```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

1 Verschiedene Versionen der Hut-Funktion

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

Diese Version funktioniert NICHT mit arrays, nur mit float oä

```
def N(x):
    if x < 0:
        return 0.0
    elif 0 <= x < 1:
        return x
    elif 1 <= x < 2:
        return 2 - x
    elif x >= 2:
        return 0.0
```

Diese Variante funktioniert NICHT mit float oä (da funktioniert len() nicht)

```
def N_loop(x):
    r = np.zeros(len(x))
    for i in np.arange(len(x)):
        r[i] = N(x[i])
    return r
```

Variante ohne Schleife (auch nur für arrays)

```
def Nv(x):
    condition1 = x < 0
    condition2 = np.logical_and(0 <= x, x < 1)
    condition3 = np.logical_and(1 <= x, x < 2)
    condition4 = x >= 2

    r = np.zeros(len(x))
    r[condition1] = 0.0
    r[condition2] = x[condition2]
    r[condition3] = 2-x[condition3]
    r[condition4] = 0.0
    return r
```

Benutze np.vectorize um die Funktion N zu vektorisieren

```
N_vec = np.vectorize(N)
```

Noch eine (unübersichtliche) Variante mit lambda

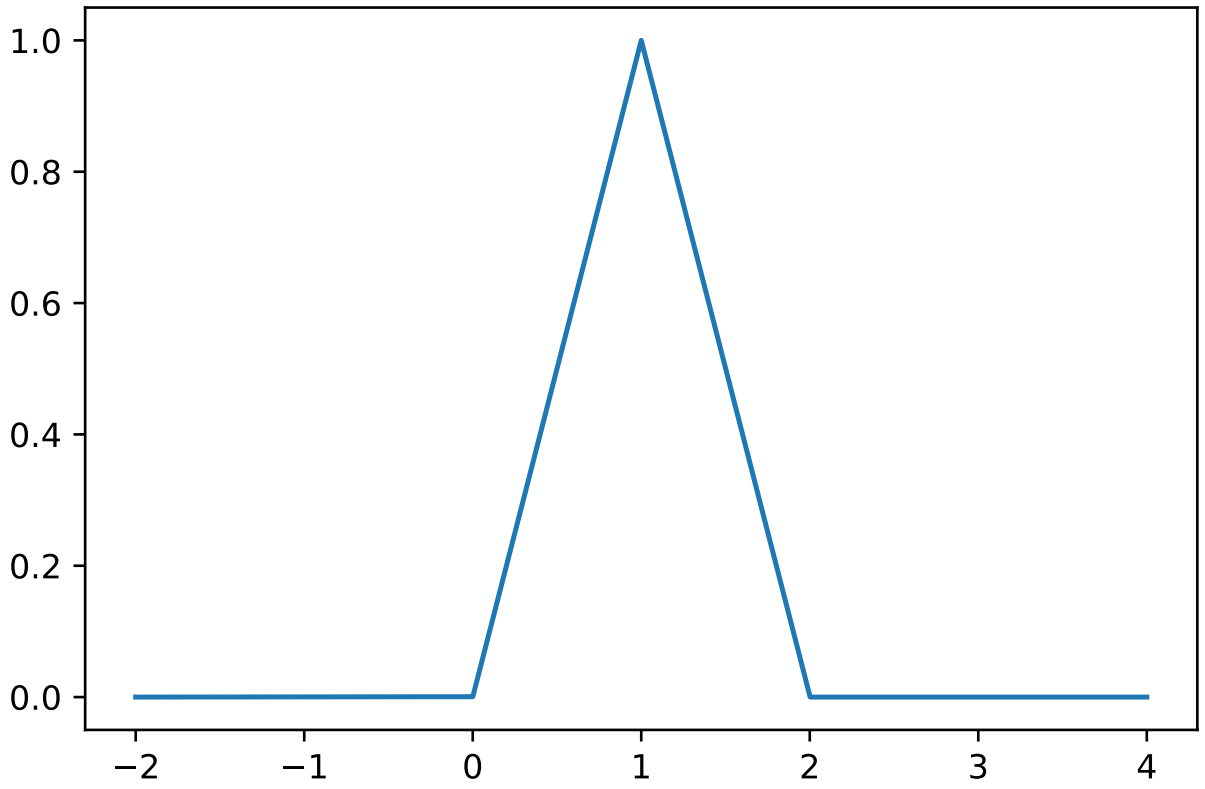
```
N_lambda=lambda x: 0*(x<0)+x*np.logical_and(0<=x,x<1)+(2-x)*np.logical_and(1<=x,x<2)+0*

n = 1000000
x = np.linspace(-2, 4,n+1)


t0 = time.clock()
y=N_vec(x)
t1 = time.clock()

plt.plot(x,y)
plt.show()
print(t1-t0)
```

```
0.3200000000000003
```



Weitere Informationen zum vektorisieren Auswerten von Funktionen finden Sie unter anderem im Buch von Langtangen in Abschnitt 5.5