KOÇ ÜNİVERSİTESİ

Lab06

COMP 125 Programming with Python

Recording Disclaimer

The synchronous sessions are recorded (audiovisual recordings). The students are not required to keep their cameras on during class.

The audiovisual recordings, presentations, readings and any other works offered as the course materials aim to support remote and online learning. They are only for the personal use of the students. Further use of course materials other than the personal and educational purposes as defined in this disclaimer, such as making copies, reproductions, replications, submission and sharing on different platforms including the digital ones or commercial usages are strictly prohibited and illegal.

The persons violating the above-mentioned prohibitions can be subject to the administrative, civil, and criminal sanctions under the Law on Higher Education Nr. 2547, the By-Law on Disciplinary Matters of Higher Education Students, the Law on Intellectual Property Nr. 5846, the Criminal Law Nr. 5237, the Law on Obligations Nr. 6098, and any other relevant legislation.

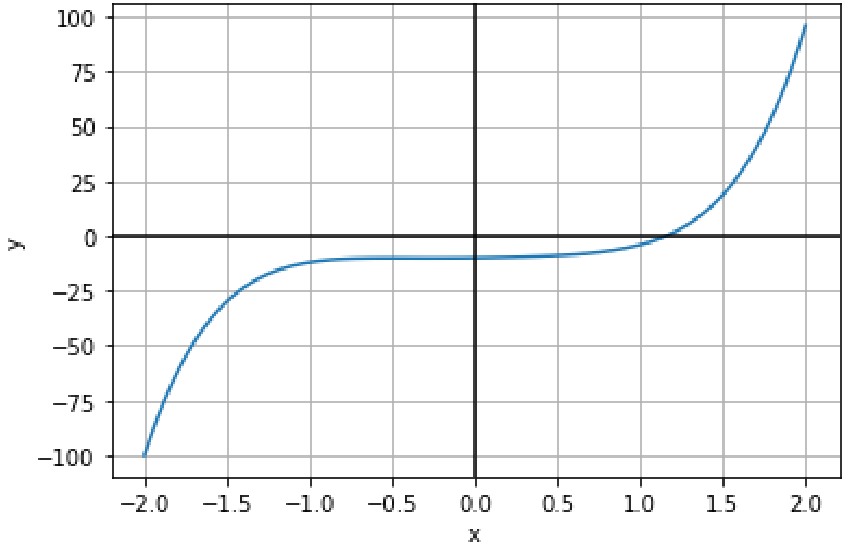
The academic expressions, views, and discussions in the course materials including the audio-visual recordings fall within the scope of the freedom of science and art.

Finding the root(s) of a function

The root of a function f(x) is defined as the value (x), where the function evaluates to zero: f(x)=0

For example $f(x) = 3x^5 + 2x^2 + x - 10$ has one root in the interval [-2,2].

If we plot this function, we get:



How can we find the value of this root?

Bisection Method

Bisection method is a robust, but inefficient, method for finding the root(s) of a function numerically.

Let's suppose that we have two x-values **xpos** and **xneg** where the function has positive and negative values:

f(xpos) > 0 and **f(xneg)<0**.

Since the function changes sign between **xneg** and **xpos**,

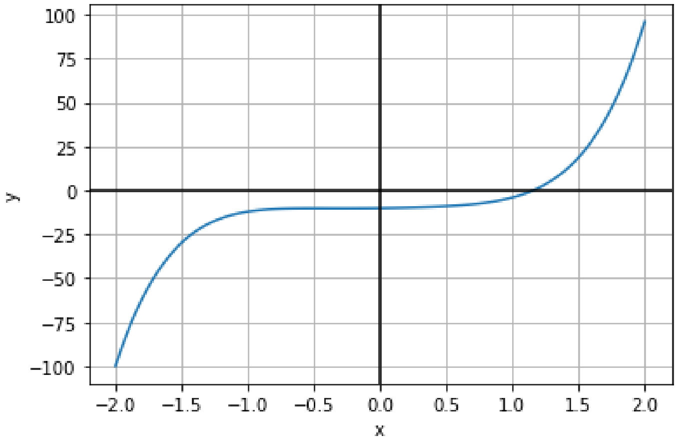
there must be at least one root in the interval **[xneg, xpos]**

