

Übung 4: Grundlagen



I Rechner

1. Lies mit der input() Funktion eine Rechnung ein. Dazu soll dreimal input() aufgerufen werden. Das erste Mal für die erst 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.

```
# Eingabe der ersten Zahl
zahl1 = input("Gib die erste Zahl ein: ")

# Eingabe des Operators (+ oder -)
operator = input("Gib + oder - ein: ")

# Eingabe der zweiten Zahl
zahl2 = input("Gib die zweite Zahl ein: ")

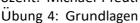
# Ausgabe der Rechnung
print(f"{zahl1}{operator}{zahl2}")
```

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.

Rechner2.py

3. Erweitere die ${\it if}$ -Verzweigung für * und / also Mal und Durch-Rechnen.



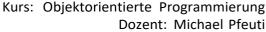


```
# Eingabe der ersten Zahl
     zahl1 = float(input("Gib die erste Zahl ein: "))
     # Eingabe des Operators (+, -, *, /)
     operator = input("Gib +, -, *, oder / ein: ")
     # Eingabe der zweiten Zahl
     zahl2 = float(input("Gib die zweite Zahl ein: "))
     # Initialisierung der Variable für das Ergebnis
     ergebnis = 0
     # Überprüfung des Operators und Berechnung des Ergebnisses
     if operator == "+":
         ergebnis = zahl1 + zahl2
     elif operator == "-":
         ergebnis = zahl1 - zahl2
     elif operator == "*":
         ergebnis = zahl1 * zahl2
     elif operator == "/":
         # Überprüfung, ob der Divisor nicht 0 ist
         if zahl2 != 0:
             ergebnis = zahl1 / zahl2
         else:
             print("Division durch 0 ist nicht erlaubt.")
             exit() # Programm beenden, da eine Division durch 0 versucht wurde
         print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
         exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
     # Ausgabe der Rechnung und des Ergebnisses
     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.



Übung 4: Grundlagen



```
berechne(zahl1, zahl2, operator):
    Die Funktion berechne führt eine Rechnung basierend auf den gegebenen Zahlen und dem Operator durch.
    if operator == "+":
       return zahl1 + zahl2
    elif operator == "-":
       return zahl1 - zahl2
    elif operator == "*":
       return zahl1 * zahl2
    elif operator == "/":
       if zahl2 != 0:
           return zahl1 / zahl2
           print("Division durch 0 ist nicht erlaubt.")
            exit() # Programm beenden, da eine Division durch 0 versucht wurde
       print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
        exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
# Eingabe der ersten Zahl
zahl1 = float(input("Gib die erste Zahl ein: "))
operator = input("Gib +, -, *, oder / ein: ")
zahl2 = float(input("Gib die zweite Zahl ein: "))
ergebnis = berechne(zahl1, zahl2, operator)
print(f"{zahl1} {operator} {zahl2} = {ergebnis}")
```

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.

Übung 4: Grundlagen

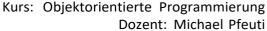


```
berechne(zahl1, zahl2, operator):
    Die Funktion berechne führt eine Rechnung basierend auf den gegebenen Zahlen und dem Operator durch.
    if operator == "+":
        return zahl1 + zahl2
    elif operator == "-":
       return zahl1 - zahl2
    elif operator == "*":
        return zahl1 * zahl2
    elif operator == "/":
        if zahl2 != 0:
            return zahl1 / zahl2
            print("Division durch 0 ist nicht erlaubt.")
            exit() # Programm beenden, da eine Division durch 0 versucht wurde
        print("Ungültiger Operator. Bitte nur +, -, *, oder / eingeben.")
        exit() # Programm beenden, da ein ungültiger Operator eingegeben wurde
eingabe = input("Gib die Rechnung ein (z.B. '4 + 9'): ")
teile = eingabe.split()
# Überprüfe, ob die Eingabe gültig ist
if len(teile) != 3:
    print("Ungültige Eingabe. Bitte die Rechnung mit Leerzeichen trennen (z.B. '4 + 9').")
    exit()
zahl1 = float(teile[0])
operator = teile[1]
zahl2 = float(teile[2])
ergebnis = berechne(zahl1, zahl2, operator)
print(f"{eingabe} = {ergebnis}")
```

Rechner5.py

II Konto

 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 wir im Konstruktor auf 0 gesetzt.



.. Übung 4: Grundlagen



