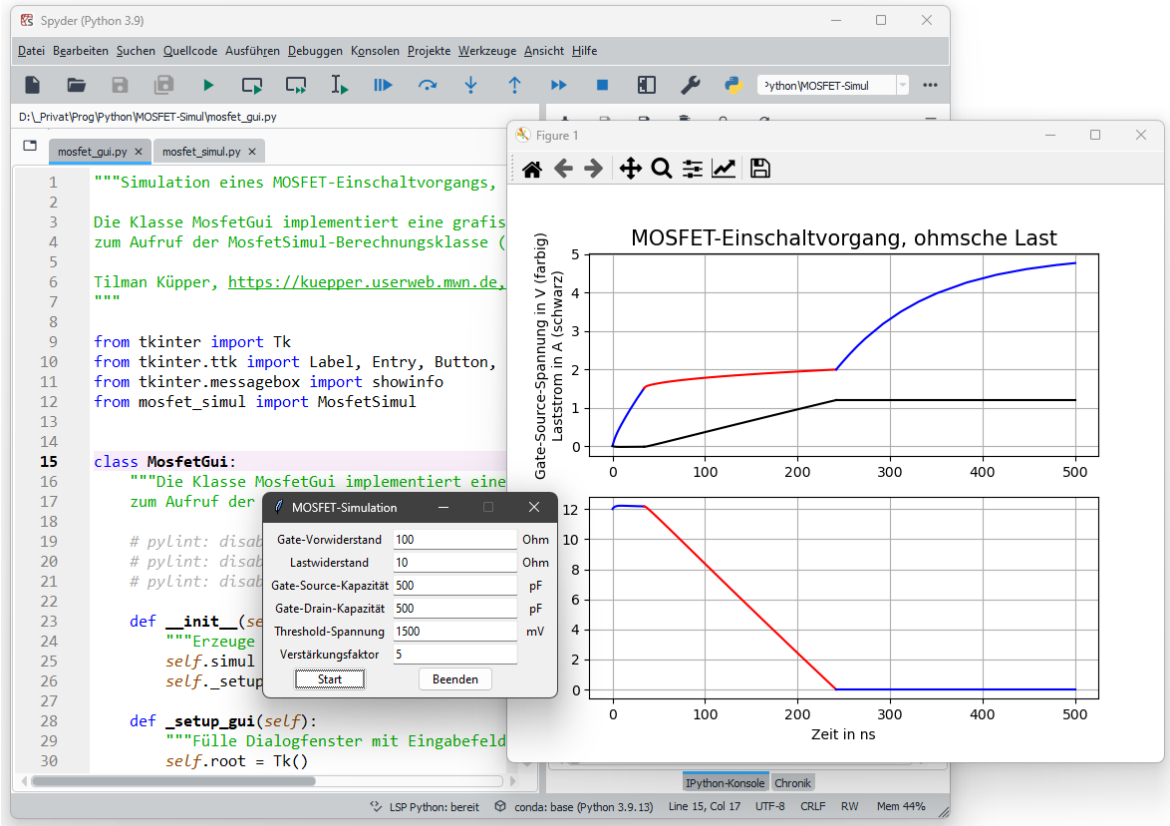
Tilman Küpper

Python-Programmierung

Aufgaben und Lösungen



Hochschule München
https://kuepper.userweb.mwn.de/

Vorwort

Diese Aufgabensammlung ist aus Lehrveranstaltungen an der Fakultät für Maschinenbau, Fahrzeugtechnik und Luftfahrttechnik der Hochschule München hervorgegangen. Die Studierenden erlernen hier zu Beginn ihres Ingenieurstudiums die Grundlagen der Python-Programmierung.

Die einzelnen Kapitel der Aufgabensammlung entsprechen in Inhalt und Niveau den Rechnerübungen, die während des Semesters stattfinden. Zu allen Aufgaben sind Lösungsvorschläge angegeben, nicht jedoch zu den Zusatzaufgaben. Es wird ausdrücklich empfohlen, gerade auch die Zusatzaufgaben selbstständig zu bearbeiten.

Die Aufgabensammlung wird fortlaufend weiterentwickelt und ergänzt. Die jeweils aktuelle Version der Aufgabensammlung kann von der Homepage des Autors heruntergeladen werden: https://kuepper.userweb.mwn.de/

Haben Sie Fehler oder Unklarheiten gefunden? Haben Sie Anregungen zu Form oder Inhalt? Dann melden Sie sich bitte bei: tilman.kuepper@hm.edu

München, 6. April 2024

Tilman Küpper

Inhaltsverzeichnis

1 Grundlagen

1.1 Umrechnung von kW nach PS

Programmieren Sie ein Python-Skript zur Umrechnung von Leistungsangaben in kW (Kilowatt) nach PS. Das Skript soll so ablaufen, wie im Bildschirmfoto (links) gezeigt.

Hinweis: 1 kW entspricht 1,35962 PS.

```
Konsole 1/A ×
Leistung in Kilowatt: 75
75.0 kW entspricht 101.9715 PS.
```

```
Konsole 1/A ×
(1) kW -> PS oder (2) PS -> kW umrechnen: 2

PS eingeben: 102
102.0 PS entspricht 75.02096173930951 kW.
```

```
Konsole 1/A ×
(1) kW -> PS oder (2) PS -> kW umrechnen: 3
Eingabefehler!
```

Zusatzaufgabe

- Erweitern Sie das von Ihnen programmierte Skript: Nach dem Start wird zunächst gefragt, ob eine Umrechnung von kW nach PS oder von PS nach kW gewünscht ist (Bildschirmfoto, rechts).
- Bei einer ungültigen Eingabe wird eine entsprechende Fehlermeldung ausgegeben (Bildschirmfoto, unten).

1.2 Mitternachtsformel

Programmieren Sie ein Skript, das die Nullstellen quadratischer Gleichungen berechnet und auf dem Bildschirm ausgibt:

$$ax^2 + bx + c = 0 \tag{1.1}$$

- Zur Berechnung der Nullstellen nutzen Sie die sog. Mitternachtsformel.
- Nach Eingabe der Koeffizienten a, b und c werden beide Nullstellen x_1 und x_2 berechnet und ausgegeben (Bildschirmfoto, links).
- Welche Ergebnisse erwarten Sie für $a = 1$, $b = 0$ und $c = 1$?
- Falls keine reellen Nullstellen existieren, soll die Meldung „keine reellen Nullstellen“ ausgegeben werden.

```
Konsole 1/A ×
a: 1

b: 0

c: -2

Nullstellen:
  x1 = -1.4142135623730951
  x2 = 1.4142135623730951
```

```
Konsole 1/A ×
a: 1

b: 2

c: 1

Doppelte Nullstelle bei x = -1.0
```

Zusatzaufgabe

Bei doppelten Nullstellen soll eine Ausgabe in der Form „doppelte Nullstelle bei x = -1.0“ erfolgen (Bildschirmfoto, rechts).

2 Verzweigungen

2.1 Stromrechnung

Programmieren Sie ein Skript zur Erstellung einer Stromrechnung. Nach Eingabe des Verbrauchs in kWh berechnet das Skript den zu zahlenden Rechnungsbetrag und gibt ihn mit zwei Nachkommastellen auf dem Bildschirm aus:

- Für die ersten 2500 kWh beträgt der Preis 0,40 € pro kWh,
- für die nächsten 2500 kWh beträgt der Preis 0,35 € pro kWh,
- oberhalb von 5000 kWh beträgt der Preis 0,30 € pro kWh.

```
Konsole 1/A ×
Bitte Verbrauch in kWh eingeben: 3000
Rechnungsbetrag: 1175.00 EUR

In [2]:
```

Beispiel, Verbrauch von 3000 kWh: $P = 2500 \cdot 0{,}40\,€ + 500 \cdot 0{,}35\,€ = 1175{,}00\,€$

Zusatzaufgabe

- Fragen Sie nach der Ausgabe des Rechnungsbetrags, ob eine weitere Berechnung erfolgen soll (Eingabe: 1) oder nicht (Eingabe: 0).
- Programmieren Sie auch die umgekehrte Berechnung: Nach der Eingabe eines Preises in EUR wird der dazugehörige Verbrauch in kWh auf dem Bildschirm ausgegeben.

2.2 Maximum und Median

Das folgende Python-Skript schreibt das Maximum von a und b in die Variable maxi.

```
a = int(input("a eingeben: "))
b = int(input("b eingeben: "))

if a > b:
    maxi = a
else:
    maxi = b

print(f"Das Maximum ist: {maxi}")
```

Erweitern Sie das Skript so, dass nun drei Werte a, b und c eingegeben werden. Das Maximum soll wiederum in die Variable maxi übertragen und ausgegeben werden.

Zusatzaufgabe

Geben Sie zusätzlich zum Maximum von a, b und c auch den Median aus.

2.3 Body-Mass-Index

Erstellen Sie ein Python-Skript, welches aus Körpermasse („Gewicht“) m in kg und Größe l in m den sog. Body-Mass-Index berechnet:

$$\text{BMI} = m/l^2 \tag{2.1}$$

Geben Sie zusätzlich zum berechneten Body-Mass-Index aus, ob es sich dabei um Untergewicht, Normalgewicht, leichtes Übergewicht oder Übergewicht handelt:

BMI < 20	Untergewicht
20 ≤ BMI < 25	Normalgewicht
25 ≤ BMI < 30	Leichtes Übergewicht
30 ≤ BMI	Übergewicht

Zusatzaufgabe

- Programmieren Sie die Verzweigungen auf unterschiedliche Arten, etwa mit verschachtelten if-else-Anweisungen oder mit einer Kette von elif-Anweisungen.
- Zeichnen Sie die Struktogramme zu Ihren Lösungen.

