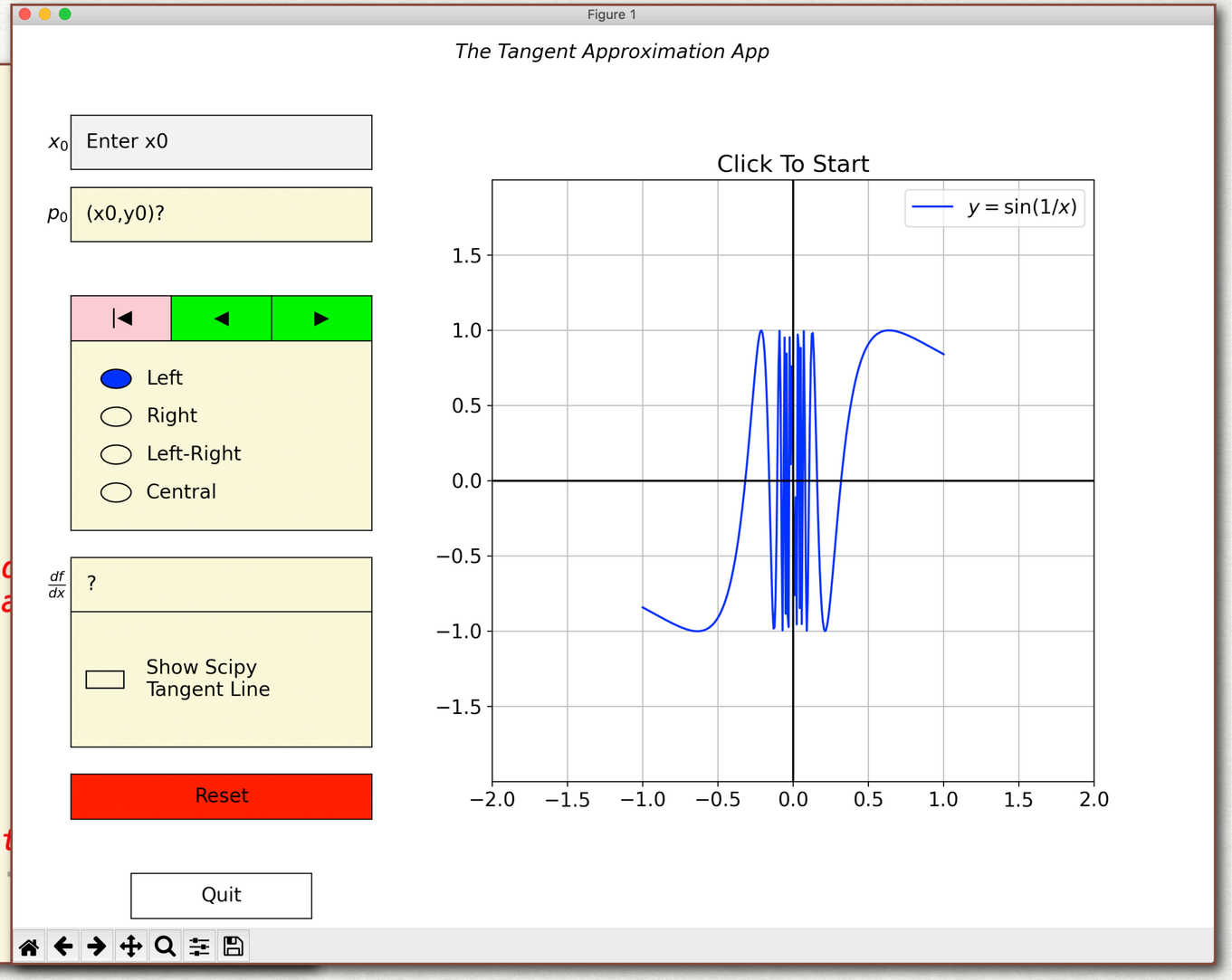
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

"Example 2: y = |x|"

@author: ahausknecht
"""
import tangentApproximationAppV1 as app
import numpy as np

print("Example 2: y = |x|")

# Use either a lambda function or use a normal functi
# if-statements which will cause a problem for numpy
f = lambda x: np.abs(x)
# Passing None for dfdx forces the app.TangentApprox
# numerical derivative method to calculate dy/dx.
dfdx = None
app.TangentApprox(f, dfdx, -np.pi, np.pi, '$y=|x|$')
```



# THE APPROXIMATION OF THE DERIVATIVE 21/25

## Testing the App: Example 3

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

Example 3: y = sin(1/x)

@author: ahausknecht
"""
import tangentApproximationAppV1 as app
import numpy as np

print("Example 3: y = sin(1/x)")

# Use either a lambda function or use a normal functic
# if-statements which will cause a problem for numpy a
def f(x):
    try:
        return np.sin(1/x)
    except ArithmeticError:
        return np.nan

# Passing None for dfdx forces the app.TangentApprox t
# numerical derivative method to calculate dy/dx.
dfdx = None
app.TangentApprox(f, dfdx, -1, 1, '$y=\sin(1/x)$')
```

