

I Rechner

1. Lies mit der `input()` Funktion eine Rechnung ein. Dazu soll dreimal `input()` aufgerufen werden. Das erste Mal für die erste Zahl, das zweite Mal für `+` oder `-` und das dritte Mal für die dritte Zahl.

Gib den Input mit der `print()` Funktion als Rechnung aus. Es soll also in der Art `4+9` ausgegeben werden.

```
1 # Eingabe der ersten Zahl
2 zahl1 = input("Gib die erste Zahl ein: ")
3
4 # Eingabe des Operators (+ oder -)
5 operator = input("Gib + oder - ein: ")
6
7 # Eingabe der zweiten Zahl
8 zahl2 = input("Gib die zweite Zahl ein: ")
9
10 # Ausgabe der Rechnung
11 print(f"{zahl1}{operator}{zahl2}")
12
```

Rechner1.py

2. Mit einer `if`-Verzweigung soll nun auch noch das Resultat berechnet und schön ausgegeben werden. Es soll also eine Ausgabe wie z.B. `4+9=13` stehen. Die `if`-Verzweigung benötigen wir, damit wird zwischen `+` und `-` unterscheiden können.

```
1 # Eingabe der ersten Zahl
2 zahl1 = float(input("Gib die erste Zahl ein: ")) # Beachte die Umwandlung in eine Gleitkommazahl
3
4 # Eingabe des Operators (+ oder -)
5 operator = input("Gib + oder - ein: ")
6
7 # Eingabe der zweiten Zahl
8 zahl2 = float(input("Gib die zweite Zahl ein: ")) # Beachte die Umwandlung in eine Gleitkommazahl
9
10 # Initialisierung der Variable für das Ergebnis
11 ergebnis = 0
12
13 # Überprüfung des Operators und Berechnung des Ergebnisses
14 if operator == "+":
15     ergebnis = zahl1 + zahl2
16 elif operator == "-":
17     ergebnis = zahl1 - zahl2
18 else:
19     print("Ungültiger Operator. Bitte nur + oder - eingeben.")
20     exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
21
22 # Ausgabe der Rechnung und des Ergebnisses
23 print(f"{zahl1} {operator} {zahl2} = {ergebnis}")
24
```

Rechner2.py

3. Erweitere die `if`-Verzweigung für `*` und `/` also Mal und Durch-Rechnen.

```
1  # Eingabe der ersten Zahl
2  zahl1 = float(input("Gib die erste Zahl ein: "))
3
4  # Eingabe des Operators (+, -, *, /)
5  operator = input("Gib +, -, *, oder / ein: ")
6
7  # Eingabe der zweiten Zahl
8  zahl2 = float(input("Gib die zweite Zahl ein: "))
9
10 # Initialisierung der Variable für das Ergebnis
11 ergebnis = 0
12
13 # Überprüfung des Operators und Berechnung des Ergebnisses
14 if operator == "+":
15     ergebnis = zahl1 + zahl2
16 elif operator == "-":
17     ergebnis = zahl1 - zahl2
18 elif operator == "*":
19     ergebnis = zahl1 * zahl2
20 elif operator == "/":
21     # Überprüfung, ob der Divisor nicht 0 ist
22     if zahl2 != 0:
23         ergebnis = zahl1 / zahl2
24     else:
25         print("Division durch 0 ist nicht erlaubt.")
26         exit() # Programm beenden, da eine Division durch 0 versucht wurde
27 else:
28     print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
29     exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
30
31 # Ausgabe der Rechnung und des Ergebnisses
32 print(f"{zahl1} {operator} {zahl2} = {ergebnis}")
33
```

Rechner3.py

4. Die if -Verzweigung soll in einer eigenen Funktion mit dem Name berechne geschrieben werden. Die Funktion braucht 3 Argumente, nämlich zahl1,zahl2, operation. Die Rückgabe der Funktion ist natürlich das Resultat der Rechnung.
Rufe nun die Funktion berechne auf damit, wie oben, die gesamte Rechnung mit Resultat ausgegeben werden kann.

```
1 def berechne(zahl1, zahl2, operator):
2     """
3     Die Funktion berechne führt eine Rechnung basierend auf den gegebenen Zahlen und dem Operator durch.
4     """
5     if operator == "+":
6         return zahl1 + zahl2
7     elif operator == "-":
8         return zahl1 - zahl2
9     elif operator == "*":
10        return zahl1 * zahl2
11    elif operator == "/":
12        if zahl2 != 0:
13            return zahl1 / zahl2
14        else:
15            print("Division durch 0 ist nicht erlaubt.")
16            exit() # Programm beenden, da eine Division durch 0 versucht wurde
17    else:
18        print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
19        exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
20
21    # Eingabe der ersten Zahl
22    zahl1 = float(input("Gib die erste Zahl ein: "))
23
24    # Eingabe des Operators (+, -, *, /)
25    operator = input("Gib +, -, *, oder / ein: ")
26
27    # Eingabe der zweiten Zahl
28    zahl2 = float(input("Gib die zweite Zahl ein: "))
29
30    # Aufruf der Funktion berechne und Ausgabe des Ergebnisses
31    ergebnis = berechne(zahl1, zahl2, operator)
32    print(f"{zahl1} {operator} {zahl2} = {ergebnis}")
33
```