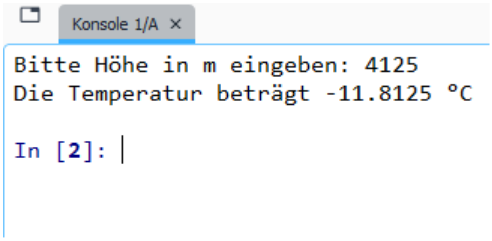
2.4 ICAO-Standardatmosphäre

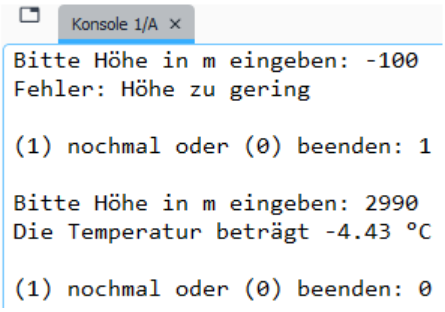
Die ICAO-Standardatmosphäre wird bei der Konstruktion, Berechnung und Prüfung von Luftfahrzeugen sowie bei der Verarbeitung von Daten aus geophysikalischen und meteorologischen Beobachtungen genutzt. Sie beschreibt den Temperatur- und Druckverlauf in unterschiedlichen Höhen, wobei zwischen den in der Tabelle angegebenen Werten linear interpoliert wird.[1]

Geopot. Höhe in km	-5	0	11	20	32	47	51	71	80
Temperatur in °C	47,5	15,0	-56,5	-56,5	-44,5	-2,5	-2,5	-58,5	-76,5

Programmieren Sie ein Skript zur Berechnung der Temperatur in einer bestimmten Höhe (Bildschirmfoto, links):

- Zunächst wird die Höhe (in Metern) eingegeben. Bei einer negativen Höhe oder einer Höhe > 47000 m erfolgt eine Fehlermeldung, ansonsten wird die Temperatur anhand der ICAO-Standardatmosphäre berechnet und ausgegeben.
- Die in der Tabelle angegebenen Werte für Höhen unterhalb von 0 m bzw. oberhalb von 47000 m werden in dieser Aufgabe also nicht berücksichtigt.





[1] International Civil Aviation Organization: Manual of the ICAO Standard Atmosphere extended to 80 kilometres (262 500 feet), 3rd Edition, Doc 7488/3, 1993.

Zusatzaufgabe

- Geben Sie die berechnete Temperatur mit zwei Nachkommastellen aus.
- Falls gewünscht, sollen in einer Schleife immer weitere Berechnungen ablaufen (Bildschirmfoto, rechts).
- Zeichnen Sie die Struktogramme zu Ihren Lösungen.

3 Schleifen, Matplotlib

3.1 Fakultät

Schreiben Sie ein Skript zur Berechnung der Fakultät $n!$ einer natürlichen Zahl n:

$$n! = 1 \cdot 2 \cdot 3 \cdots n \tag{3.1}$$

- Zu Beginn des Skripts wird der Wert n eingegeben.
- Falls ein ungültiger Wert $n < 1$ oder $n > 50$ eingegeben wurde, erfolgt eine Fehlermeldung. In diesem Fall wird die Eingabe wiederholt.
- Nach der Eingabe eines gültigen Werts wird die Fakultät $n!$ in einer while-Schleife berechnet und auf dem Bildschirm ausgegeben.

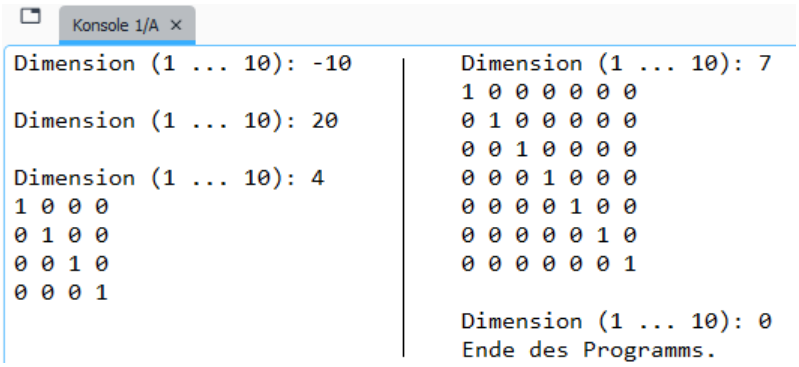
Zusatzaufgabe

Nutzen Sie zur Berechnung der Fakultät eine for-Schleife anstelle der zunächst verwendeten while-Schleife.

3.2 Einheitsmatrix

Programmieren Sie ein Python-Skript zur Ausgabe einer quadratischen Einheitsmatrix auf dem Bildschirm. Bei einer Einheitsmatrix sind die Elemente auf der Hauptdiagonalen gleich eins, alle anderen Elemente sind gleich null.

- Zunächst wird die Dimension der Matrix über die Tastatur eingegeben.
- Negative Dimensionen oder Dimensionen größer als 10 sind nicht zulässig. Die Eingabe wird solange wiederholt, bis eine gültige Dimension eingegeben wird.
- Bei einer Eingabe von 0 (null) wird das Skript beendet.



Zusatzaufgabe

Programmieren Sie das Skript auf unterschiedliche Arten:

- Verwenden Sie zunächst zwei verschachtelte Schleifen. Mit der äußeren Schleife durchlaufen Sie die Zeilen der Matrix, mit der inneren die Spalten.
- Versuchen Sie, das Skript mit einer einzigen Schleife zu programmieren.
- Ersetzen Sie zur Übung alle for-Schleifen durch while-Schleifen und umgekehrt.
- Erstellen Sie Struktogramme von Ihren Lösungen.

3.3 Würfelspiel

Schreiben Sie ein Python-Skript zur Simulation eines Würfelspiels:

- Es werden 1000 ganzzahlige Zufallszahlen im Bereich 1 ... 6 erzeugt.
- Geben Sie auf dem Bildschirm aus, welche Zahl wie oft gewürfelt wurde.
- Zur Erzeugung der Zufallszahlen importieren Sie zunächst die Funktion `randint`: `from random import randint`
- Die einzelnen Zufallszahlen erzeugen Sie mit `w = randint(1, 6)`.

3.4 Funktionswerte berechnen

Das folgende Skript berechnet die Funktion $f(x) = 2x - 2$ im Intervall $-2 \leq x \leq 2$. Ändern bzw. erweitern Sie das Skript wie angegeben:

```
x = -2
while x <= 2:
    y = 2 * x - 2
    print(f"{x} {y}")
    x = x + 0.25
```

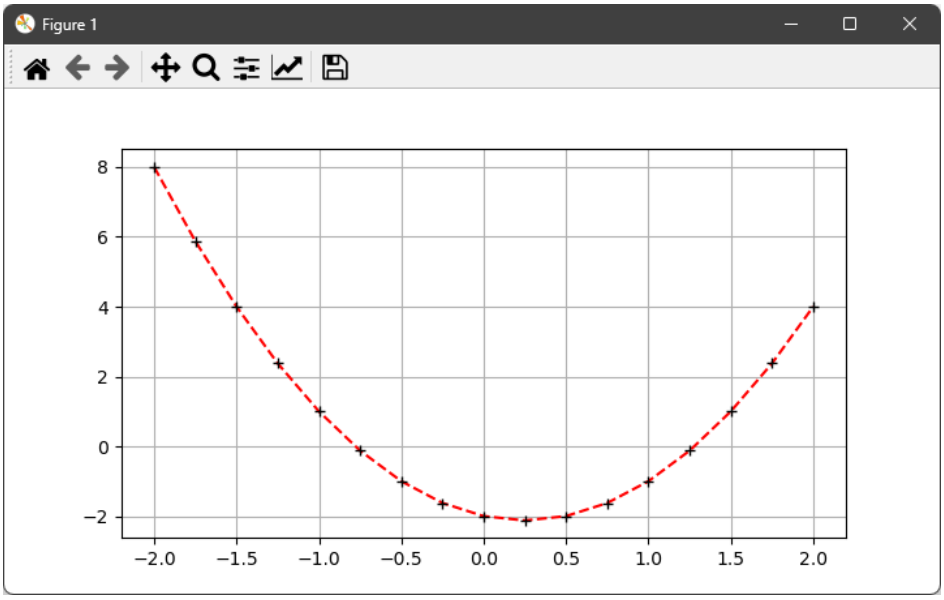
- Nun soll die Funktion $f(x) = 2x^2 - x - 2$ berechnet werden.
- Alle Werte sollen tabellarisch mit drei Nachkommastellen ausgegeben werden.
- Das gewünschte x-Intervall soll per Tastatur eingegeben werden.

3.5 Funktionsgraphen zeichnen

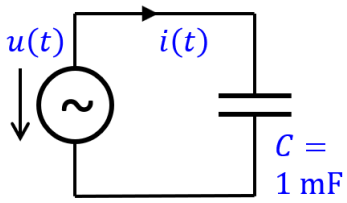
Erweitern Sie das Skript aus Aufgabe 3.4 so, dass zusätzlich zur Ausgabe der Funktionswerte der Funktionsgraph gezeichnet wird. Variieren Sie zur Übung die Linienform des Funktionsgraphen (durchgezogen, gestrichelt usw.) und auch seine Farbe.

Zusatzaufgabe

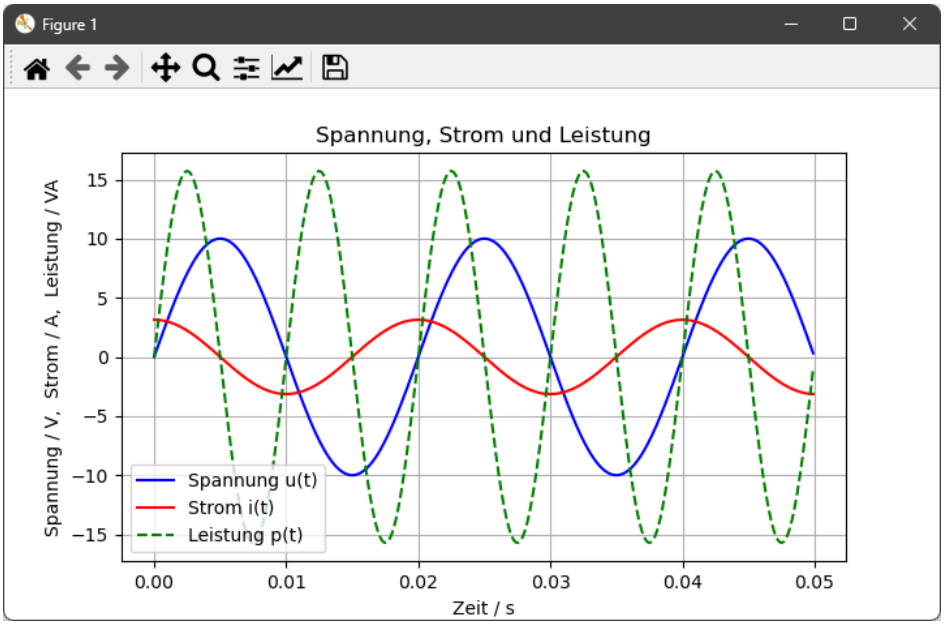
- Zeichen Sie nun die Sinuskurve $f(x) = \sin(x)$ im Intervall $-2\pi \leq x \leq 2\pi$.
- Wie viele x- und y-Werte werden in Ihrem Skript berechnet?
- Wie groß ist die Schrittweite Δx zwischen den einzelnen x-Werten?
- Variieren Sie die Schrittweite Δx und beobachten Sie, wie sich die grafische Darstellung verändert.