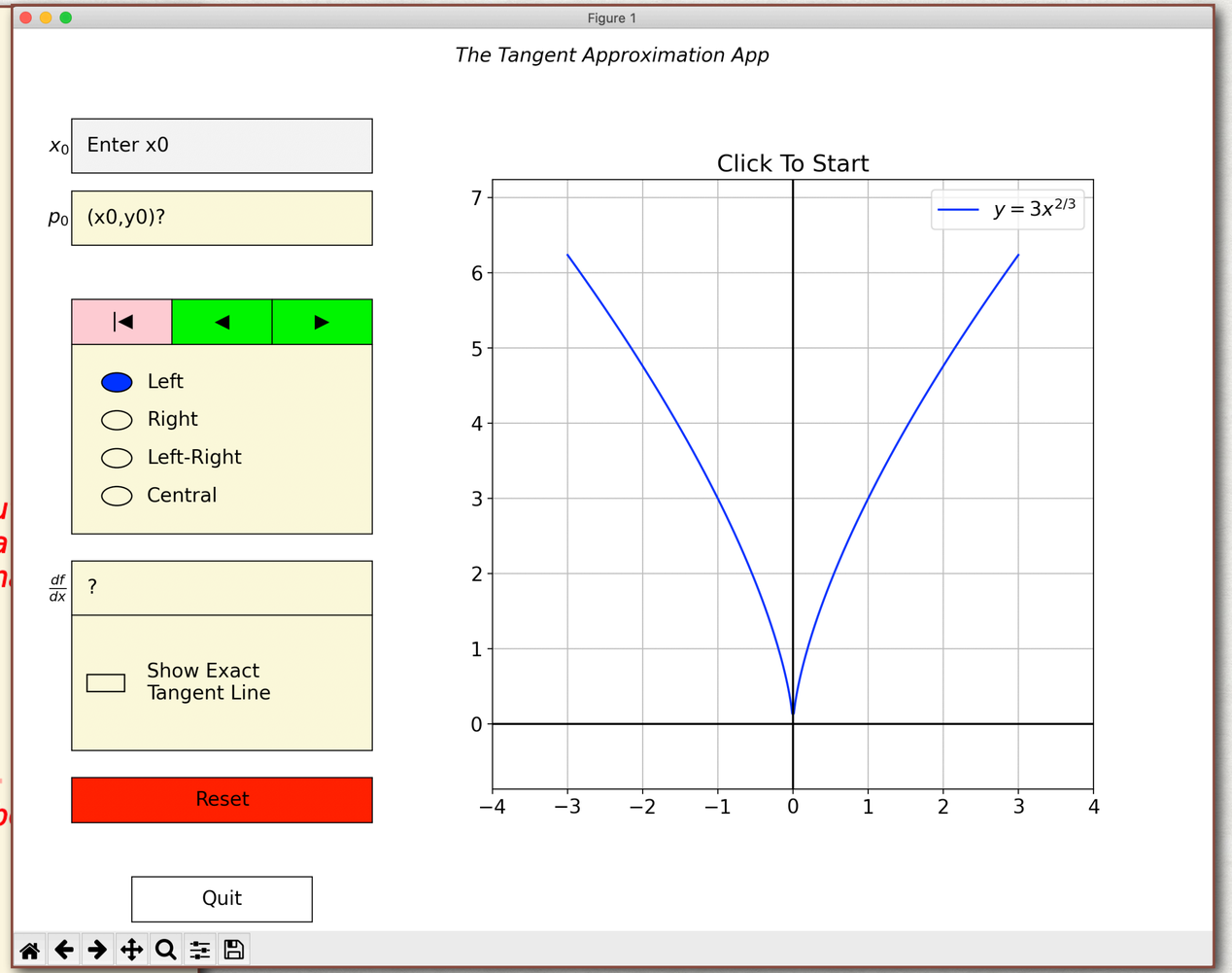



UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 22/25

## Testing the App: Example 4

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

"Example 4: y = 3x^(2/3)"

@author: ahausknecht
"""
import tangentApproximationAppV1 as app
import numpy as np

print("Example 4: y = 3x^(2/3)")

