



Ihre bisherigen Kenntnisse in Algorithmen und der Sprache Python verwenden Sie in dieser ersten Übung von Lektion 3, um

- eine Lösung der Gleichung  $f(x) = 0$  zu bestimmen
- ein Optimierungsproblem zu lösen.

#### Lernziele :

- Arbeiten mit einer Funktion in Python
- Einsatz der **while** - Schleife

### Nullstellenbestimmung durch Intervallhalbierung

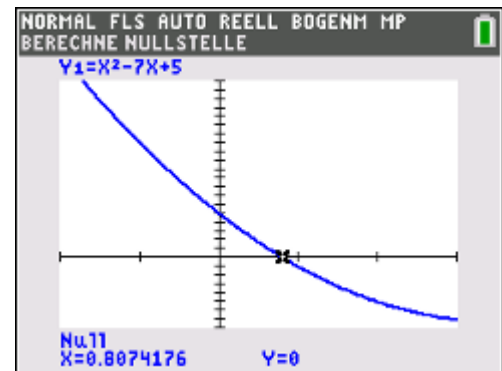
Wir betrachten den Graphen der im Intervall  $[-2,3]$  durch

$$f(x) = x^2 - 7x + 5$$

definierten Funktion  $f$ .

In Python soll nun ein Skript erstellt werden, mit dem die Nullstelle berechnet wird. Dazu wird das folgende Verfahren verwendet:

- Das Intervall  $[a,b] = [-2,3]$  wird halbiert:  $m = \frac{a+b}{2}$ .
- Ist das Vorzeichen von  $f(a)$  ungleich dem von  $f(m)$ , so liegt die Nullstelle im Intervall  $[a,m]$ , andernfalls in  $[m,b]$ .
- Das Intervall, das die Nullstelle enthält, wird nun wieder geteilt.
- Nun werden wieder die Vorzeichen bestimmt und ein neues Intervall ausgesucht.
- Das wird dann wieder geteilt, usw.
- Der Algorithmus läuft, solange die Intervalllänge eine bestimmte Grenze  $dx$  nicht unterschreitet:  $(b-a) > dx$



### Ein erstes Programm

- Schreiben Sie das nebenstehende Skript.
- Alle Befehle und Sonderzeichen erreichen Sie über « Fns .. ».

```

EDITOR: NULLST
PROGRAM LINE 0011
def f(x):
    return x**2-7*x+5
def nst(a,b,dx):
    while (b-a)>dx:
        c=(a+b)/2
        if f(a)*f(c)<=0:
            b=c
        else:
            a=c
    return a,b

```

The image shows a Python editor window titled 'EDITOR: NULLST' with 'PROGRAM LINE 0011'. It contains the Python code for the bisection method. The code defines a function  $f(x)$  and a function  $nst(a,b,dx)$  that uses a **while** loop to iteratively narrow down the interval  $[a,b]$  until its length is less than  $dx$ . The **if** statement checks the sign of the function at the midpoint  $c$  to determine which half of the interval contains the root.



So sieht dann das Ergebnis aus.

```

PYTHON SHELL

>>> # Shell Reinitialized
>>> # Running NULLST
>>> from NULLST import *
>>> nst(-2,3,.01)
(0.802734375, 0.8125)
>>> |
  
```

Andere Lösungen

1. Anstelle der Genauigkeit dx kann man auch die Anzahl n der Teilungen angeben (Bilder rechts).

```

EDITOR: NULLSTZ
PROGRAM LINE 0005

def f(x):
    return x**2-7*x+5
def nst(a,b,n):
    for i in range(n):
        c=(a+b)/2
        if f(a)*f(c)<=0:
            b=c
        else:
            a=c
    return a,b
  
```

```

PYTHON SHELL

>>> # Shell Reinitialized
>>> # Running NULLSTZ
>>> from NULLSTZ import *
>>> nst(-2,3,25)
(0.8074175119400024, 0.8074176609516144)
>>> |
  
```

2. Das Programm kann auch rekursiv formuliert werden (Bild rechts).

```

EDITOR: NULLSTY
PROGRAM LINE 0001

def f(x):
    return x**2-7*x+5
def nst(a,b,dx):
    if (b-a)<=dx:
        return a,b
    else:
        c=(a+b)/2
        if f(a)*f(c)<=0:
            return nst(a,c,dx)
        else:
            return nst(c,b,dx)
  
```