5. Nun möchten wir die Eingabe einfacher haben. Die Rechnung soll auf einmal eingegeben werden können. Also, rufen wir `input()` nur noch einmal auf anstatt dreimal. Weil nun die Rechnung in einem String ist, müssen wir die beiden Zahlen und das `+, -, *, /` aus dem String lesen. Um alles einfacher zu machen muss die Rechnung mit Leerschlägen eingegeben werden. Also die zwei Zahlen und das Rechnungszeichen müssen mit einem Leerschlag getrennt sein - zum Beispiel `4 + 9` und nicht `4+9`. Jetzt kann die Methode `split()` auf dem String verwendet werden, um die Zahlen und das Zeichen zu trennen. Der Rest soll wie bei 3) funktionieren.

```
1 def berechne(zahl1, zahl2, operator):
2     """
3     Die Funktion berechne führt eine Rechnung basierend auf den gegebenen Zahlen und dem Operator durch.
4     """
5     if operator == "+":
6         return zahl1 + zahl2
7     elif operator == "-":
8         return zahl1 - zahl2
9     elif operator == "*":
10        return zahl1 * zahl2
11    elif operator == "/":
12        if zahl2 != 0:
13            return zahl1 / zahl2
14        else:
15            print("Division durch 0 ist nicht erlaubt.")
16            exit() # Programm beenden, da eine Division durch 0 versucht wurde
17    else:
18        print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
19        exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
20
21 # Eingabe der Rechnung als String
22 eingabe = input("Gib die Rechnung ein (z.B. '4 + 9'): ")
23
24 # Trenne die Zahlen und den Operator mit split()
25 teile = eingabe.split()
26
27 # Überprüfe, ob die Eingabe gültig ist
28 if len(teile) != 3:
29     print("Ungültige Eingabe. Bitte die Rechnung mit Leerzeichen trennen (z.B. '4 + 9').")
30     exit()
31
32 # Extrahiere die Zahlen und den Operator
33 zahl1 = float(teile[0])
34 operator = teile[1]
35 zahl2 = float(teile[2])
36
37 # Aufruf der Funktion berechne und Ausgabe der gesamten Rechnung mit Ergebnis
38 ergebnis = berechne(zahl1, zahl2, operator)
39 print(f"{eingabe} = {ergebnis}")
40
```

Rechner5.py

II Konto

1. Schreibe eine Klasse Konto, mit den Attributen name und kontostand. Die Klasse hat zwei Methoden einzahlen(betrag) und auszahlen(betrag). Der Name wird über den Konstruktor angegeben (bsp: Konto("Donald Duck").) Das Konto ist zu am Anfang leer, das heisst der kontostand wird im Konstruktor auf 0 gesetzt.

```
1 class Konto:
2     def __init__(self, name):
3         self.name = name
4         self.kontostand = 0
5
6     def einzahlen(self, betrag):
7         if betrag > 0:
8             self.kontostand += betrag
9             print(f"{betrag} CHF wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} CHF")
10        else:
11            print("Ungültiger Betrag für Einzahlung.")
12
13    def auszahlen(self, betrag):
14        if 0 < betrag <= self.kontostand:
15            self.kontostand -= betrag
16            print(f"{betrag} CHF wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} CHF")
17        else:
18            print("Ungültiger Betrag für Auszahlung.")
19
20    # Beispiel-Nutzung:
21    donalds_konto = Konto("Donald Duck")
22    donalds_konto.einzahlen(100)
23    donalds_konto.auszahlen(30)
24
```