# Use either a lambda function or use a normal function withou
# if-statements which will cause a problem for numpy array eva
# Rewritten so that x is squared before rasing to the fraction
# exponent 1/3. This way, it will handle negative numbers.
f = lambda x:3*(x**2)**(1/3)

def dfdx(x):
    try:
        # Rewritten so that x is squared before rasing to the
        # exponent -2/3. This way it will handle negative numb
        # dfdx = 2x^(-1/3)=2*sign(x)/(x^2)^(1/6).
        return 2*np.sign(x)/( (x**2)**(1/6) )
    except RuntimeError:
        return np.nan

app.TangentApprox(f, dfdx, -3, 3, '$y=3x^{2/3}$')
```

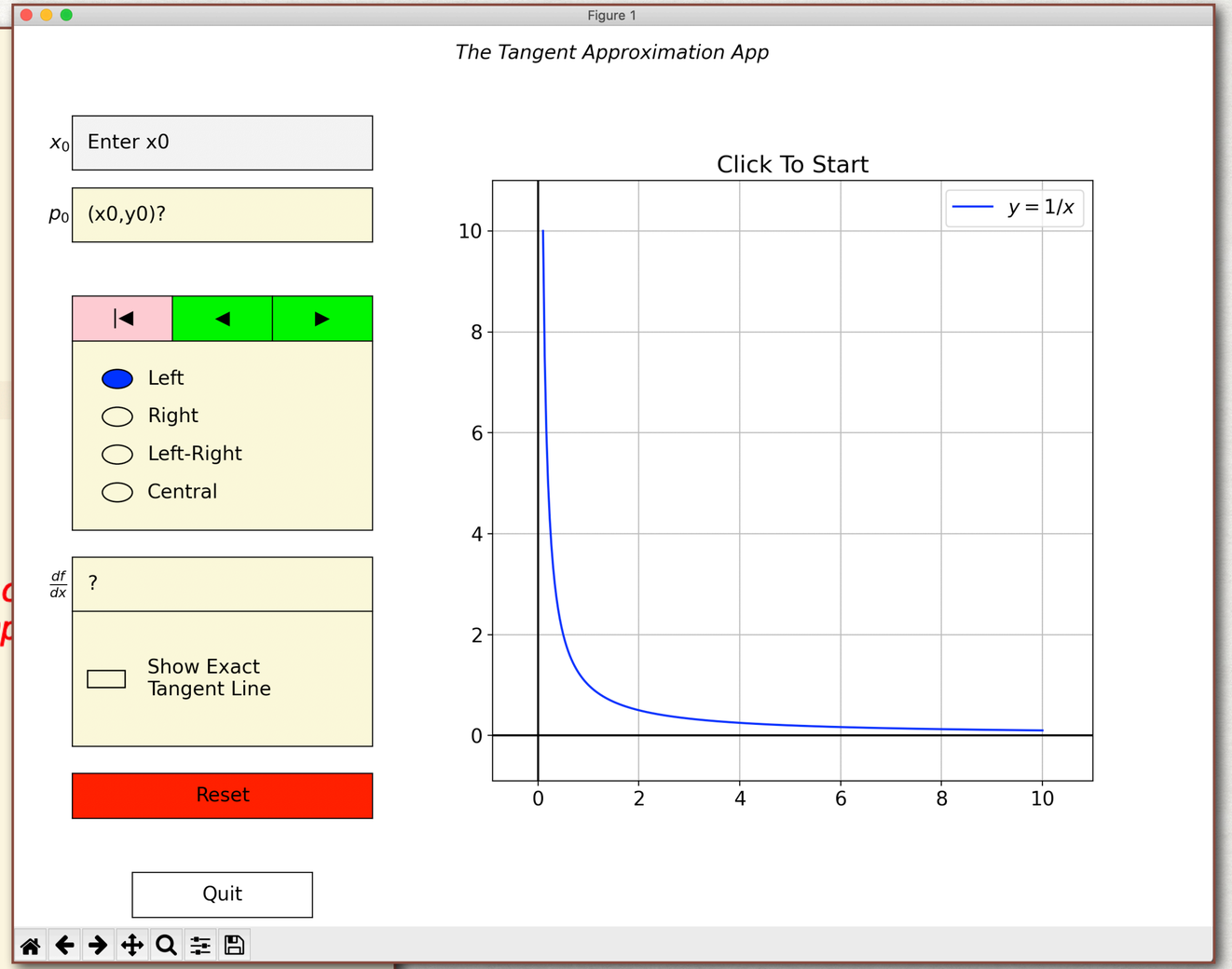



UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 23/25

## Testing the App: Example 5

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

Example 5: y = 1/x

@author: ahausknecht
"""
import tangentApproximationStudentAppV1 as app
import numpy as np

print("Example 5: y = 1/x")

# Use either a lambda function or use a normal func
# if-statements which will cause a problem for nump
f = lambda x:1/x

def dfdx(x):
    try:
        return -1/x**2
    except RuntimeError:
        return np.nan

app.TangentApprox(f, dfdx, 0.1, 10, '$y=1/x$')
```