For example for the function

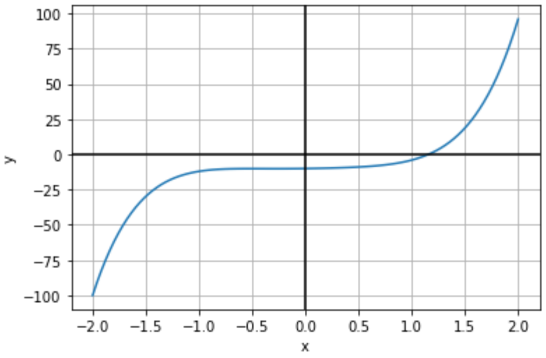
$$f(x) = 3x^5 + 2x^2 + x - 10$$

we can use the interval **[-2,2]**, since

$f(-2) = -100$ and $f(2) = 96$.



Bisection Method: Algorithm



Bisection method is an iterative method. The basic steps are:

1) Find the mid point of xneg and xpos.

$$xnew = \frac{xneg + xpos}{2}$$

2) Evaluate the function at this point

$$f(xnew)$$

3) If $f(xnew) > 0$, set

xpos = xnew,

otherwise set

xneg = xnew.

4) Now we have a new (narrower) interval that includes the root.

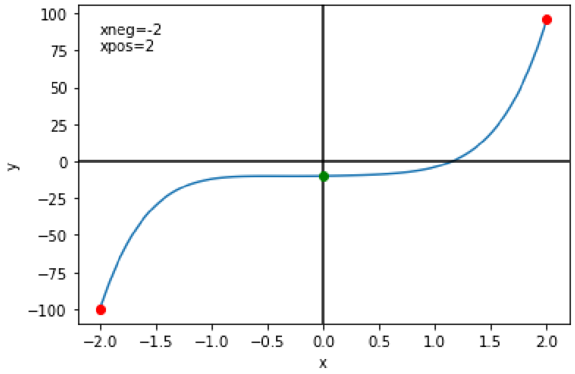
5) Repeat until the interval width

$$\delta = |xpos - xneg|$$

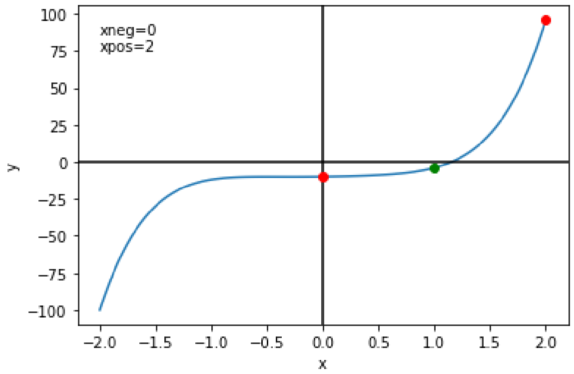
$\text{is less than a certain threshold value (e.g. } 0.000001)$

Let’s try

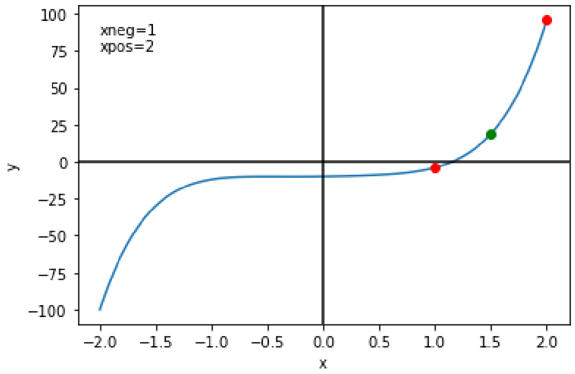
1)



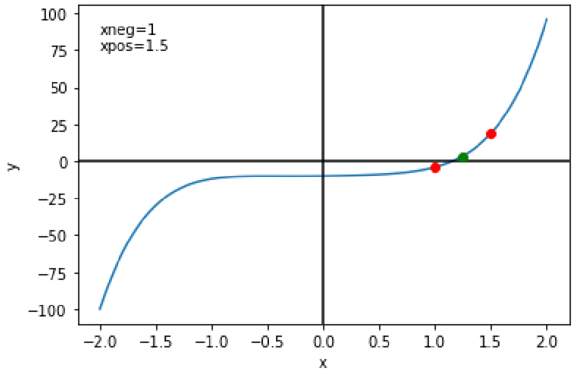
2)