```
def __init__(self, name):
        self.name = name
        self.kontostand = 0
    def einzahlen(self, betrag):
        if betrag > 0:
            self.kontostand += betrag
            print(f"{betrag} CHF wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} CHF")
            print("Ungültiger Betrag für Einzahlung.")
    def auszahlen(self, betrag):
        if 0 < betrag <= self.kontostand:</pre>
            self.kontostand -= betrag
            print(f"{betrag} CHF wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} CHF")
            print("Ungültiger Betrag für Auszahlung.")
donalds_konto = Konto("Donald Duck")
donalds_konto.einzahlen(100)
donalds_konto.auszahlen(30)
```

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 genügen Geld da ist. Wenn der Kontostand zu tief ist, soll das mit print ausgegeben werden und natürlich den Kontostand nicht verändern.

```
class Konto:

def _init_(self, name):
    self.name = name
    self.kontostand = 0

def einzahlen(self, betrag):
    if betrag > 0:
        self.kontostand += betrag
        print(f*(betrag) EUR wurden auf das Konto von {self.name} eingezahlt. Neuer Kontostand: {self.kontostand} EUR")

def auszahlen(self, betrag):
    if betrag > 0 and betrag <= self.kontostand:
        self.kontostand -= betrag
        print(f*(betrag) EUR wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} EUR")

def auszahlen(self, betrag):
    if betrag > 0 and betrag <= self.kontostand:
        self.kontostand -= betrag
        print(f*(betrag) EUR wurden vom Konto von {self.name} abgehoben. Neuer Kontostand: {self.kontostand} EUR")

delse:
    print(f*Ungültiger Betrag für Auszahlung oder nicht ausreichend Geld auf dem Konto von {self.name}. Kontostand bleibt unverändert: {self.kontostand} EUR")

## Beispiel-Nutzung:

donalds_konto = Konto("Donald Duck")

donalds_konto = konto("Donald Duck")

donalds_konto.auszahlen(30)

donalds_konto.auszahlen(30) # Versuch, mehr Geld abzuheben als auf dem Konto vorhanden
```

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.

.. Übung 4: Grundlagen



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.

```
class Konto:

class Konto

class Kon
```





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 eingegebene Zahl auf und gib die Liste mit print aus.

```
def null_bis_eingabe(zahl):
    return list(range(zahl + 1))

# Benutzereingabe mit Fehlerüberprüfung

while True:

try:
    eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))

break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde

except ValueError:

print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Aufruf der Funktion und Ausgabe der Liste

ergebnis_liste = null_bis_eingabe(eingabe_zahl)

print("Liste von Null bis", eingabe_zahl, ":", ergebnis_liste)
```

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.

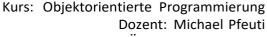
```
def eingabe_bis_null(zahl):
    return list(range(zahl, -1, -1))

# Benutzereingabe mit Fehlerüberprüfung
while True:
    try:
    eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
    break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
except ValueError:
    print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Aufruf der Funktion und Ausgabe der Liste
ergebnis_liste = eingabe_bis_null(eingabe_zahl)
print("Liste von", eingabe_zahl, "bis Null:", ergebnis_liste)
```

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.



Übung 4: Grundlagen



```
def minus_eingabe_bis_eingabe(zahl):
    return list(range(-zahl, zahl + 1))

# Benutzereingabe mit Fehlerüberprüfung
while True:
    try:
    eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
    break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
except ValueError:
    print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Aufruf der Funktion und Ausgabe der Liste
ergebnis_liste = minus_eingabe_bis_eingabe(eingabe_zahl)
print("Liste von -", eingabe_zahl, "bis", eingabe_zahl, ":", ergebnis_liste)
```

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].

```
def paare_bis_null(zahl):
    result_list = []
    for i in range(zahl, -1, -1):
        result_list.extend([-i, i])
    return result_list

# Benutzereingabe mit Fehlerüberprüfung
while True:

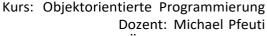
try:
    eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
    break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
except ValueError:
    print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Aufruf der Funktion und Ausgabe der Liste
ergebnis_liste = paare_bis_null(eingabe_zahl)
print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)

# Aufruf der Funktion und Ausgabe der Liste
```

Listen4.py

5. Zähle alle Zahlen in der List in einer Variable summe zusammen und gibt 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.



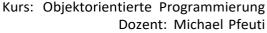
.. Übung 4: Grundlagen