Konto1.py

2. Programmiere die Methoden um Geld ein- und auszuzahlen. Achte darauf, dass das Konto nicht überzogen werden kann. Also der Kontostand darf nicht unter 0 fallen. Das heisst es muss in der Methode auszahlen(betrag) mit **if** geprüft werden, ob genügend Geld da ist. Wenn der Kontostand zu tief ist, soll das mit **print** ausgegeben werden und natürlich den Kontostand nicht verändern.

```
1 class Konto:
2     def __init__(self, name):
3         self.name = name
4         self.kontostand = 0
5
6     def einzahlen(self, betrag):
7         if betrag > 0:
8             self.kontostand += betrag
9             print(f"{betrag} EUR wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} EUR")
10        else:
11            print("Ungültiger Betrag für Einzahlung.")
12
13    def auszahlen(self, betrag):
14        if betrag > 0 and betrag <= self.kontostand:
15            self.kontostand -= betrag
16            print(f"{betrag} EUR wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} EUR")
17        else:
18            print(f"Ungültiger Betrag für Auszahlung oder nicht ausreichend Geld auf dem Konto von {self.name}. Kontostand bleibt unverändert: {self.kontostand} EUR")
19
20    # Beispiel-Nutzung:
21    donalds_konto = Konto("Donald Duck")
22    donalds_konto.einzahlen(100)
23    donalds_konto.auszahlen(30)
24    donalds_konto.auszahlen(80) # Versuch, mehr Geld abzuheben als auf dem Konto vorhanden
25
```

Konto2.py

3. Teste ob alles wie beschrieben funktioniert, indem ein Konto erstellt wird und eine paar Ein- und Auszahlungen gemacht werden. Gibt jedes mal den Kontostand mit **print** aus.

```
1 class Konto:
2     def __init__(self, name):
3         self.name = name
4         self.kontostand = 0
5
6     def einzahlen(self, betrag):
7         if betrag > 0:
8             self.kontostand += betrag
9             print(f"{betrag} EUR wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} EUR")
10        else:
11            print("Ungültiger Betrag für Einzahlung.")
12
13    def auszahlen(self, betrag):
14        if betrag > 0 and betrag <= self.kontostand:
15            self.kontostand -= betrag
16            print(f"{betrag} EUR wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} EUR")
17        else:
18            print(f"Ungültiger Betrag für Auszahlung oder nicht ausreichend Geld auf dem Konto von {self.name}. Kontostand bleibt unverändert: {self.kontostand} EUR")
19
20    # Beispiel-Nutzung:
21    donalds_konto = Konto("Donald Duck")
22    donalds_konto.einzahlen(100)
23    donalds_konto.auszahlen(30)
24    donalds_konto.auszahlen(80) # Versuch, mehr Geld abzuheben als auf dem Konto vorhanden
25    # Konto erstellen
26    donalds_konto = Konto("Donald Duck")
27
28    # Einzahlungen
29    donalds_konto.einzahlen(100)
30    donalds_konto.einzahlen(50)
31
32    # Auszahlungen
33    donalds_konto.auszahlen(30)
34    donalds_konto.auszahlen(80) # Versuch, mehr Geld abzuheben als auf dem Konto vorhanden
35
36    # Kontostand ausgeben
37    print(f"Aktueller Kontostand von {donalds_konto.name}: {donalds_konto.kontostand} EUR")
```

Konto3.py

4. Damit die Ausgabe etwas einfacher wird, erweitern wir die Klasse mit der Magic Method `__str__()`. Diese Funktion muss einfach einen String zurückgeben. Der String soll den Namen und den aktuellen Kontostand enthalten. Achtung: Der Kontostand ist eine Zahl und muss mit der `str` Funktion in einen String verwandelt werden. Danach können die Strings konkateniert werden (zusammenhängen mit `+`). Alternativ kann man auch mit f-Strings arbeiten `f"Name: {name} Kontostand: {kontostand}"` wobei `name` und `kontostand` Variablen sind.

Wir können nun anstelle von `print(mein.konto.kontostand)` direkt `print(mein.konto)` verwenden. Es wird dann der String angezeigt, den `__str__()` zurück gibt. Verwende das in deinem Test-Codestück von oben.

```
1 class Konto:
2     def __init__(self, name):
3         self.name = name
4         self.kontostand = 0
5
6     def einzahlen(self, betrag):
7         if betrag > 0:
8             self.kontostand += betrag
9             print(f"{betrag} EUR wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} EUR")
10        else:
11            print("Ungültiger Betrag für Einzahlung.")
12
13    def auszahlen(self, betrag):
14        if betrag > 0 and betrag <= self.kontostand:
15            self.kontostand -= betrag
16            print(f"{betrag} EUR wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} EUR")
17        else:
18            print(f"Ungültiger Betrag für Auszahlung oder nicht ausreichend Geld auf dem Konto von {self.name}. Kontostand bleibt unverändert: {self.kontostand} EUR")
19
20    def __str__(self):
21        return f"Name: {self.name}, Kontostand: {self.kontostand} EUR"
22
23    # Beispiel-Nutzung:
24    donalds_konto = Konto("Donald Duck")
25    donalds_konto.einzahlen(100)
26    donalds_konto.einzahlen(50)
27    donalds_konto.auszahlen(30)
28    donalds_konto.auszahlen(80)
29
30    # Kontostand ausgeben
31    print(donalds_konto)
32
```