3.6 Spannung, Strom und Leistung



Ein Kondensator $C = 1\,\mathrm{mF}$ ist an eine Wechselspannungsquelle $u(t) = 10\,\mathrm{V} \cdot \sin(2\pi f t)$ angeschlossen. Die Frequenz der Wechselspannung beträgt $f = 50\,\mathrm{Hz}$. Es fließt der Strom $i(t) = 1\,\mathrm{A} \cdot \pi \cdot \cos(2\pi f t)$. Stellen Sie die zeitlichen Verläufe von $u(t)$, $i(t)$ und der Leistung $p(t)$ (= Produkt aus Spannung und Strom) grafisch dar.



Zusatzaufgabe

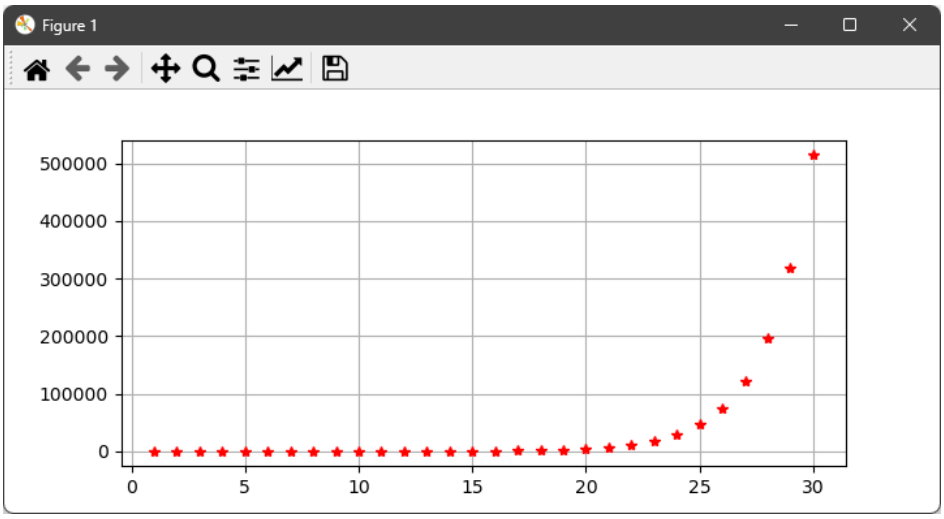
- Fügen Sie den Titel „Spannung, Strom und Leistung“ zum Diagramm hinzu.
- Fügen Sie Achsenbeschriftungen und eine Legende zum Diagramm hinzu.

3.7 Fibonacci-Folge

Die Fibonacci-Folge ist eine mathematische Folge, bei der jede Zahl die Summe der beiden vorangegangenen ist. Sie wurde erstmals im 13. Jahrhundert von Leonardo Fibonacci beschrieben. Schreiben Sie ein Python-Skript, welches die ersten 30 Fibonacci-Zahlen auf dem Bildschirm auflistet: 0, 1, 1, 2, 3, 5, 8, 13 ...

Zusatzaufgabe

Stellen Sie die ersten 30 Fibonacci-Zahlen in grafischer Form dar.



4 Funktionen

4.1 Quersumme

Schreiben Sie eine Funktion „qsum(zahl)“ zur Berechnung der Quersumme einer ganzen Zahl. Die Quersumme einer Zahl ist die Summe ihrer Ziffern. Die Quersumme von 987 ist gleich 9 + 8 + 7, also 24.

Die ganze Zahl wird als Übergabeparameter an die Funktion „qsum(zahl)“ übergeben, die berechnete Quersumme wird als Rückgabewert zurückgegeben. Falls die Funktion mit einem negativen Wert aufgerufen wird, ist der Rückgabewert gleich 0 (null).

Programmieren Sie auch ein Skript, in dem zunächst eine Zahl eingegeben, anschließend die Funktion „qsum(zahl)“ aufgerufen und schließlich die Quersumme auf dem Bildschirm ausgegeben wird.

Tipp: Verwenden Sie (zahl % 10), um die letzte Ziffer einer ganzen Zahl zu ermitteln.

4.2 Primzahlen

Die folgende Funktion prüft, ob die als Parameter übergebene Zahl eine Primzahl ist (Rückgabewert: True) oder nicht (Rückgabewert: False).

```
def primzahl(zahl):
    if zahl < 2:
        return False

    for t in range(2, zahl):
        if zahl % t == 0:
            return False

    return True
```

Programmieren Sie ein Skript, das nach der Eingabe von zwei ganzzahligen Werten a und b alle Primzahlen im Intervall a ... b ermittelt und auf dem Bildschirm ausgibt.

Zusatzaufgabe

- Geben Sie auch die Anzahl der Primzahlen im Intervall a ... b aus.
- Zeichnen Sie ein Struktogramm der Funktion „primzahl(zahl)“ .

4.3 Mitternachtsformel

Das folgende Python-Skript berechnet die Nullstellen der quadratischen Gleichung $ax^2 + bx + c = 0$ und gibt die Ergebnisse auf dem Bildschirm aus (vergl. Aufgabe 1.2).

```
a = float(input("a: "))
b = float(input("b: "))
c = float(input("c: "))

n, x1, x2 = qsolve(a, b, c)  # Die Definition von qsolve fehlt noch.

if n == 0:
    print("Keine reellen Nullstellen!")
elif n == 1:
    print(f"Doppelte Nullstelle bei {x1:.2f}")
else:
    print(f"Nullstellen bei {x1:.2f} und {x2:.2f}")
```

Die Funktion „qsolve(a, b, c)“ hat mehrere Rückgabewerte:

- n: Anzahl der reellen Nullstellen (0, 1 oder 2)
- x1, x2: Nullstellen

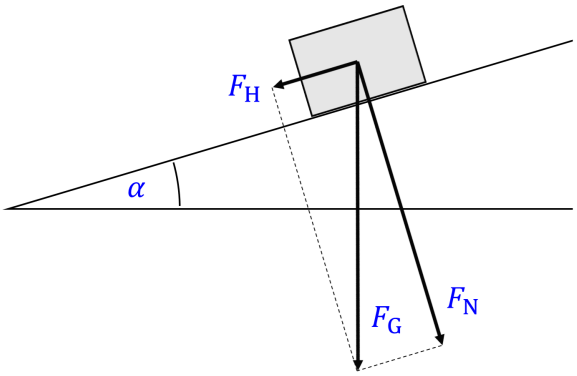
Falls es keine reellen Nullstellen gibt, wird für x1 und x2 jeweils 0 (null) zurückgegeben. Fügen Sie die fehlende Definition der Funktion „qsolve(a, b, c)“ zum Skript hinzu.

Zusatzaufgabe

Falls es keine reellen Nullstellen gibt, soll das Skript stattdessen die beiden komplexen Nullstellen der quadratischen Gleichung ermitteln und auf dem Bildschirm ausgeben. Tipp: Importieren Sie anstelle von „math“ das Modul „cmath“.

4.4 Zerlegung eines Kraftvektors

Ein Körper der Masse m liegt auf einer schiefen Ebene mit dem Neigungswinkel α. Für die Fallbeschleunigung gilt: $g = 9{,}81\,\mathrm{m/s^2}$



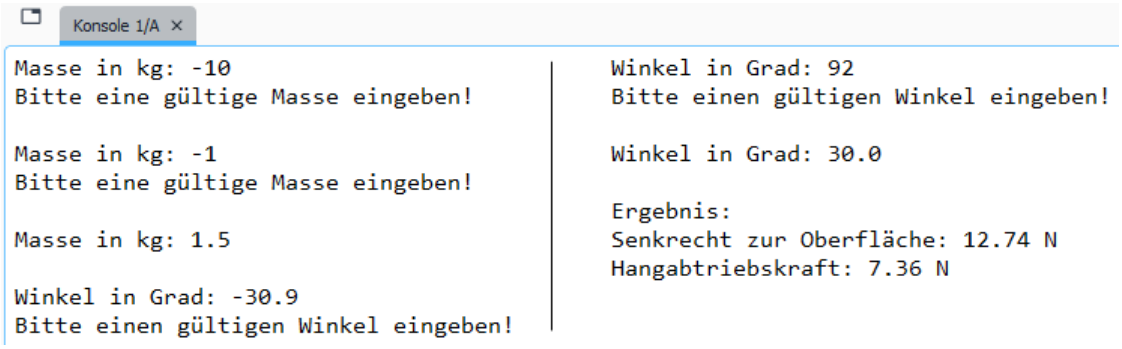
Fügen Sie die noch fehlende Definition der Funktion „kraft(m, alpha)“ zum abgebildeten Skript hinzu. Diese Funktion zerlegt die Gewichtskraft $F_G = m \cdot g$ in die zwei Komponenten F_N (senkrecht zur Oberfläche) und F_H (parallel zur Oberfläche, Hangabtriebskraft).

```
m = float(input("Masse in kg: "))
alpha = float(input("Winkel in Grad: "))

fn, fh = kraft(m, alpha)   # Die Definition von kraft fehlt noch.

print(f"Senkrecht zur Oberfläche: {fn:.2f} N")
print(f"Hangabtriebskraft: {fh:.2f} N")
```

Die Funktion „kraft(m, alpha)“ gibt die beiden Ergebnisse F_N und F_H als Rückgabewerte zurück, sie hat also mehr als einen Rückgabewert.

