Das math-Modul in Python

Definition:
Das math-Modul ist Teil der Standardbibliothek in Python und stellt mathematische Funktionen und Konstanten zur Verfügung, die auf Gleitkommazahlen arbeiten.

Wenn man Python installiert, werden automatisch etwa 200 Module wie math, string, random, datetime oder os mitgeliefert. Diese gehören zur sogenannten Standardbibliothek und können direkt verwendet werden – ganz ohne zusätzliche Installation oder Download.

Zweck:
Das math-Modul unterstützt bei:

- Mathematischen Berechnungen (z. B. Wurzeln, Logarithmen)
- Nutzung mathematischer Konstanten (z. B. π, e)
- Runden, Potenzen, trigonometrische Funktionen u. v. m.

Das math-Modul in Python

Wichtige Funktionen & Konstanten aus math

```
import math

print(math.sqrt(16))     # Quadratwurzel: 4.0
print(math.pow(2, 3))    # Potenz: 8.0

print(math.pi)           # Kreiszahl π: 3.141592653589793 # https://www.mathsisfun.com/numbers/pi.html
print(math.e)            # Eulersche Zahl: 2.718281828459045 # https://www.mathsisfun.com/numbers/e-eulers-number.html

print(math.floor(4.7))   # Abrunden: 4
print(math.ceil(4.1))    # Aufrunden: 5
```

Weitere nützliche Funktionen:

math.log(x, base) → Logarithmus zur Basis # https://www.mathsisfun.com/algebra/logarithms.html
math.fabs(x) → absoluter Wert # https://www.mathsisfun.com/numbers/absolute-value.html
math.sin(x) → Sinus (x in Radiant) # https://www.mathsisfun.com/sine-cosine-tangent.html

Das math-Modul in Python

Praxisbeispiele

```
# Beispiel 1: Kreisumfang berechnen
import math

radius = 5
umfang = 2 * math.pi * radius
print("Kreisumfang:", umfang)



# Beispiel 2: Quadratwurzeln 1–10
for i in range(1, 11):
    print(f"Wurzel von {i}: {math.sqrt(i):.2f}")



# Beispiel 3: Auf- und Abrunden
zahlen = [4.1, 4.9, -2.3]
for z in zahlen:
    print(f"{z} → floor: {math.floor(z)}, ceil: {math.ceil(z)}")
```

Das math-Modul in Python

Wichtige Hinweise - Winkel, Radiant und Genauigkeit mit math

math-Funktionen erwarten numerische Werte – meist vom Typ int oder float, intern arbeiten sie mit float

Bei Winkel-Funktionen wie sin(), cos() muss der Winkel in Radiant angegeben werden

```
import math

print(math.sin(math.radians(90)))  # Wir berechnen das Sinusverhältnis zu 90°, indem wir zuerst in Radiant umrechnen
                                   # 90 Grad = π/2 Radiant → sin(π/2) = 1.0
                                   # https://www.mathsisfun.com/geometry/radians.html
                                   # https://www.mathsisfun.com/sine-cosine-tangent.html
```

Mit math.pi lässt sich die hohe Genauigkeit mathematischer Konstanten demonstrieren – und gleichzeitig der Einsatz von round() und Formatstrings (f"") zur Steuerung der Ausgabegenauigkeit üben

```
print(math.pi)              #3.141592653589793
print(round(math.pi, 5))    # 3.14159
print(f"{math.pi:.10f}")    # 3.1415926536
```

Das math-Modul in Python

Wichtige Hinweise

- Funktionen aus dem math-Modul vs. eingebaute Funktionen (ohne Import verwendbar) - **

Python stellt manche Funktionen doppelt bereit – sowohl als eingebaute Funktionen als auch über das math-Modul:

Exponentiation: ** vs. math.pow()

```
import math

print(2 ** 3)            # 8   (int)

print(math.pow(2, 3)) # 8.0 (float)
```

- Der Operator ** ist kurz, aber gibt je nach Fall int oder float zurück
- math.pow() gibt **immer float** zurück → besser bei komplexeren Rechnungen

Das math-Modul in Python

Wichtige Hinweise

- Funktionen aus dem math-Modul vs. eingebaute Funktionen (ohne Import verwendbar) - round()

Python stellt manche Funktionen doppelt bereit – sowohl als eingebaute Funktionen als auch über das math-Modul:

Runden: round() vs. math.floor() / math.ceil()

```
import math

print(round(2.4))          #  2 (rundet 2.4 auf die nächste ganze Zahl  → Standardrundung)

print(round(2.5))          # Python verwendet bei .5-Werten das sogenannte "Banker's Rounding":
                           # Wenn genau zwischen zwei Zahlen, wird auf die nächste gerade Zahl gerundet ->  Ergebnis: 2

print(round(3.5))          #  4 (Banker's Rounding → nächste gerade Zahl)

print(math.floor(2.5))   #  2 (immer abrunden)
print(math.ceil(2.5))    #  3 (immer aufrunden)
```

- round() ist eine eingebaute Funktion mit „Banker's Rounding“
- math.floor() / math.ceil() bieten **präzise Steuerung**

Das math-Modul in Python

Wichtige Hinweise & Fehlerquellen

- Funktionen aus dem math-Modul vs. eingebaute Funktionen (ohne Import verwendbar) - abs()

Python stellt manche Funktionen doppelt bereit – sowohl als eingebaute Funktionen als auch über das math-Modul:

Der Betrag (Absolute Value) - der Abstand einer Zahl zu Null – immer positiv, egal ob die Zahl negativ oder positiv ist

```
import math

print(abs(-5))        # 5 (int bleibt int)
print(abs(-3.2))      # 3.2 (float bleibt float)
print(abs(3 + 4j))    # 5.0 (funktioniert mit komplexen Zahlen)

print(math.fabs(-5))    # 5.0 (immer float)
print(math.fabs(-3.2))  # 3.2
# print(math.fabs(3 + 4j)) → Fehler
```

- math.fabs() funktioniert nur mit int oder float
- abs() ist eingebaut und funktioniert auch mit komplexen Zahlen (https://www.mathsisfun.com/numbers/complex-numbers.html)
- math.fabs() ist strenger und liefert immer float zurück → nützlich für präzise numerische Berechnungen

Das math-Modul in Python

Wichtige Punkte

- Das **math**-Modul bietet präzise mathematische Funktionen für wissenschaftliches Rechnen
- Ideal für: Simulationen, Statistik, technische Berechnungen
- Klarer Vorteil gegenüber dem normalen ** oder round()
- Wir nutzen math mehrfach im Kurs – z. B. für Zufallszahlen, Berechnungen und eigene Funktionen – und auch später in der Programmierung, z. B. in Statistik, Simulationen oder wissenschaftlichen Anwendungen.

Mehr erkunden

https://docs.python.org/3/library/math.html
https://www.w3schools.com/python/module_math.asp
https://realpython.com/python-math-module/