1. Lies vom Benutzer mit `input()` eine Zahl ein. Schreibe eine Funktion `null_bis_eingabe(zahl)` welche die eingegebene Zahl als Argument entgegen nimmt und eine Liste zurück gibt mit den Zahlen von Null bis zur eingegebenen Zahl. Rufe die Funktion mit der eingegebenen Zahl auf und gib die Liste mit `print` aus.

```
1 def null_bis_eingabe(zahl):
2     return list(range(zahl + 1))
3
4 # Benutzereingabe mit Fehlerüberprüfung
5 while True:
6     try:
7         eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
8         break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
9     except ValueError:
10        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
11
12 # Aufruf der Funktion und Ausgabe der Liste
13 ergebnis_liste = null_bis_eingabe(eingabe_zahl)
14 print("Liste von Null bis", eingabe_zahl, ":", ergebnis_liste)
15
```

Listen1.py

2. Eingabe wie oben, aber schreibe eine Funktion `eingabe_bis_null(zahl)`, welche eine Liste zurück gibt mit den Zahlen von der Eingabe bis Null. Also nicht `[0,1,2,3,4]` sondern `[4,3,2,1,0]`. Rufe die Funktion mit der eingegebenen Zahl auf und gib die Liste mit `print` aus.

```
1 def eingabe_bis_null(zahl):
2     return list(range(zahl, -1, -1))
3
4 # Benutzereingabe mit Fehlerüberprüfung
5 while True:
6     try:
7         eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
8         break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
9     except ValueError:
10        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
11
12 # Aufruf der Funktion und Ausgabe der Liste
13 ergebnis_liste = eingabe_bis_null(eingabe_zahl)
14 print("Liste von", eingabe_zahl, "bis Null:", ergebnis_liste)
15
```

Listen2.py

3. Eingabe wie oben, aber schreibe eine Funktion `minus_eingabe_bis_eingabe(zahl)`. Die zurück gegebene Liste soll nun die Zahlen von minus der eingegebenen Zahl bis zu der eingegebenen Zahl enthalten. Beispiel: `[-3,-2,-1,0,1,2,3]`. Rufe die Funktion mit der eingegebenen Zahl auf und gib die Liste mit `print` aus.

```
1 def minus_eingabe_bis_eingabe(zahl):
2     return list(range(-zahl, zahl + 1))
3
4 # Benutzereingabe mit Fehlerüberprüfung
5 while True:
6     try:
7         eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
8         break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
9     except ValueError:
10        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
11
12 # Aufruf der Funktion und Ausgabe der Liste
13 ergebnis_liste = minus_eingabe_bis_eingabe(eingabe_zahl)
14 print("Liste von -", eingabe_zahl, "bis", eingabe_zahl, ":", ergebnis_liste)
15
```

Listen3.py

4. Eingabe wie oben, aber schreibe eine Funktion `paare_bis_null(zahl)`. Die zurückgegebene Liste soll von der Eingabe bis Null immer die Minuszahl und dann die Pluszahl in der Liste haben. Beispiel: `[-3, 3, -2, 2, -1, 1, 0]`.

```
1 def paar_bis_null(zahl):
2     result_list = []
3     for i in range(zahl, -1, -1):
4         result_list.extend([-i, i])
5     return result_list
6
7 # Benutzereingabe mit Fehlerüberprüfung
8 while True:
9     try:
10        eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
11        break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
12    except ValueError:
13        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
14
15 # Aufruf der Funktion und Ausgabe der Liste
16 ergebnis_liste = paar_bis_null(eingabe_zahl)
17 print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)
18
```

Listen4.py

5. Zähle alle Zahlen in der List in einer Variable `summe` zusammen und gib die Summe mit `print` aus. Verwende dazu auch eine `for`-Schleife. Teste den Code mit den Listen die du mit den obigen Funktionen erstellen kannst.


```
1 def paare_bis_null(zahl):
2     result_list = []
3     for i in range(zahl, -1, -1):
4         result_list.extend([-i, i])
5     return result_list
6
7 # Benutzereingabe mit Fehlerüberprüfung
8 while True:
9     try:
10         eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
11         break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
12     except ValueError:
13         print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
14
15 # Aufruf der Funktion und Ausgabe der Liste
16 ergebnis_liste = paare_bis_null(eingabe_zahl)
17 print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)
18 def summe_der_liste(liste):
19     summe = 0
20     for zahl in liste:
21         summe += zahl
22     return summe
23
24 # Beispiel: Verwendung der Funktion paare_bis_null
25 eingabe_zahl = 3 # Hier sollte die zuvor eingegebene Zahl verwendet werden
26 ergebnis_liste = paare_bis_null(eingabe_zahl)
27
28 # Berechnung der Summe und Ausgabe
29 gesamte_summe = summe_der_liste(ergebnis_liste)
30 print("Summe der Liste:", gesamte_summe)
31
```