Zusatzaufgabe

- Falls der eingegebene Winkel nicht im Bereich $0° \leq \alpha \leq 90°$ liegt, wird die Eingabe solange wiederholt, bis ein gültiger Wert vorliegt.
- Auch nach der Eingabe einer negativen Masse m erfolgt eine erneute Abfrage (siehe Bildschirmfoto).



4.5 Euklidischer Algorithmus

Legen Sie drei Listen aa, bb und cc an. Füllen Sie die Listen aa und bb mit jeweils 100 ganzzahligen Zufallszahlen im Bereich 1 ... 100.

Programmieren Sie eine Funktion „ggT(a, b)“ zur Berechnung des größten gemeinsamen Teilers (ggT) von a und b. Verwenden Sie dazu den euklidischen Algorithmus.[2]

Mithilfe der Funktion „ggT(a, b)“ wird nun der ggT von aa[0] und bb[0] berechnet und in cc[0] gespeichert. Der ggT von aa[1] und bb[1] wird in cc[1] gespeichert, der ggT von aa[2] und bb[2] wird in cc[2] gespeichert und so weiter.

Geben Sie die in den drei Listen gespeicherten Elemente tabellarisch auf dem Bildschirm aus.

Zusatzaufgabe

- Geben Sie auch die kleinsten gemeinsamen Vielfache (kgV) der Elemente in den Listen aa und bb aus.
- Geben Sie die Ergebnisse in umgekehrter Reihenfolge auf dem Bildschirm aus, also beginnend bei Nr. 100 rückwärts bis Nr. 1.

4.6 Numerische Integration

Das bestimmte Integral der Funktion $f(x) = (\sin x)^2$ auf dem Intervall $[a; b]$ soll mit einem „Monte-Carlo-Verfahren“ folgendermaßen numerisch ermittelt werden:

Zwei Variablen z1 und z2 werden zunächst auf null gesetzt, die Integrationsgrenzen a und b werden per Tastatur eingegeben. Eine Schleife wiederholt die folgenden Schritte 10000 mal:

- Es wird ein zufälliger x-Wert (Kommazahl, float) im Bereich $[a; b]$ erzeugt.
- Es wird ein zufälliger y-Wert (Kommazahl, float) im Bereich $[0; 1]$ erzeugt.
- Wenn $f(x) < y$ ist, dann wird z1 inkrementiert, ansonsten wird z2 inkrementiert.
- Der Näherungswert A_1 des gesuchten Integrals wird berechnet und auf dem Bildschirm ausgegeben: $A_1 = z1 \cdot (b - a)/(z1 + z2)$.

Definieren Sie zur Berechnung von $f(x)$ bzw. $F(x)$ (Zusatzaufgabe) zwei separate Funktionen: An diese Funktionen werden jeweils einzelne x-Werte übergeben, die daraus berechneten Werte $f(x)$ bzw. $F(x)$ werden als Rückgabewerte zurückgegeben.

Zusatzaufgabe

- Berechnen Sie über die Stammfunktion $F(x) = (x - \sin x \cdot \cos x)/2$ den exakten Wert A_2 des Integrals.
- Geben Sie zusätzlich zu den berechneten Werten A_1 und A_2 auch deren prozentuale Abweichung auf dem Bildschirm aus.

[2] Details zum euklidischen Algorithmus finden Sie zum Beispiel unter `https://de.wikipedia.org/wiki/Euklidischer_Algorithmus` (Stand: 21.01.2024).

4.7 Reihen- und Parallelschaltung von Widerständen

Das folgende Foto zeigt ein Widerstandssortiment für Hobbyelektroniker. Es enthält 20 unterschiedliche Widerstandswerte: 10 Ω, 47 Ω, 100 Ω, 150 Ω, 220 Ω, 330 Ω, 470 Ω, 1 kΩ, 1,5 kΩ, 2,2 kΩ, 4,7 kΩ, 10 kΩ, 22 kΩ, 27 kΩ, 33 kΩ, 47 kΩ, 100 kΩ, 220 kΩ, 470 kΩ und 1 MΩ.

Falls für eine elektronische Schaltung Widerstandswerte benötigt werden, die nicht im Sortiment enthalten sind, kann man sich oft mit Parallel- oder Reihenschaltungen behelfen.

Programmieren Sie ein Skript, das zu einem bestimmten Widerstandswert („Sollwert“) Parallel- und Reihenschaltungen von zwei Widerständen vorschlägt, die diesem Widerstandswert möglichst nahe kommen. Parallel- und Reihenschaltungen von mehr als zwei Widerständen sollen in dieser Aufgabe nicht berücksichtigt werden.

Um das Skript übersichtlich zu halten, programmieren Sie bitte separate Funktionen zur Ermittlung des besten Einzelwiderstands, zur Ermittlung der besten Reihenschaltung sowie zur Ermittlung der besten Parallelschaltung. Das – sehr kurze – Hauptprogramm umfasst dann nur noch die Eingabe des Sollwerts sowie die Aufrufe der drei genannten Funktionen.

Der Gesamtwiderstand einer Reihenschaltung wird folgendermaßen berechnet:

$$R_{\text{ges}} = R_1 + R_2 \tag{4.1}$$

Für eine Parallelschaltung von zwei Widerständen gilt:

$$R_{\text{ges}} = (R_1 \cdot R_2)/(R_1 + R_2) \tag{4.2}$$

Zusatzaufgabe

- Geben Sie eine Empfehlung aus, welche der drei Varianten (bester Einzelwiderstand, beste Reihen- oder beste Parallelschaltung) zur geringesten Abweichung vom Sollwert führt.
- Ihr Skript soll in einer Schleife solange immer weitere Berechnungen durchführen, bis ein negativer Sollwert eingegeben wird.
- Zeichnen Sie ein Struktogramm der von Ihnen programmierten Funktion zur Ermittlung der besten Reihenschaltung.

```
Konsole 1/A ×
Berechnung von Widerstandsschaltungen
=====================================

Sollwert in Ohm, negativer Wert zum Beenden: 1300

A. Bester Einzelwiderstand: 1500 Ohm.
Abweichung vom Sollwert: 200 Ohm (15.38 %).

B. Beste Reihenschaltung: 330 und 1000 Ohm.
Gesamtwiderstand: 1330 Ohm.
Abweichung vom Sollwert: 30 Ohm (2.31 %).

C. Beste Parallelschaltung: 1500 und 10000 Ohm.
Gesamtwiderstand: 1304.35 Ohm.
Abweichung vom Sollwert: 4.35 Ohm (0.33 %).

Empfehlung: Variante C

Sollwert in Ohm, negativer Wert zum Beenden: -1

In [2]:
```

5 Aufgaben zur Prüfungsvorbereitung

5.1 Mondrakete Saturn V

Schreiben Sie ein Skript, das die Geschwindigkeit der Mondrakete Saturn V während der Brenndauer der ersten Stufe in Zeitschritten der Größe $\Delta t = 0{,}05\,\mathrm{s}$ berechnet und auf dem Bildschirm ausgibt.

Die folgenden technischen Daten[3] sind gegeben:

- Startmasse inkl. Treibstoff und Nutzlast: $m = 2{,}85 \cdot 10^6\,\mathrm{kg}$
- Treibstoffverbrauch („Brennrate“): $R = 13{,}84 \cdot 10^3\,\mathrm{kg/s}$
- Schubkraft der ersten Raketenstufe: $F_{\mathrm{Schub}} = 34 \cdot 10^6\,\mathrm{N}$
- Fallbeschleunigung, wird als gleichbleibend angenommen: $g = 9{,}81\,\mathrm{m/s^2}$

Dabei macht der Treibstoff der ersten Raketenstufe 73 % der Startmasse aus.

[3] Vergl. Tipler/Mosca: Physik, 6. Auflage, Beispiel 8.19 auf Seite 307

Gehen Sie bei der Programmierung des Skripts wie folgt vor:

- Zu Beginn ($t = 0$) befindet sich die Rakete in Ruhe ($v = 0$).
- Wenn Masse $m(t)$ und Geschwindigkeit $v(t)$ zum Zeitpunkt t bekannt sind, können $m(t + \Delta t)$ und $v(t + \Delta t)$ zum Zeitpunkt $t + \Delta t$ berechnet werden:

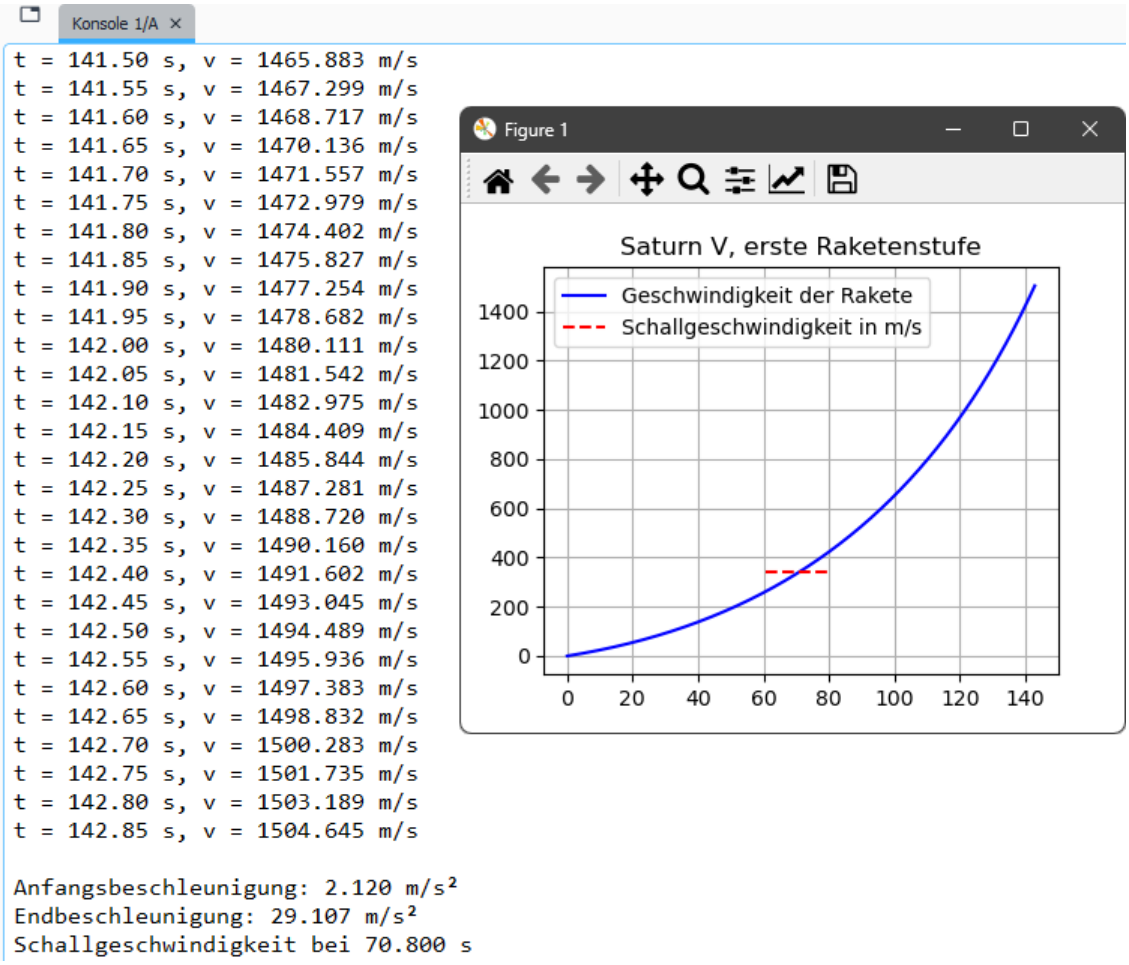
$$v(t + \Delta t) = v(t) + \Delta t \cdot a(t) \text{ mit } a(t) = F_{\text{Schub}}/m(t) - g \tag{5.1}$$

$$m(t + \Delta t) = m(t) - \Delta t \cdot R \tag{5.2}$$

- Nach jedem Zeitschritt $\Delta t = 0{,}05\,\text{s}$ werden die aktuellen Werte von t (zwei Nachkommastellen) und v (drei Nachkommastellen) tabellarisch ausgegeben.
- Die Berechnung endet, sobald der Treibstoff der ersten Stufe zu mehr als 95 % aufgebraucht ist.
- Unmittelbar vor dem Ende des Skripts werden die folgenden Werte mit drei Nachkommastellen ausgegeben: Die Anfangsbeschleunigung $a(t = 0)$, die Beschleunigung zum Berechnungsende sowie der Zeitpunkt t_{Schall}, zu dem die Rakete die Schallgeschwindigkeit $c = 340\,\text{m/s}$ erstmals überschreitet.

Zusatzaufgabe

Geben Sie – zusätzlich zur Tabelle mit den Berechnungsergebnissen – den zeitlichen Verlauf der Geschwindigkeit $v(t)$ in grafischer Form aus.

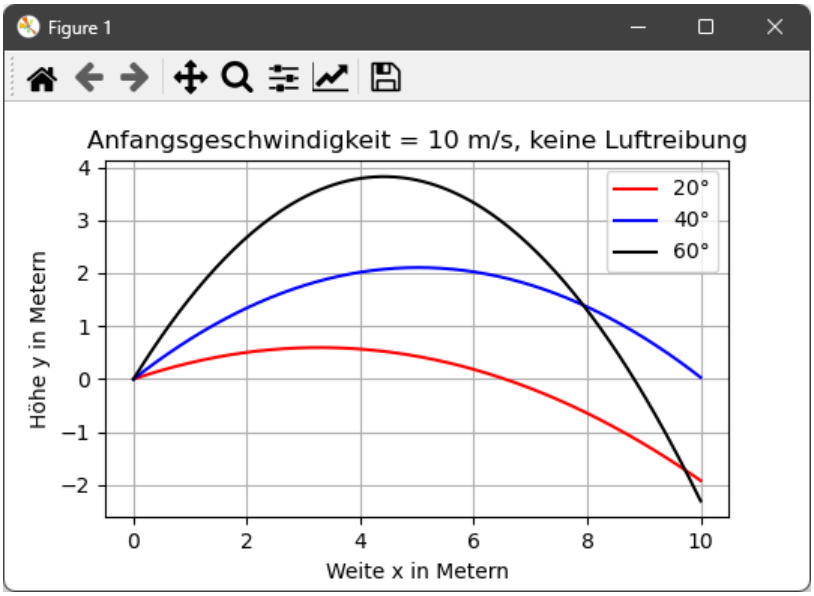


5.2 Wurfparabel

Eine Kugel wird unter Winkeln von $\alpha_0 = 20°$, $\alpha_0 = 40°$ sowie $\alpha_0 = 60°$ schräg nach oben geworfen. Die Anfangsgeschwindigkeit beträgt $v_0 = 10\,\mathrm{m/s}$, Luftreibung wird vernachlässigt. Stellen Sie den Verlauf der drei Flugbahnen grafisch dar:

$$y(x) = -\frac{g}{2} \cdot \left(\frac{x}{v_{x0}}\right)^2 + v_{y0} \cdot \left(\frac{x}{v_{x0}}\right) \tag{5.3}$$

mit $v_{x0} = v_0 \cdot \cos\alpha_0$, $v_{y0} = v_0 \cdot \sin\alpha_0$ und $g = 9{,}81\,\mathrm{m/s^2}$.



Zusatzaufgabe

- Fügen Sie einen Titel und eine Legende zur Abbildung hinzu.
- Beschriften Sie Koordinatenachsen („Weite x in Metern“, „Höhe y in Metern“).

5.3 Zeichenketten aus Textdatei lesen

Das folgende Skript gibt den Inhalt einer Textdatei auf dem Bildschirm aus.

```
# Textdatei zum Lesen öffnen
with open("data.txt", "r") as file:

    # Nächste Zeile aus der Textdatei lesen
    for zeile in file:

        # Zeile (Zeichenkette, String) ausgeben
        print(f"{zeile}", end="")
```

Ändern bzw. erweitern Sie das abgebildete Skript:

- Der Inhalt der Textdatei soll in Großbuchstaben ausgegeben werden.
- Die Anzahl der Zeilen in der Textdatei soll ausgegeben werden.
- Die Anzahl der Ziffern (0, 1, 2 ... 9) soll ebenfalls ausgegeben werden.

5.4 Funktion von zwei Veränderlichen

Schreiben Sie ein Skript, das die folgende Funktion von zwei Veränderlichen x und y berechnet und tabellarisch mit drei Nachkommastellen auf dem Bildschirm ausgibt:

$$f(x,y) = \frac{\mathrm{e}^{-(x^2+y^2)}}{\sqrt{1+x^2+y^2}} \tag{5.4}$$

Beachten Sie die folgenden Hinweise:

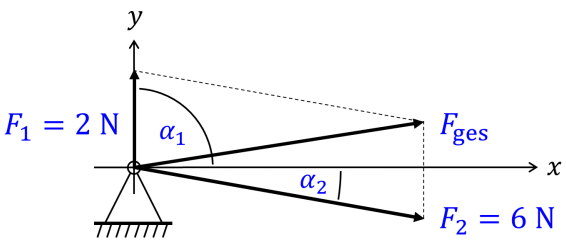
- Die y-Werte laufen von 0,0 bis 0,8 mit einer Schrittweite von 0,1 nach rechts.
- Das x-Werte laufen von 0,0 bis 1,4 mit einer Schrittweite von 0,2 nach unten.

```
Konsole 1/A ×
x|y    0.0   0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8
0.0 | 1.000 0.985 0.942 0.875 0.791 0.697 0.598 0.502 0.412
0.2 | 0.942 0.928 0.888 0.826 0.747 0.659 0.567 0.476 0.391
0.4 | 0.791 0.780 0.747 0.697 0.632 0.559 0.482 0.406 0.335
0.6 | 0.598 0.590 0.567 0.530 0.482 0.428 0.371 0.314 0.260
0.8 | 0.412 0.406 0.391 0.366 0.335 0.299 0.260 0.221 0.184
1.0 | 0.260 0.257 0.247 0.233 0.213 0.191 0.167 0.143 0.119
1.2 | 0.152 0.150 0.145 0.136 0.125 0.113 0.099 0.085 0.071
1.4 | 0.082 0.081 0.078 0.074 0.068 0.061 0.054 0.046 0.039
```

5.5 Addition von Kraftvektoren

Auf ein Festlager wirken mehrere Kräfte (ebenes Kräftesystem, alle Kräfte liegen in der xy-Ebene). Schreiben Sie ein Python-Skript, welches die Gesamtkraft berechnet, die auf das Festlager wirkt:

- Zunächst werden die Beträge und Winkel aller Kräfte eingegeben. Die Eingabe endet, wenn ein negativer Betrag (oder null) eingegeben wird.
- Die eingegebenen Kräfte und Winkel werden in Listen gespeichert.
- Nach dem Ende der Eingabe werden alle Kräfte und Winkel tabellarisch untereinander mit drei Nachkommastellen ausgegeben. Zu Beginn jeder Zeile wird eine laufende Nummer (1, 2, 3 ...) ausgegeben.
- Schließlich wird die Gesamtkraft berechnet. Betrag und Winkel der Gesamtkraft werden mit 3 Nachkommastellen ausgegeben. Falls gar keine Kräfte eingegeben wurden, erfolgt ein entsprechender Hinweis.



Die Abbildung zeigt ein Beispiel mit (nur) zwei Kräften: $F_1 = 2\,\mathrm{N}$ mit einem Winkel von $\alpha_1 = 90°$ sowie $F_2 = 6\,\mathrm{N}$ mit einem Winkel von $\alpha_2 = -10°$. Dadurch ergibt sich eine Gesamtkraft $F_{ges} = 5{,}986\,\mathrm{N}$ mit einem Winkel von $\alpha_{ges} = 9{,}210°$.

