



math

# math



Das Modul *math* stellt viele Funktionen und Konstanten für mathematische Probleme zur Verfügung.

# math



Das Modul *math* stellt viele Funktionen und Konstanten für mathematische Probleme zur Verfügung.

Erinnerung: verschiedene Arten der Einbindung

```
import math
import math as ...
from math import ...
from math import ... as ...
```

# math



Die Funktion `factorial(n)` liefert die Fakultät von `n`.

- $\text{factorial}(n) = n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 2 \cdot 1$
- nur Ganzzahlen

```
>>> math.factorial(5)
120
>>> math.factorial(3.5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'float' object cannot be interpreted as an integer
```

# math



`math.floor(x)` liefert den größten ganzzahligen Wert kleiner als `x`

`math.ceil(x)` liefert den kleinsten ganzzahligen Wert größer als `x`

Benutzen Sie die beiden Funktionen in einer Funktion `intervall(x)`, die Folgendes produziert:

```
>>> intervall(3.4)
3.4 liegt zwischen 3 und 4
```

# math



`math.trunc( x )` liefert den ganzzahligen Anteil der Kommazahl `x`.

```
>>> math.trunc( 4.3 )  
4  
  
>>> math.trunc( -4.3 )  
-4
```

# math



`math.trunc( x )`

- wie `math.floor()` für positive Zahlen
- wie `math.ceil()` für negative Zahlen

# math

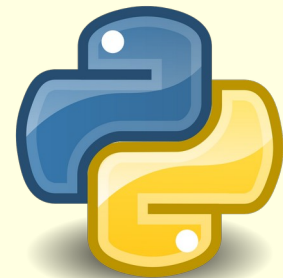


`math.trunc( x )`

- identisch zu `int( x )` für Kommazahlen als Argument
- `int()` verwendet `trunc()`, also `trunc()` schneller



# math



`math.sqrt(x)` berechnet die Quadratwurzel von `x`

- Fehler bei negativen

Werten

```
>>> math.sqrt( 16 )
4.0

>>> math.sqrt( 15.4 )
3.924283374069717

>>> math.sqrt( -15.4 )
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
ValueError: math domain error
```

# math



`math.sqrt(x)` berechnet die Quadratwurzel von `x`

- Vorsicht bei Berechnung  
über `**0.5`

```
>>> math.sqrt( 15.4 )
3.924283374069717

>>> 15.4 ** 0.5
3.924283374069717

>>> -15.4 ** 0.5
-3.924283374069717

>>> (-15.4) ** 0.5
(2.4029305365008273e-16+3.924283374069717j)
```

# math



`math.hypot(x1,x2,x3...)` liefert die Euklidische Norm der Parameter, d.h.

$$\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}$$

```
>>> math.hypot( 15.4 )  
15.4
```

```
>>> math.hypot( 3, 5 )  
5.830951894845301
```

```
>>> math.hypot( 3, 4 )  
5.0
```

# math



## Aufgabe

Schreiben Sie eine Funktion `hypot_vgl(x1,x2,x3...)`, die für drei Parameter einmal via `hypot` einmal mit `sqrt()` die Euklidische Norm berechnet

`hypot_vgl()` soll `True` zurückliefern, wenn der Vergleich der beiden Ergebnisse `True` ergibt, ansonsten `False`