Listen5.py

6. Als nächstes schreiben wir eine Funktion `minus(liste)` welche eine Liste mit Zahlen als Argument entgegen nimmt und bei allen Zahlen das Vorzeichen wechselt. Die Funktion gibt nichts zurück, da die Liste direkt verändert werden soll. Zeige an einem Beispiel, dass die Funktion korrekt funktioniert, also dass zu Beispiel aus der List `[2,-3,-10]` die Liste `[-2,3,10]` wird.

```
1 def paare_bis_null(zahl):
2     result_list = []
3     for i in range(zahl, -1, -1):
4         result_list.extend([-i, i])
5     return result_list
6
7 # Benutzereingabe mit Fehlerüberprüfung
8 while True:
9     try:
10         eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
11         break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
12     except ValueError:
13         print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
14
15 # Aufruf der Funktion und Ausgabe der Liste
16 ergebnis_liste = paare_bis_null(eingabe_zahl)
17 print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)
18 def minus(liste):
19     for i in range(len(liste)):
20         liste[i] = -liste[i]
21
22 # Beispiel: Verwendung der Funktion mit einer Beispiel-Liste
23 beispiel_liste = [2, -3, -10]
24 print("Vorher:", beispiel_liste)
25
26 # Aufruf der Funktion und Änderung der Liste
27 minus(beispiel_liste)
28
29 # Ausgabe der veränderten Liste
30 print("Nachher:", beispiel_liste)
31
```

Listen6.py

IV Rennen

Wir lesen Distanzen ein die eine Person an verschiedenen Rennen gerannt ist. Hier kann nicht nur eine Distanz eingegeben werden sondern so viele bis der User nichts mehr eingeben will.

1. Lies mit einer **while**-Schleife so lange Zahlen (Distanzen) ein bis die Eingabe keine Zahl mehr ist. In anderen Worte, wenn einfach "Enter" gedrückt wird, bricht die Schleife ab (Tipp: was gibt in dem Fall die **input()** Funktion zurück?). Gib die jeweils eingegebene Zahl direkt wieder aus.

```
1 # Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern
2 distanzen = []
3
4 # Verwende eine while-Schleife, um Distanzen einzulesen, bis der Benutzer nichts eingibt
5 while True:
6     # Lies die Eingabe des Benutzers ein
7     eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")
8
9     # Überprüfe, ob die Eingabe leer ist (Enter wurde gedrückt)
10    if eingabe == "":
11        break # Beende die Schleife, wenn Enter gedrückt wurde
12
13    try:
14        # Versuche, die Eingabe in eine Zahl umzuwandeln und füge sie der Liste hinzu
15        distanz = float(eingabe)
16        distanzen.append(distanz)
17        print("Eingegebene Distanz:", distanz)
18    except ValueError:
19        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
20
21 # Gib die gesamte Liste der eingegebenen Distanzen aus
22 print("Eingegebene Distanzen:", distanzen)
23
```

Rennen1.py

2. Anstatt die Zahlen nur auszugeben, sollen die Zahlen in einer Liste gespeichert werden. Gibt am Schluss, also wenn die Schleife verlassen wurde, die Liste mit den erfassten Zahlen aus.

```
1 # Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern
2 distanzen = []
3
4 # Verwende eine while-Schleife, um Distanzen einzulesen, bis der Benutzer nichts eingibt
5 while True:
6     # Lies die Eingabe des Benutzers ein
7     eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")
8
9     # Überprüfe, ob die Eingabe leer ist (Enter wurde gedrückt)
10    if eingabe == "":
11        break # Beende die Schleife, wenn Enter gedrückt wurde
12
13    try:
14        # Versuche, die Eingabe in eine Zahl umzuwandeln und füge sie der Liste hinzu
15        distanz = float(eingabe)
16        distanzen.append(distanz)
17        print("Eingegebene Distanz:", distanz)
18    except ValueError:
19        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
20
21 # Gib die gesamte Liste der eingegebenen Distanzen aus
22 print("Eingegebene Distanzen:", distanzen)
23
```

