

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

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

**Lernziele :**

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

**Nullstellenbestimmung durch Intervallhalbierung**

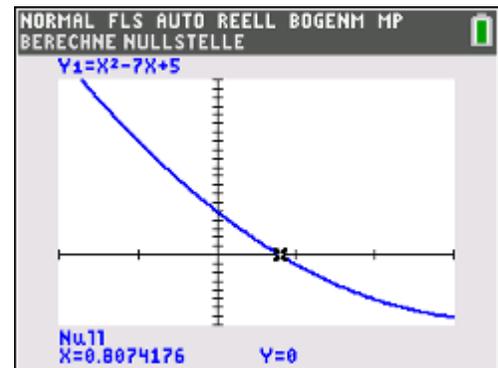
Wir betrachten den Graphen der im Intervall  $[-2,3]$  durch

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

definierten Funktion  $f$ .

In Python soll nun ein Skript erstellt werden, mit dem eine 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**

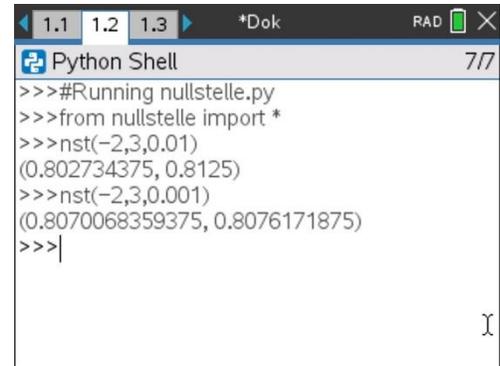
- Schreiben Sie das nebenstehende Programm.

```

1.1 1.2 1.3 *Dok RAD
nullstelle.py saved successfully
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
    
```

Als Ergebnis werden die Grenzen des Intervalls angegeben, das die Nullstelle enthält..



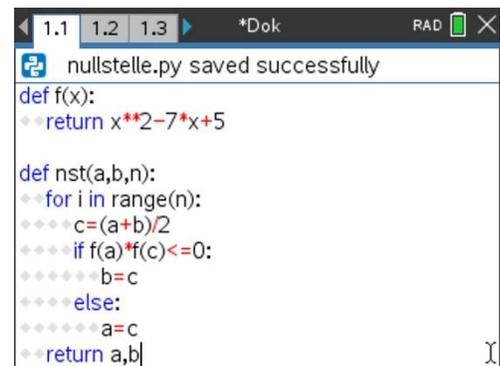
```

1.1 1.2 1.3 *Dok RAD
Python Shell 7/7
>>>#Running nullstelle.py
>>>from nullstelle import *
>>>nst(-2,3,0.01)
(0.802734375, 0.8125)
>>>nst(-2,3,0.001)
(0.8070068359375, 0.8076171875)
>>>

```

Andere Lösungen

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

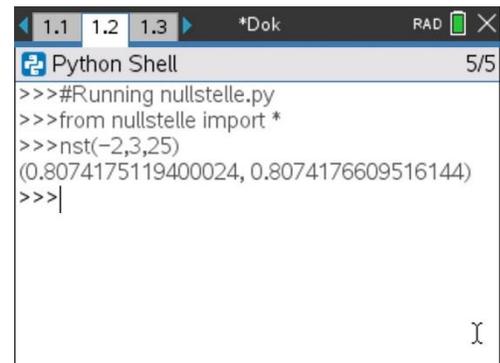


```

1.1 1.2 1.3 *Dok RAD
nullstelle.py saved successfully
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

```

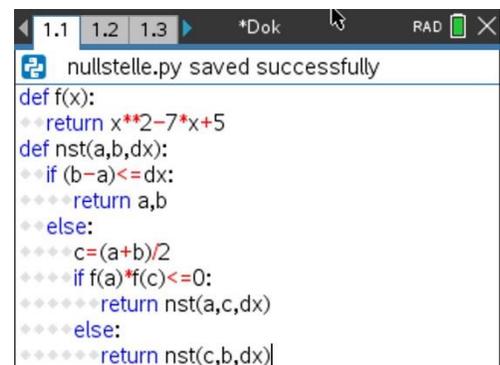


```

1.1 1.2 1.3 *Dok RAD
Python Shell 5/5
>>>#Running nullstelle.py
>>>from nullstelle import *
>>>nst(-2,3,25)
(0.8074175119400024, 0.8074176609516144)
>>>

```

2. Das Programm kann auch rekursiv formuliert werden (Bild rechts). Zu Rekursionen später mehr.



```

1.1 1.2 1.3 *Dok RAD
nullstelle.py saved successfully
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)

```