5.6 Kreiszahl Pi

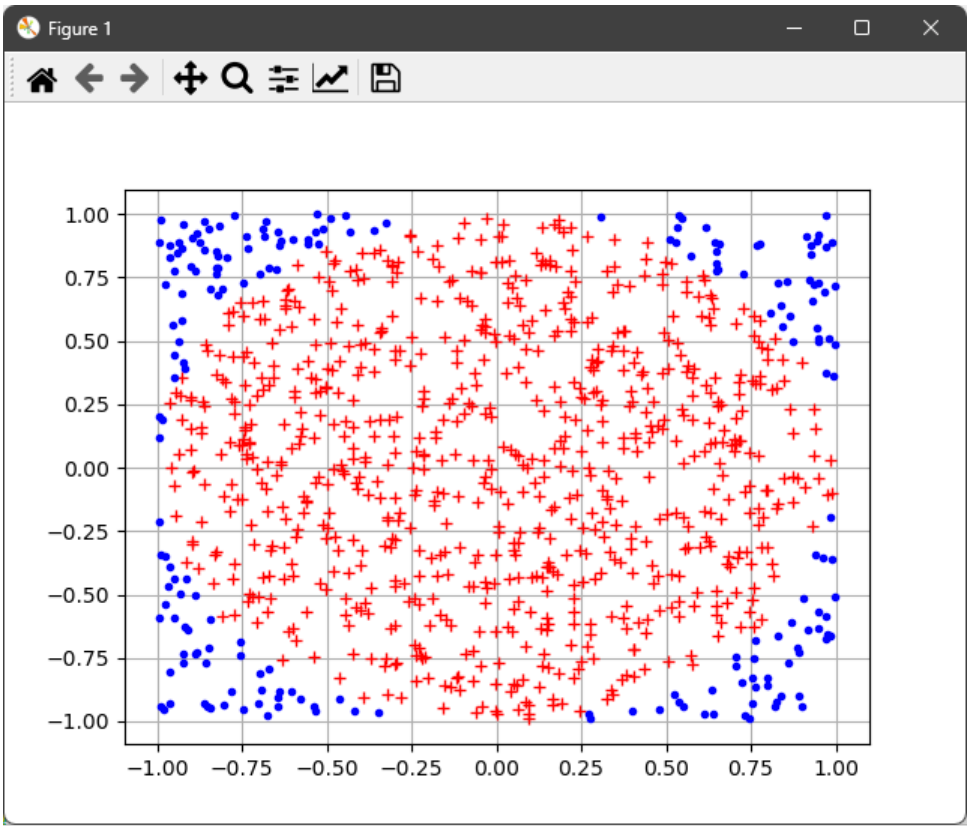
Schreiben Sie ein Skript zur näherungsweisen Berechnung der Kreiszahl Pi mithilfe eines „Monte-Carlo-Verfahrens“. Zu Beginn des Skripts wird über die Tastatur die gewünschte Anzahl n der Berechnungsschritte[4] eingegeben. Außerdem wird die Variable n_kreis auf 0 gesetzt.

In einer Schleife werden die folgenden Schritte n mal wiederholt:

- Es wird ein zufälliger x-Wert im Bereich von -1.0 bis 1.0 generiert (Kommazahl, beide Grenzen eingeschlossen).
- Es wird ein zufälliger y-Wert im Bereich von -1.0 bis 1.0 generiert (Kommazahl, beide Grenzen eingeschlossen).
- Falls der Punkt $P(x;y)$ auf dem Rand oder innerhalb des Einheitskreises[5] liegt, wird die Variable n_kreis inkrementiert.

Das Skript berechnet nun mit $\pi \approx 4 \cdot n_{\text{kreis}}/n$ einen Näherungswert für die Kreiszahl Pi und gibt diesen mit 10 Nachkommastellen auf dem Bildschirm aus.

Zusätzlich werden die n zufällig generierten Punkte $P(x;y)$ grafisch dargestellt: Alle Punkte auf dem Rand oder innerhalb des Einheitskreises werden als rote Pluszeichen (+) dargestellt, alle anderen als blaue Punkte. Denken Sie auch an die Ausgabe des Koordinatensystems (Gitterlinien, wie im Bildschirmfoto gezeigt).



[4] Größere n führen zu genaueren Ergebnissen. Typische Werte sind $n = 10000$ oder $n = 100000$.
[5] Der Einheitskreis ist ein Kreis mit dem Radius $r = 1$ um den Koordinatenursprung $M(0;0)$.

5.7 Lithium-Ionen-Akku

Ein Lithium-Ionen-Akku wird entladen. Die Akkuspannung wird in regelmäßigen Zeitabständen gemessen und in der Textdatei „messung.txt“ gespeichert. In jeder Zeile der Textdatei stehen ein Zeitpunkt (in Sekunden) sowie die zu diesem Zeitpunkt gemessene Akkuspannung (in Volt).

Das folgende Python-Skript gibt alle in der Textdatei gespeicherten Zeitpunkte und Akkuspannungen auf dem Bildschirm aus:

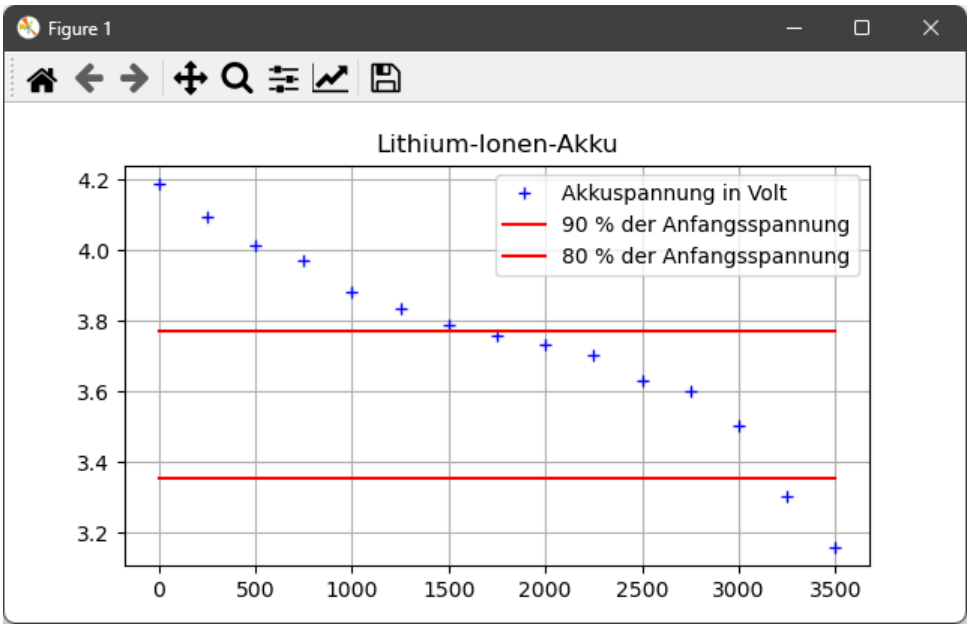
```
# Textdatei zum Lesen öffnen
with open("messung.txt", "r") as file:

    # Der Reihe nach alle Zeilen durchlaufen
    for line in file:
        values = line.split()
        t = float(values[0])
        u = float(values[1])

        # In t steht der Zeitpunkt, in u steht die
        # Akkuspannung aus der aktuellen Zeile
        print(f"{t} {u}")
```

Erweitern Sie das Skript, sodass die Zeitpunkte und Akkuspannungen nun in grafischer Form dargestellt werden:

- Die einzelnen Messpunkte werden durch blaue Pluszeichen (+) angezeigt.
- Zeichnen Sie zwei horizontale rote Linien, die obere Linie bei 90 %, die untere Linie bei 80 % der Anfangsspannung.
- Denken Sie auch an die Ausgabe des Koordinatengitters und des Titels.

Geben Sie zusätzlich zur grafischen Darstellung die folgenden beiden Zeitpunkte in Textform auf dem Bildschirm aus (in Sekunden, mit einer Nachkommastelle):

- Wann sinkt die Spannung erstmals unter 90 % der Anfangsspannung?
- Wann sinkt die Spannung erstmals unter 80 % der Anfangsspannung?



A Kurzfragen

Operatoren

Wie lauten die Ergebnisse der folgenden Berechnungen?

```
a = 1999
b = 4

c = a % b
d = a // b
e = (c == d)
f = b ** b

print(f"c = {c}")
print(f"d = {d}")
print(f"e = {e}")
print(f"f = {f}")
```

```
x1 = 2 ** 2 ** 3
x2 = 2 ** (2 ** 3)
x3 = (2 ** 2) ** 3

print(f"x1 = {x1}")
print(f"x2 = {x2}")
print(f"x3 = {x3}")
```

Verzweigungen

In diesem Skript zur Berechnung des Body-Mass-Index (BMI, vergl. Aufgabe 2.3) hat sich ein Fehler versteckt. Finden Sie ihn?

```
m = float(input("Masse in kg: "))
l = float(input("Länge in m: "))

bmi = m / l ** 2
print(f"BMI = {bmi}")

if bmi < 30:
    print("Leichtes Übergewicht")
elif bmi < 25:
    print("Normalgewicht")
elif bmi < 20:
    print("Untergewicht")
else:
    print("Übergewicht")
```

Welche Ausgabe erfolgt nach der Eingabe von -1, von 0 und von 1?

```
zahl = int(input("1 oder 0 eingeben: "))

if zahl == 0:
    print("Null")
if zahl == 1:
    print("Eins")
else:
    print("Fehler")
```

Schleifen

Wie lauten die Ausgaben der folgenden Skripte?

```
i = 0
for a in range(5):
    for b in range(10):
        for c in range(2):
            i += 1

print(f"i = {i}")
```

```
i = 0
for x in range(1, 11):
    for y in range(2, 11, 2):
        for z in range(1, 8):
            if z == 6:
                break
            i += 1

print(f"i = {i}")
```

```
i = 0
x = 0
while x < 10:
    y = 0
    while y < 10:
        y += 1
        if y > 5:
            continue
        i += 1
    x += 1

print(f"i = {i}")
```