Rennen2.py

3. Wir möchte nun eine schöne Textausgabe mit spannenden Informationen machen. Es soll stehen, wie viele Rennen eingegeben wurden, welches das kürzeste (**min()** Funktion) und das längste (**max()** Funktion) Rennen war. Zudem soll auch stehen wie lange alle

Rennen zusammen waren (sum() Funktion). Weiter soll auch die durchschnittliche Distanz
ausgegeben werden.

```
1 # Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern
2 distanzen = []
3
4 # Verwende eine while-Schleife, um Distanzen einzulesen, bis der Benutzer nichts eingibt
5 while True:
6     # Lies die Eingabe des Benutzers ein
7     eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")
8
9     # Überprüfe, ob die Eingabe leer ist (Enter wurde gedrückt)
10    if eingabe == "":
11        break # Beende die Schleife, wenn Enter gedrückt wurde
12
13    try:
14        # Versuche, die Eingabe in eine Zahl umzuwandeln und füge sie der Liste hinzu
15        distanz = float(eingabe)
16        distanzen.append(distanz)
17        print("Eingegebene Distanz:", distanz)
18    except ValueError:
19        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
20
21 # Überprüfe, ob Rennen eingegeben wurden
22 if not distanzen:
23     print("Keine Rennen eingegeben.")
24 else:
25     # Gib die Gesamtanzahl der Rennen aus
26     anzahl_rennen = len(distanzen)
27     print("Anzahl der eingegebenen Rennen:", anzahl_rennen)
28
29     # Gib die kürzeste und längste Distanz aus
30     kuerzeste_distanz = min(distanzen)
31     laengste_distanz = max(distanzen)
32     print("Kürzeste Distanz:", kuerzeste_distanz)
33     print("Längste Distanz:", laengste_distanz)
34
35     # Gib die Gesamtdistanz und die durchschnittliche Distanz aus
36     gesamtdistanz = sum(distanzen)
37     durchschnitt_distanz = gesamtdistanz / anzahl_rennen
38     print("Gesamtdistanz aller Rennen:", gesamtdistanz)
39     print("Durchschnittliche Distanz:", durchschnitt_distanz)
40
```

Rennen3.py

V Telefonbuch

Wir erstellen mit Dictionaries ein kleines Telefonbuch

1. Erstelle ein Dictionary wo die Schlüssel die Namen der Personen sind und die Werte Telefonnummern. Erfinde einfach fünf Personen und deren Telefonnummern. Gib das Dictionary mit print aus.

```
1 telefonbuch = {  
2     'Max Mustermann': '123-456789',  
3     'Eva Beispiel': '987-654321',  
4     'Anna Musterfrau': '555-123456',  
5     'Peter Test': '789-456123',  
6     'Lisa Probe': '321-654987'  
7 }  
8  
9 print(telefonbuch)  
10
```

Telefonbuch1.py

2. Schreibe eine `while`-Schleife, wie in der Aufgabe *Rennen*, welche solange wiederholt wird bis die Eingabe vom User "Enter" ist. Der User kann in jedem Durchlauf einen Namen eingeben. Gib danach mit `print` die entsprechende Telefonnummer zu dem Namen aus. Achte darauf, dass auch der Fall funktioniert, wenn der Name nicht im Dictionary ist (Tipp: `if` -Verzweigung mit einer `key in my dict` Bedingung)

```
1 telefonbuch = {  
2     'Max Mustermann': '123-456789',  
3     'Eva Beispiel': '987-654321',  
4     'Anna Musterfrau': '555-123456',  
5     'Peter Test': '789-456123',  
6     'Lisa Probe': '321-654987'  
7 }  
8  
9 while True:  
10     eingabe = input("Gib einen Namen ein (oder 'Enter' zum Beenden): ")  
11  
12     # Überprüfe, ob der Benutzer "Enter" eingegeben hat  
13     if eingabe.lower() == 'enter':  
14         break # Beende die Schleife, wenn "Enter" eingegeben wurde  
15  
16     # Überprüfe, ob der eingegebene Name im Telefonbuch ist  
17     if eingabe in telefonbuch:  
18         telefonnummer = telefonbuch[eingabe]  
19         print(f"Die Telefonnummer von {eingabe} ist: {telefonnummer}")  
20     else:  
21         print(f"Der Name {eingabe} ist nicht im Telefonbuch.")  
22  
23     # Hier gelangt der Code hin, nachdem die Schleife beendet wurde  
24     print("Programm beendet.")  
25
```

Telefonbuch2.py

3. Wir erweitern das Programm durch eine Steuerung (if-Verzweigungen). Der Benutzer wird

immer noch in einer `while`-Schleife um Eingabe gebeten. Nun soll aber der Benutzer zuerst sagen was gemacht werden soll. Dazu muss ein Buchstabe eingegeben werden und je nach Eingabe passiert etwas anderes. Schreibe für jeden Modus eine eigene Funktion die je nach Eingabe aufgerufen wird.