3)



4)



Let’s implement the Bisection method

- Download the file lab06.zip and unzip it to the Desktop.
- Start Spyder and navigate to Lab06 folder

```
In [1]: pwd
Out[1]: 'C:\\Users\\msayar'

In [2]: cd Desktop
C:\Users\msayar\Desktop

In [3]: cd Lab06
C:\Users\msayar\Desktop\Lab06

In [4]: pwd
Out[4]: 'C:\\Users\\msayar\\Desktop\\Lab06'

In [5]:
```

Import plot module

- In the folder Lab06 we have a file “plot.py”

```python
import matplotlib.pyplot as plt
import numpy as np

def plot(func, xmin, xmax):
    """ Requires three arguments:
    func is the name of the function to be plotted.
    xmin and xmax define the x-limits of the plot.
    """
    n_points=100
    x=np.arange(xmin,xmax+(xmax-xmin)/(n_points-1),(xmax-xmin)/(n_points-1))
    f_x=[]
    for i in range(len(x)):
        f_x +=[func(x[i])]

    plt.plot(x,f_x)
    plt.axhline(y=0, color='k')
    plt.axvline(x=0, color='k')
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show(block=False)
    return
```

- Import this module

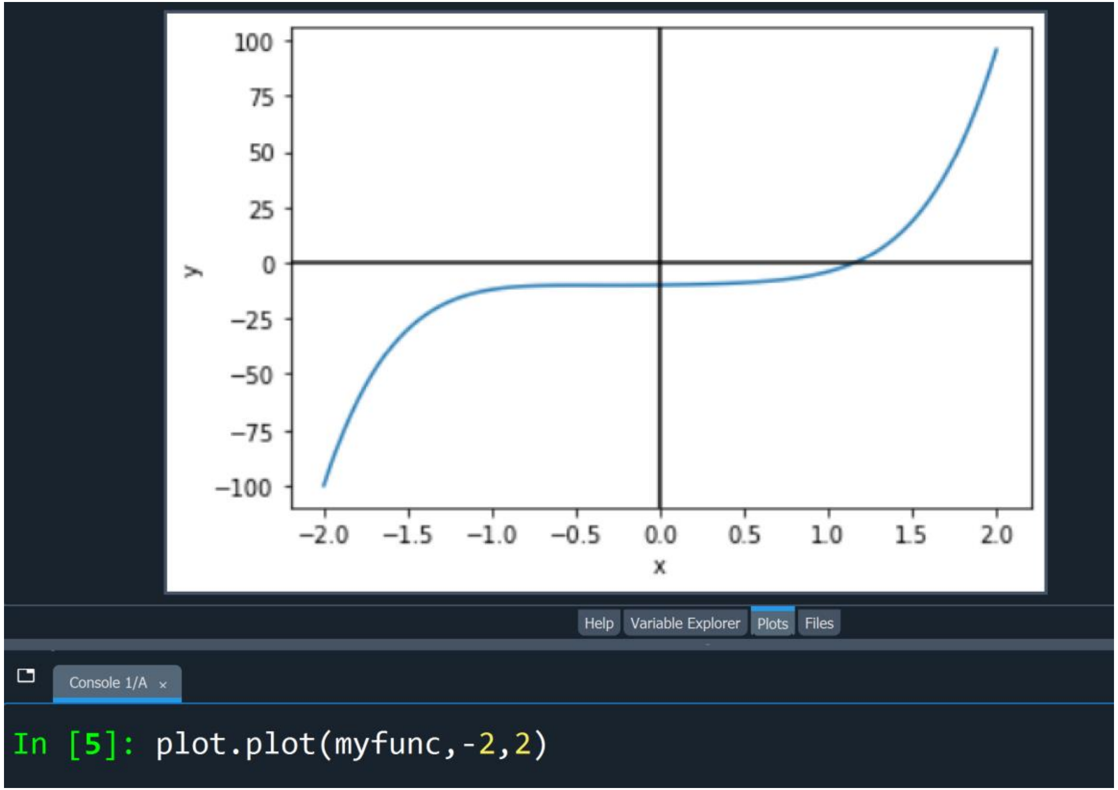
```
In [2]: import plot

In [3]: help(plot.plot)
Help on function plot in module plot:

plot(func, xmin, xmax)
    Requires three arguments:
    func is the name of the function to be plotted.
    xmin and xmax define the x-limits of the plot.
```