Funktionen

Wie lauten die Ausgaben der folgenden Skripte?

```
from cmath import sqrt, exp, pi

z1 = sqrt(-1)
z2 = 10 * exp(1j * pi / 2)

print(f"z1 = {z1:.1f}")
print(f"z2 = {z2:.1f}")
```

```
from math import sin, cos, pi


def my_fun(a, b):
    a *= pi
    b *= pi
    return sin(a), cos(b)


a, b = my_fun(1, 2)
print(f"a = {a:.1f}, b = {b:.1f}")
```

B Lösungsvorschläge

Kapitel 1

```
# Aufgabe 1.1
# Umrechnung von kW nach PS

xx = input("Leistung in Kilowatt: ")
kw = float(xx)

ps = kw * 1.35962
print(f"{kw} kW entspricht {ps} PS.")
```

```
# Aufgabe 1.2
# Mitternachtsformel

from math import *

a = float(input("a: "))
b = float(input("b: "))
c = float(input("c: "))
print("")

diskrim = b**2 - 4 * a * c

if diskrim < 0:
    print("Keine reellen Nullstellen!")
else:
    x1 = (-b - sqrt(diskrim)) / (2 * a)
    x2 = (-b + sqrt(diskrim)) / (2 * a)
    print("Nullstellen:")
    print(f"  x1 = {x1}")
    print(f"  x2 = {x2}")
```

Kapitel 2

```
# Aufgabe 2.1
# Stromrechnung

kWh = float(input("Verbrauch in kWh: "))

if kWh > 5000:
    preis = 2500 * 0.40 + 2500 * 0.35 + (kWh - 5000) * 0.30
elif kWh > 2500:
    preis = 2500 * 0.40 + (kWh - 2500) * 0.35
else:
    preis = kWh * 0.40

print(f"Rechnungsbetrag: {preis:.2f} EUR")
```

```python
# Aufgabe 2.2, Variante a
# Maximum von drei Werten

a = int(input("a eingeben: "))
b = int(input("b eingeben: "))
c = int(input("c eingeben: "))

maxi = a
if b > maxi:
    maxi = b
if c > maxi:
    maxi = c

print(f"Das Maximum ist: {maxi}")
```

```python
# Aufgabe 2.2, Variante b
# Maximum von drei Werten

a = int(input("a eingeben: "))
b = int(input("b eingeben: "))
c = int(input("c eingeben: "))

if a > b:
    if a > c:
        maxi = a
    else:
        maxi = c
else:
    if b > c:
        maxi = b
    else:
        maxi = c

print(f"Das Maximum ist: {maxi}")
```

```python
# Aufgabe 2.3
# Body-Mass-Index

m = float(input("Körpergewicht in kg: "))
l = float(input("Größe in m: "))
bmi = m / l ** 2

if bmi < 20:
    print(f"BMI = {bmi:.1f}, Untergewicht")
else:
    if bmi < 25:
        print(f"BMI = {bmi:.1f}, Normalgewicht")
    else:
        if bmi < 30:
            print(f"BMI = {bmi:.1f}, leichtes Übergewicht")
        else:
            print(f"BMI = {bmi:.1f}, Übergewicht")
```

```python
# Aufgabe 2.4, Variante a
# ICAO-Standardatmosphäre

hx = float(input("Bitte Höhe in m eingeben: "))

if hx < 0:
    print("Fehler: Höhe zu gering")
elif hx > 47000:
    print("Fehler: Höhe zu groß")
else:
    if hx < 11000:
        h1 = 0.0
        h2 = 11000.0
        T1 = 15.0
        T2 = -56.5
    elif hx < 20000:
        h1 = 11000.0
        h2 = 20000.0
        T1 = -56.5
        T2 = -56.5
    elif hx < 32000:
        h1 = 20000.0
        h2 = 32000.0
        T1 = -56.5
        T2 = -44.5
    else:
        h1 = 32000.0
        h2 = 47000.0
        T1 = -44.5
        T2 = -2.5

    # TODO: Temperatur in der Höhe hx berechnen und ausgeben
```

```python
# Aufgabe 2.4, Variante b, mit Numpy
# ICAO-Standardatmosphäre

from numpy import interp

hh = [0, 11000, 20000, 32000, 47000]
tt = [15.0, -56.5, -56.5, -44.5, -2.5]
hx = float(input("Bitte Höhe in m eingeben: "))

if hx < hh[0]:
    print("Fehler: Höhe zu gering")
elif hx > hh[-1]:
    print("Fehler: Höhe zu groß")
else:
    Tx = interp(hx, hh, tt)
    print(f"Die Temperatur beträgt {Tx} °C")
```

Kapitel 3

```
# Aufgabe 3.1
# Fakultät

n = 0
while n < 1 or n > 50:
    n = int(input("n eingeben: "))

fakultaet = 1
while n > 1:
    fakultaet *= n
    n -= 1

print(f"Ergebnis: {fakultaet}")
```

```
# Aufgabe 3.2
# Einheitsmatrix

n = 1
while n != 0:
    n = int(input("Dimension (1 ... 10): "))

    # TODO: Eingabe wiederholen, falls ungültig

    for zeile in range(n):
        for spalte in range(n):
            if zeile == spalte:
                print("1 ", end="")
            else:
                print("0 ", end="")
        print("")

print("Ende des Programms.")
```

```
# Aufgabe 3.3, Variante a, mit Liste
# Würfelspiel

from random import randint

anz = [0, 0, 0, 0, 0, 0]
for i in range(1000):
    x = randint(1, 6)
    anz[x - 1] += 1

for i in range(6):
    print(f"Anzahl {i+1}: {anz[i]}")
```

```python
# Aufgabe 3.3, Variante b
# Würfelspiel

from random import randint

anz1 = 0
anz2 = 0
anz3 = 0
anz4 = 0
anz5 = 0
anz6 = 0

for i in range(1000):
    x = randint(1, 6)
    if x == 1:
        anz1 = anz1 + 1
    elif x == 2:
        anz2 = anz2 + 1
    elif x == 3:
        anz3 = anz3 + 1
    elif x == 4:
        anz4 = anz4 + 1
    elif x == 5:
        anz5 = anz5 + 1
    else:
        anz6 = anz6 + 1

print(f"Anzahl 1: {anz1}")
print(f"Anzahl 2: {anz2}")
print(f"Anzahl 3: {anz3}")
print(f"Anzahl 4: {anz4}")
print(f"Anzahl 5: {anz5}")
print(f"Anzahl 6: {anz6}")
```

```python
# Aufgabe 3.4
# Funktionswerte

x = float(input("Start des x-Intervalls: "))
x1 = float(input("Ende des x-Intervalls: "))

while x <= x1:
    y = 2 * x ** 2 - x - 2
    print(f"{x:8.3f} {y:8.3f}")
    x = x + 0.25
```

```python
# Aufgabe 3.5
# Funktionsgraphen zeichnen

from matplotlib.pyplot import *

x = float(input("Start des x-Intervalls: "))
x1 = float(input("Ende des x-Intervalls: "))

xx = []   # Liste mit x-Werten
yy = []   # Liste mit y-Werten

while x <= x1:
    y = 2 * x ** 2 - x - 2
    xx.append(x)
    yy.append(y)
    print(f"{x:8.3f} {y:8.3f}")
    x = x + 0.125

plot(xx, yy, "r-")
grid(True)
show()
```

```python
# Aufgabe 3.6, Variante a
# Spannung, Strom und Leistung

from matplotlib.pyplot import *
from math import *  # pi, sin, cos

tt = []
uu = []
ii = []
pp = []

t = 0
while t < 0.05:
    u = 10 * sin(2 * pi * 50 * t)
    i = pi * cos(2 * pi * 50 * t)
    p = u * i
    tt.append(t)
    uu.append(u)
    ii.append(i)
    pp.append(p)
    t += 0.0001

plot(tt, uu, "b-")
plot(tt, ii, "r-")
plot(tt, pp, "g--")
grid(True)
show()
```

```python
# Aufgabe 3.6, Variante b, mit Numpy
# Spannung, Strom und Leistung

import matplotlib.pyplot as plt
from numpy import linspace, pi, sin, cos

tt = linspace(0, 0.05, 1000)
uu = 10 * sin(2 * pi * 50 * tt)
ii = pi * cos(2 * pi * 50 * tt)
pp = uu * ii

plt.plot(tt, uu, "b-")
plt.plot(tt, ii, "r-")
plt.plot(tt, pp, "g--")
plt.grid(True)
plt.show()
```

```python
# Aufgabe 3.7
# Fibonacci-Folge

n1 = 0
n2 = 1

for i in range(30):
    print(f"Nr. {i+1:2d}: {n1:6d}")
    n3 = n1 + n2
    n1 = n2
    n2 = n3
```