- Buchstabe "t" - Telefonnummer-Modus: Es soll das passieren was gerade in der vorherigen Aufgabe programmiert wurde. Also Name eingeben und die passende Telefonnummer ausgeben.
- Buchstabe "l" - Löschen-Modus: Es wird nach dem Namen gefragt, welcher aus dem Dictionary gelöscht werden soll (`pop(key)` Methode des Dictionarys).
- Buchstabe "a" - Ändern-Modus: Es wird zuerst nach dem Namen gefragt und dann nach der Telefonnummer. Das Dictionary soll mit der Eingabe erweitert werden. Das heisst wenn der Name noch nicht in dem Dictionary ist wird ein neuer Eintrag gemacht und wenn es den Namen bereits gibt wird die Telefonnummer geändert.
- Buchstabe "?" - Such-Modus: Der User kann einen Buchstaben eingeben. Es werden dann alle Namen, welche mit diesem Buchstaben beginnen, ausgegeben. Gib auch gerade die dazugehörigen Telefonnummern aus. (Tipp: Ein String ist wie eine Liste, es kann mit `[]` auf einzelne Buchstaben im String zugegriffen werden.). Hier muss mit einer `for`-Schleife über alle Schlüssel (`keys()` Methode des Dictionarys) iteriert werden und geprüft werden, ob der erste Buchstabe mit der Eingabe übereinstimmt.

```
1 telefonbuch = {
2     'Max Mustermann': '123-456789',
3     'Eva Beispiel': '987-654321',
4     'Anna Musterfrau': '555-123456',
5     'Peter Test': '789-456123',
6     'Lisa Probe': '321-654987'
7 }
8
9 def telefonnummer_modus():
10     eingabe = input("Gib einen Namen ein: ")
11     telefonnummer = telefonbuch.get(eingabe, "Der Name ist nicht im Telefonbuch.")
12     print(f"Die Telefonnummer von {eingabe} ist: {telefonnummer}")
13
14 def loesch_modus():
15     eingabe = input("Gib den Namen ein, der gelöscht werden soll: ")
16     if eingabe in telefonbuch:
17         del telefonbuch[eingabe]
18         print(f"Der Eintrag für {eingabe} wurde gelöscht.")
19     else:
20         print(f"Der Name {eingabe} ist nicht im Telefonbuch.")
21
22 def aendern_modus():
23     eingabe_name = input("Gib den Namen ein: ")
24     eingabe_telefonnummer = input("Gib die neue Telefonnummer ein: ")
25     telefonbuch[eingabe_name] = eingabe_telefonnummer
26     print(f"Der Eintrag für {eingabe_name} wurde geändert.")
27
28 def such_modus():
29     eingabe = input("Gib einen Buchstaben ein: ")
30     gefunden = False
31     for name, telefonnummer in telefonbuch.items():
32         if eingabe.lower() in name.lower():
33             print(f"{name}: {telefonnummer}")
34             gefunden = True
35     if not gefunden:
36         print(f"Keine Einträge gefunden, die den Buchstaben '{eingabe}' enthalten.")
37
```

```
38 while True:
39     print("\nWähle einen Modus:")
40     print("t - Telefonnummer-Modus")
41     print("l - Lösch-Modus")
42     print("a - Ändern-Modus")
43     print("? - Such-Modus")
44     print("Enter - Beenden")
45
46     modus = input("Gib einen Buchstaben ein: ")
47
48     if modus.lower() == 'enter':
49         break
50
51     if modus.lower() == 't':
52         telefonnummer_modus()
53     elif modus.lower() == 'l':
54         loesch_modus()
55     elif modus.lower() == 'a':
56         aendern_modus()
57     elif modus.lower() == '?':
58         such_modus()
59     else:
60         print("Ungültige Eingabe. Bitte wähle einen der Buchstaben t, l, a, ? oder 'Enter' zum Beenden.")
61
```

4. Als nächste Erweiterung programmieren wir, das speichern und laden des Telefonbuches. Dazu gibt es zwei weitere Befehle:

- Buchstabe "s" - Speichern: Schreibe das Dictionary in die Datei "telefonbuch.csv". Wobei jeder Eintrag auf einer Zeile stehen soll und der Vorname und die Telefonnummer mit einem ; getrennt seid. Beispiel: Donald Duck;0799999999