Implement “myfunc” function

- Function myfunc
- Should take one argument: x
- Should return the value of $f(x) = 3x^5 + 2x^2 + x - 10$

```python
def myfunc(x):
    """Evaluates the value of the function at x"""

    return
```

Let’s test it



100
75
50
25
0
−25
−50
−75
−100
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x
Help
Variable Explorer
Plots
Files
Console 1/A
In [5]: plot.plot(myfunc,-2,2)

Implement the “bisection_update” function

```
def bisection_update(func, xneg, xpos):
    """
    Performs one iteration of the Bisection algorithm
    Takes three arguments
    1) name of the function f(x))
    2) xneg, such that f(xneg)<0.
    3) xpos, such that f(xpos)>0.

    Returns the updated xneg and xpos values
    """


    "Write your code here"


    return xneg, xpos
```

Implement the “bisection” method

- Should check the given xneg and xpos to make sure that f(xneg)<0 and f(xpos)>0
- If not, return after printing the message: "Please provide two x-values where the sign of the function differs.“
- Use $\delta = 0.00001$as convergence criteria
- Use the “bisection_update” function to iterate until the convergence criteria is satisfied: $\delta = |xpos - xneg|$
- At every iteration print out the current xneg, xpos, delta, f(xneg) and f(pos)
- If found, return the root value

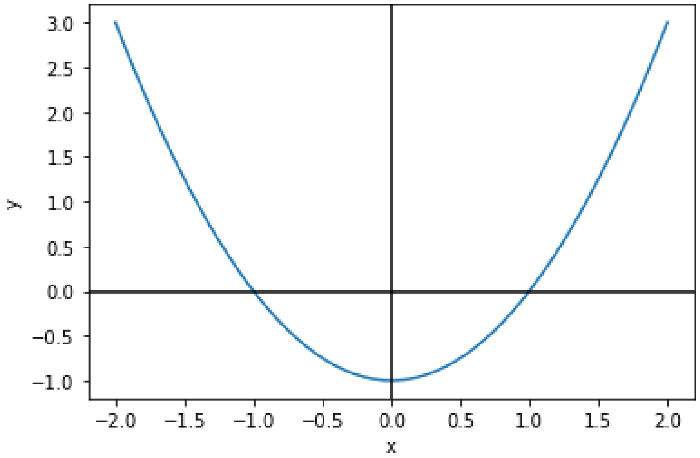
```
def bisection(func, xneg, xpos):
    """ Takes three arguments
    1) name of the function f(x))
    2) xneg, such that f(xneg)<0.
    3) xpos, such that f(xpos)>0.

    Returns the root if found
    """


    "Write your code here"

    return root
```