Kapitel 4

```python
# Aufgabe 4.1
# Quersumme als Funktion

def qsum(zahl):
    if zahl < 1:
        print("Funktion qsum: Bitte positive Zahl übergeben!")
        return 0

    summe = 0
    while zahl > 0:
        letzte_ziffer = zahl % 10
        summe = summe + letzte_ziffer
        zahl = zahl // 10

    return summe


z = int(input("Bitte ganze, positive Zahl eingeben: "))
q = qsum(z)

if q > 0:
    print(f"Die Quersumme von {z} beträgt {q}.")
```

```python
# Aufgabe 4.2
# Primzahlen

def primzahl(zahl):
    if zahl < 2:
        return False

    for t in range(2, zahl):
        if zahl % t == 0:
            return False

    return True


a = int(input("Primzahlen ermitteln von: "))
b = int(input("... bis: "))

for n in range(a, b + 1):
    if primzahl(n):
        print(f"{n:6d}")
```

```python
# Aufgabe 4.3
# Mitternachtsformel als Funktion

from math import sqrt


def qsolve(a, b, c):
    diskrim = b**2 - 4 * a * c

    if diskrim < 0:
        n = 0
        x1 = x2 = 0
    elif diskrim == 0:
        n = 1
        x1 = x2 = -b / (2 * a)
    else:
        n = 2
        x1 = (-b - sqrt(diskrim)) / (2 * a)
        x2 = (-b + sqrt(diskrim)) / (2 * a)

    return n, x1, x2


a = float(input("a: "))
b = float(input("b: "))
c = float(input("c: "))

n, x1, x2 = qsolve(a, b, c)

if n == 0:
    print("Keine reellen Nullstellen!")
elif n == 1:
    print(f"Doppelte Nullstelle bei {x1:.2f}")
else:
    print(f"Nullstellen bei {x1:.2f} und {x2:.2f}")
```

```python
# Aufgabe 4.4
# Zerlegung eines Kraftvektors

from math import sin, cos, pi


def kraft(m, alpha):
    g = 9.81
    fg = m * g

    fn = cos(alpha * pi / 180) * fg
    fh = sin(alpha * pi / 180) * fg
    return fn, fh


m = float(input("Masse in kg: "))
alpha = float(input("Winkel in Grad: "))

fn, fh = kraft(m, alpha)

print(f"Senkrecht zur Oberfläche: {fn:.2f} N")
print(f"Hangabtriebskraft: {fh:.2f} N")
```

```python
# Aufgabe 4.5
# Euklidischer Algorithmus

from random import randint


def ggT(a, b):
    while b != 0:
        tmp = a % b
        a = b
        b = tmp
    return a


aa = []
bb = []
cc = []

for i in range(100):
    zuf_a = randint(1, 100)
    zuf_b = randint(1, 100)

    aa.append(zuf_a)
    bb.append(zuf_b)
    cc.append(ggT(zuf_a, zuf_b))

print("Nr.   aa   bb   cc")
for i in range(100):
    print(f"{i+1:3d}", end="")
    print(f"{aa[i]:5d}", end="")
    print(f"{bb[i]:5d}", end="")
    print(f"{cc[i]:5d}")
```

```python
# Aufgabe 4.6
# Numerische Integration

from random import uniform
from math import sin

z1 = 0
z2 = 0
a = float(input("a: "))
b = float(input("b: "))


def f(x):
    return sin(x) ** 2


for i in range(10000):
    x = uniform(a, b)
    y = uniform(0, 1)

    if f(x) < y:
        z1 += 1
    else:
        z2 += 1

A1 = (b - a) / (z1 + z2) * z2
print(f"A1 = {A1:.3f}")
```

```python
# Aufgabe 4.7
# Reihen- und Parallelschaltung von Widerständen

werte = [10, 47, 100, 150, 220, 330, 470, 1000, 1500, 2200,
         4700, 10000, 22000, 27000, 33000, 47000, 100000,
         220000, 470000, 1000000]


# Besten Einzelwiderstand ermitteln
def widerstand_einzel(soll):
    delta_min = 1e20
    for r in werte:
        delta = abs(soll - r)
        if delta < delta_min:
            delta_min = delta
            r_optimal = r

    proz = 100 * delta_min / soll
    print(f"A. Bester Einzelwiderstand: {r_optimal} Ohm.")
    print(f"Abweichung vom Sollwert: {delta_min} Ohm ({proz:.2f} %).")


# Beste Reihenschaltung ermitteln
def widerstand_reihe(soll):
    delta_min = 1e20
    for r1 in werte:
        for r2 in werte:
            r_gesamt = r1 + r2
            delta = abs(soll - r_gesamt)
            if delta < delta_min:
                delta_min = delta
                r1_optimal = r1
                r2_optimal = r2

    proz = 100 * delta_min / soll
    r_ges = r1_optimal + r2_optimal
    print(f"B. Reihenschaltung: {r1_optimal} und {r2_optimal} Ohm.")
    print(f"Gesamtwiderstand: {r_ges} Ohm.")
    print(f"Abweichung vom Sollwert: {delta_min} Ohm ({proz:.2f} %).")


# Beste Parallelschaltung ermitteln
def widerstand_parallel(soll):
    delta_min = 1e20
    for r1 in werte:
        for r2 in werte:
            r_gesamt = r1 * r2 / (r1 + r2)
            delta = abs(soll - r_gesamt)
            if delta < delta_min:
                delta_min = delta
                r1_optimal = r1
                r2_optimal = r2

    proz = 100 * delta_min / soll
    r_ges = r1_optimal * r2_optimal / (r1_optimal + r2_optimal)
    print(f"C. Parallelschaltung: {r1_optimal} und {r2_optimal} Ohm.")
    print(f"Gesamtwiderstand: {r_ges:.2f} Ohm.")
    print(f"Abweichung vom Sollwert: {delta_min} Ohm ({proz:.2f} %).")


# Hauptprogramm
print("Berechnung von Widerstandsschaltungen")
print("=====================================")
soll = int(input("Sollwert in Ohm: "))
widerstand_einzel(soll)
widerstand_reihe(soll)
widerstand_parallel(soll)
```

Kapitel 5

```
# Aufgabe 5.1
# Mondrakete Saturn V

m = 2850000
r = 13840
f_schub = 34000000
g = 9.81
delta_t = 0.05
t = 0
v = 0

m_treibstoff = 0.73 * m
m_ende = m - 0.95 * m_treibstoff

while m > m_ende:
    a = f_schub / m - g
    m = m - delta_t * r
    v = v + delta_t * a
    t = t + delta_t
    print(f"t ={t:7.2f} s, v ={v:9.3f} m/s")

# TODO: Anfangs- und Endbeschleunigung ausgeben;
# wann wird die Schallgeschwindigkeit erreicht?
```

```
# Aufgabe 5.2, Variante a
# Wurfparabel

import matplotlib.pyplot as plt
from math import sin, cos, radians


def wurfparabel(alpha0_grad, farbe):
    v0 = 10
    g = 9.81
    v_x0 = v0 * cos(radians(alpha0_grad))
    v_y0 = v0 * sin(radians(alpha0_grad))

    xx = []
    yy = []

    x = 0
    while x <= 10:
        y = -g / 2 * (x / v_x0) ** 2 + v_y0 * (x / v_x0)
        xx.append(x)
        yy.append(y)
        x = x + 0.01

    plt.plot(xx, yy, farbe)


wurfparabel(20, "r-")
wurfparabel(40, "b-")
wurfparabel(60, "k-")

# TODO: Titel, Legende, Achsenbeschriftung ausgeben

plt.grid(True)
plt.show()
```

```python
# Aufgabe 5.2, Variante b, mit NumPy
# Wurfparabel

import matplotlib.pyplot as plt
from numpy import sin, cos, radians, linspace


def wurfparabel(alpha0_grad, farbe):
    v0 = 10
    g = 9.81
    v_x0 = v0 * cos(radians(alpha0_grad))
    v_y0 = v0 * sin(radians(alpha0_grad))

    xx = linspace(0, 10, 250)
    yy = -g / 2 * (xx / v_x0) ** 2 + v_y0 * (xx / v_x0)
    plt.plot(xx, yy, farbe)


wurfparabel(20, "r-")
wurfparabel(40, "b-")
wurfparabel(60, "k-")

# TODO: Titel, Legende, Achsenbeschriftung ausgeben

plt.grid(True)
plt.show()
```

```python
# Aufgabe 5.3
# Zeichenketten aus Textdatei lesen

anz_zeilen = 0
anz_ziffern = 0

with open("data.txt", "r") as file:
    for zeile in file:
        anz_zeilen += 1
        for z in zeile:
            if z in "0123456789":
                anz_ziffern += 1
            print(z.upper(), end="")

print("")
print("")
print(f"--> Anzahl Zeilen: {anz_zeilen}")
print(f"--> Anzahl Ziffern: {anz_ziffern}")
```

```python
# Aufgabe 5.4
# Funktion von zwei Veränderlichen

from math import exp, sqrt


def f(x, y):
    zaehler = exp(-x**2 - y**2)
    nenner = sqrt(1 + x**2 + y**2)
    return zaehler / nenner


print("x|y    0.0   0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8")

for x_tmp in range(0, 15, 2):
    x = x_tmp / 10
    print(f"{x:3.1f} |", end="")

    for y_tmp in range(0, 9, 1):
        y = y_tmp / 10
        z = f(x, y)
        print(f"{z:6.3f}", end="")

    print("")
```

```python
# Aufgabe 5.5
# Addition von Kraftvektoren

kraefte = []
winkel = []
anzahl = 0

while True:
    k = float(input("Kraft  / N:    "))
    if k <= 0:
        break

    w = float(input("Winkel / Grad: "))
    kraefte.append(k)
    winkel.append(w)
    anzahl += 1

if anzahl == 0:
    print("Es wurden keine Kräfte eingegeben.")
else:
    print("")
    for idx in range(anzahl):
        k = kraefte[idx]
        w = winkel[idx]
        print(f"{idx+1:3d}: {k:8.3f} N, {w:8.3f} Grad")

# TODO: Gesamtkraft, Gesamtwinkel berechnen und ausgeben
```

```python
# Aufgabe 5.6
# Kreiszahl Pi

from random import uniform
import matplotlib.pyplot as plt

n_kreis = 0
n = int(input("Anzahl Punkte: "))

for i in range(n):
    x = uniform(-1, 1)
    y = uniform(-1, 1)

    if x**2 + y**2 <= 1:
        n_kreis += 1

# TODO: Punkte P(x;y) grafisch darstellen

approx = 4 * n_kreis / n
print(f"Näherungswert für Pi: {approx:.10f}")
```

```python
# Aufgabe 5.7
# Lithium-Ionen-Akku

import matplotlib.pyplot as plt

tt = []
uu = []

with open("messung.txt", "r") as file:
    for line in file:
        values = line.split()
        t = float(values[0])
        u = float(values[1])

        tt.append(t)
        uu.append(u)

# TODO: Zeitpunkte ausgeben, wann die Akkuspannung erstmals
# unter 90 % bzw. unter 80 % der Anfangsspannung sinkt

t_anf = tt[0]
t_end = tt[-1]

u_90 = uu[0] * 0.9
u_80 = uu[0] * 0.8

plt.plot(tt, uu, "b+")
plt.plot([t_anf, t_end], [u_90, u_90], "r-")
plt.plot([t_anf, t_end], [u_80, u_80], "r-")

plt.title("Lithium-Ionen-Akku")
plt.grid(True)
plt.show()
```