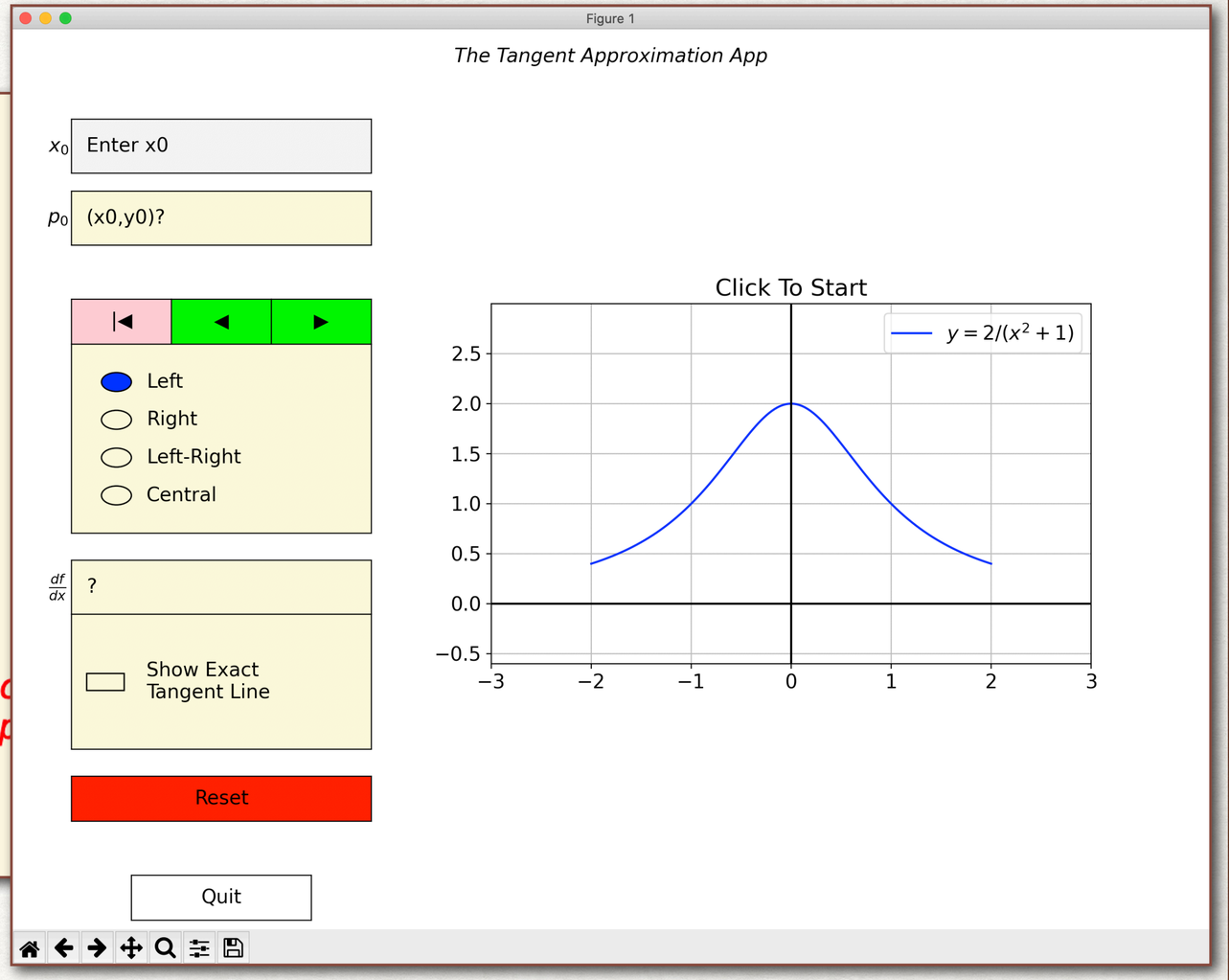



UMass | Dartmouth

# THE APPROXIMATION OF THE DERIVATIVE 24/25

## Testing the App: Example 6

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 26 21:21:02 2021

 Example 6: y = 1/(x^2+1)

@author: ahausknecht
"""
import tangentApproximationAppV1 as app
#import numpy as np

print("Example 6: y = 1/(x^2+1)")

# Use either a lambda function or use a normal func
# if-statements which will cause a problem for nump
f = lambda x:2/(x**2+1)
dfdx = lambda x:-4*x/(x**2+1)**2
app.TangentApprox(f, dfdx, -2, 2, '$y=2/(x^2+1)$')
```




UMass | Dartmouth

AN ACCOMPLISH PROJECT MODULE

# THE END