Um in eine Datei zu schreiben, kann das folgende Codestück verwendet und erweitert werden. Mit write kann ein String in die Datei geschrieben werden. Aber es muss noch der Zeilenumbruch \n hinzugefügt werden, sonst steht alles auf derselben Zeile.

```
with open("telefonbuch.csv", "w") as file:
    file.write("Donald Duck;0799999999 \n")
```

- Buchstabe "o" - Öffnen: Lade die Einträge in "telefonbuch.csv" in das Dictionary. Dazu muss die Datei geöffnet werden. Jede Zeile wird mit line.split(";") in Name und Telefonnummer zerteilt und in das Dictionary eingefügt. Es kann folgendes Codestück verwendet und erweitert werden, um jede Zeile in der Datei zu verarbeiten.

```
with open("telefonbuch.csv") as file:
    for line in file:
        print(line)
```



```
1 import csv
2 import os
3
4 telefonbuch = {
5     'Max Mustermann': '123-456789',
6     'Eva Beispiel': '987-654321',
7     'Anna Musterfrau': '555-123456',
8     'Peter Test': '789-456123',
9     'Lisa Probe': '321-654987'
10 }
11
12 def telefonnummer_modus():
13     eingabe = input("Gib einen Namen ein: ")
14     telefonnummer = telefonbuch.get(eingabe, "Der Name ist nicht im Telefonbuch.")
15     print(f"Die Telefonnummer von {eingabe} ist: {telefonnummer}")
16
17 def loesch_modus():
18     eingabe = input("Gib den Namen ein, der gelöscht werden soll: ")
19     if eingabe in telefonbuch:
20         del telefonbuch[eingabe]
21         print(f"Der Eintrag für {eingabe} wurde gelöscht.")
22     else:
23         print(f"Der Name {eingabe} ist nicht im Telefonbuch.")
24
25 def aendern_modus():
26     eingabe_name = input("Gib den Namen ein: ")
27     eingabe_telefonnummer = input("Gib die neue Telefonnummer ein: ")
28     telefonbuch[eingabe_name] = eingabe_telefonnummer
29     print(f"Der Eintrag für {eingabe_name} wurde geändert.")
30
31 def such_modus():
32     eingabe = input("Gib einen Buchstaben ein: ")
33     gefunden = False
34     for name, telefonnummer in telefonbuch.items():
35         if eingabe.lower() in name.lower():
36             print(f"{name}: {telefonnummer}")
37             gefunden = True
38     if not gefunden:
```

```

38     if not gefunden:
39         print(f"Keine Einträge gefunden, die den Buchstaben '{eingabe}' enthalten.")
40
41 def speichern():
42     try:
43         with open("telefonbuch.csv", "w", newline='') as file:
44             writer = csv.writer(file, delimiter=';')
45             for name, telefonnummer in telefonbuch.items():
46                 writer.writerow([name, telefonnummer])
47             print("Telefonbuch erfolgreich gespeichert.")
48     except Exception as e:
49         print(f"Fehler beim Speichern: {e}")
50
51 def oeffnen():
52     try:
53         telefonbuch.clear() # Löscht bestehende Einträge, um die Datei zu laden
54         if os.path.exists("telefonbuch.csv"):
55             with open("telefonbuch.csv", newline='') as file:
56                 reader = csv.reader(file, delimiter=';')
57                 for row in reader:
58                     if len(row) == 2:
59                         name, telefonnummer = row
60                         telefonbuch[name] = telefonnummer
61             print("Telefonbuch erfolgreich geladen.")
62         else:
63             print("Die Datei existiert noch nicht. Es wurden keine Einträge geladen.")
64     except Exception as e:
65         print(f"Fehler beim Laden: {e}")
66
67 # Speichern und Öffnen beim Programmstart
68 speichern()
69 oeffnen()
70
71 while True:
72     print("\nWähle einen Modus:")
73     print("t - Telefonnummer-Modus")
74     print("l - Lösch-Modus")
75     print("a - Ändern-Modus")
76     print("? - Such-Modus")
77
78     # Optionen nur anzeigen, wenn bereits gespeichert und geladen wurde
79     if telefonbuch:
80         print("s - Speichern")
81         print("o - Öffnen")
82
83     print("Enter - Beenden")
84
85     modus = input("Gib einen Buchstaben ein: ").lower()
86
87     if modus == 't':
88         telefonnummer_modus()
89     elif modus == 'l':
90         loesch_modus()
91     elif modus == 'a':
92         aendern_modus()
93     elif modus == '?':
94         such_modus()
95     elif modus == 's':
96         speichern()
97     elif modus == 'o':
98         oeffnen()
99     elif not modus:
100         break
101     else:
102         print("Ungültige Eingabe. Bitte wähle einen der Buchstaben t, l, a, ?, s, o oder drücke Enter zum Beenden.")
103

```