Sample output

```
xneg=-2.00000e+00, xpos=2.00000e+00, delta=4.00000e+00, f(xneg)=-1.00000e+02, f(xpos)=9.60000e+01
xneg=0.00000e+00, xpos=2.00000e+00, delta=2.00000e+00, f(xneg)=-1.00000e+01, f(xpos)=9.60000e+01
xneg=1.00000e+00, xpos=2.00000e+00, delta=1.00000e+00, f(xneg)=-4.00000e+00, f(xpos)=9.60000e+01
xneg=1.00000e+00, xpos=1.50000e+00, delta=5.00000e-01, f(xneg)=-4.00000e+00, f(xpos)=1.87812e+01
xneg=1.00000e+00, xpos=1.25000e+00, delta=2.50000e-01, f(xneg)=-4.00000e+00, f(xpos)=3.53027e+00
xneg=1.12500e+00, xpos=1.25000e+00, delta=1.25000e-01, f(xneg)=-9.37653e-01, f(xpos)=3.53027e+00
xneg=1.12500e+00, xpos=1.18750e+00, delta=6.25000e-02, f(xneg)=-9.37653e-01, f(xpos)=1.09199e+00
xneg=1.12500e+00, xpos=1.15625e+00, delta=3.12500e-02, f(xneg)=-9.37653e-01, f(xpos)=2.99110e-02
xneg=1.14062e+00, xpos=1.15625e+00, delta=1.56250e-02, f(xneg)=-4.65229e-01, f(xpos)=2.99110e-02
xneg=1.14844e+00, xpos=1.15625e+00, delta=7.81250e-03, f(xneg)=-2.20555e-01, f(xpos)=2.99110e-02
xneg=1.15234e+00, xpos=1.15625e+00, delta=3.90625e-03, f(xneg)=-9.60528e-02, f(xpos)=2.99110e-02
xneg=1.15430e+00, xpos=1.15625e+00, delta=1.95312e-03, f(xneg)=-3.32545e-02, f(xpos)=2.99110e-02
xneg=1.15527e+00, xpos=1.15625e+00, delta=9.76562e-04, f(xneg)=-1.71778e-03, f(xpos)=2.99110e-02
xneg=1.15527e+00, xpos=1.15576e+00, delta=4.88281e-04, f(xneg)=-1.71778e-03, f(xpos)=1.40851e-02
xneg=1.15527e+00, xpos=1.15552e+00, delta=2.44141e-04, f(xneg)=-1.71778e-03, f(xpos)=6.18078e-03
xneg=1.15527e+00, xpos=1.15540e+00, delta=1.22070e-04, f(xneg)=-1.71778e-03, f(xpos)=2.23078e-03
xneg=1.15527e+00, xpos=1.15533e+00, delta=6.10352e-05, f(xneg)=-1.71778e-03, f(xpos)=2.56317e-04
xneg=1.15530e+00, xpos=1.15533e+00, delta=3.05176e-05, f(xneg)=-7.30778e-04, f(xpos)=2.56317e-04
xneg=1.15532e+00, xpos=1.15533e+00, delta=1.52588e-05, f(xneg)=-2.37242e-04, f(xpos)=2.56317e-04
Root of the function myfunc is:  1.155323028564453l
```

Test with the following functions and initial values

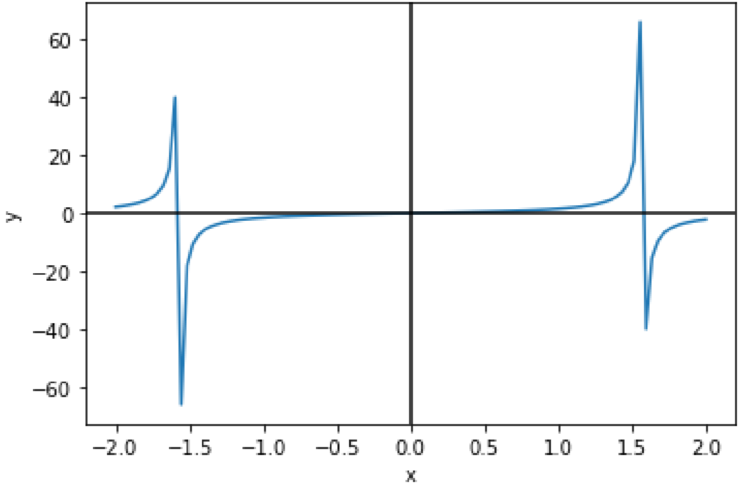
- $f(x) = x^2 - 1; xneg = 0; xpos = 1.5$
- Hint: implement these new functions with new names. You can simply call bisection function by changing the argument name.




```
In [35]: plot.plot(myfunc2,-2,2)

In [36]: bisection(myfunc2,0,1.5)
```

Test with the following functions and initial values

- $f(x) = \tan(x)\,; xneg = -0.5; xpos = 1.$
- Hint: implement these new functions with new names. You can simply call bisection function by changing the argument name.




```
In [42]: plot.plot(myfunc3,-2,2)

In [43]: bisection(myfunc3,-0.5,1)
```

Warning: What happens if you have a singularity.
Test the bisection method with xpos=-2.0 and xneg=-1.0.
Bisection method Works when the function is continuous.