```
def paare_bis_null(zahl):
    result_list = []
    for i in range(zahl, -1, -1):
        result_list.extend([-i, i])
    return result_list
# Benutzereingabe mit Fehlerüberprüfung
        eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
        break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
    except ValueError:
        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
ergebnis_liste = paare_bis_null(eingabe_zahl)
print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)
def summe_der_liste(liste):
    summe = 0
    for zahl in liste:
       summe += zahl
    return summe
# Beispiel: Verwendung der Funktion paare_bis_null
eingabe_zahl = 3  # Hier sollte die zuvor eingegebene Zahl verwendet werden
ergebnis_liste = paare_bis_null(eingabe_zahl)
gesamte_summe = summe_der_liste(ergebnis_liste)
print("Summe der Liste:", gesamte_summe)
```

Listen5.py

6. Als nächstes schreiben wir eine Funktion minus(liste) welche ein 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]



... Übung 4: Grundlagen



```
paare bis null(zahl):
         result_list = []
         for i in range(zahl, -1, -1):
             result_list.extend([-i, i])
         return result_list
             eingabe_zahl = int(input("Bitte gib eine Zahl ein: "))
             break # Beende die Schleife, wenn die Eingabe erfolgreich in eine Zahl umgewandelt wurde
         except ValueError:
             print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
     ergebnis_liste = paare_bis_null(eingabe_zahl)
     print("Paare von -", eingabe_zahl, "bis Null:", ergebnis_liste)
     def minus(liste):
         for i in range(len(liste)):
             liste[i] = -liste[i]
     # Beispiel: Verwendung der Funktion mit einer Beispiel-Liste
     beispiel_liste = [2, -3, -10]
     print("Vorher:", beispiel_liste)
     # Aufruf der Funktion und Änderung der Liste
     minus(beispiel liste)
     # Ausgabe der veränderten Liste
     print("Nachher:", beispiel_liste)
30
```

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.

Kurs: Objektorientierte Programmierung

Dozent: Michael Pfeuti Übung 4: Grundlagen



```
# Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern

distanzen = []

# Verwende eine while-Schleife, um Distanzen einzulesen, bis der Benutzer nichts eingibt

while True:

# Lies die Eingabe des Benutzers ein

eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")

# Überprüfe, ob die Eingabe leer ist (Enter wurde gedrückt)

if eingabe == "":

| break # Beende die Schleife, wenn Enter gedrückt wurde

try:

# Versuche, die Eingabe in eine Zahl umzuwandeln und füge sie der Liste hinzu

distanz = float(eingabe)

distanzen.append(distanz)

print("Eingegebene Distanz:", distanz)

except ValueError:

print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Gib die gesamte Liste der eingegebenen Distanzen aus

print("Eingegebene Distanzen:", distanzen)
```

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.

```
# Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern

distanzen = []

# Verwende eine while-Schleife, um Distanzen einzulesen, bis der Benutzer nichts eingibt

while True:

# Lies die Eingabe des Benutzers ein

eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")

# Überprüfe, ob die Eingabe leer ist (Enter wurde gedrückt)

if eingabe == "":

break # Beende die Schleife, wenn Enter gedrückt wurde

try:

# Versuche, die Eingabe in eine Zahl umzuwandeln und füge sie der Liste hinzu

distanz = float(eingabe)

distanzzen.append(distanz)

print("Eingegebene Distanz:", distanz)

except ValueError:

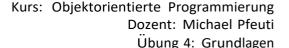
print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")

# Gib die gesamte Liste der eingegebenen Distanzen aus

print("Eingegebene Distanzen:", distanzen)
```

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.

```
# Initialisiere eine leere Liste, um die eingegebenen Distanzen zu speichern
distanzen = []
    eingabe = input("Gib eine Distanz ein (oder drücke Enter zum Beenden): ")
    if eingabe == "":
        break # Beende die Schleife, wenn Enter gedrückt wurde
        distanz = float(eingabe)
        distanzen.append(distanz)
        print("Eingegebene Distanz:", distanz)
        print("Ungültige Eingabe. Bitte gib eine gültige Zahl ein.")
# Überprüfe, ob Rennen eingegeben wurden
if not distanzen:
    print("Keine Rennen eingegeben.")
    anzahl_rennen = len(distanzen)
    print("Anzahl der eingegebenen Rennen:", anzahl_rennen)
    # Gib die kürzeste und längste Distanz aus
    kuerzeste_distanz = min(distanzen)
    laengste_distanz = max(distanzen)
    print("Kürzeste Distanz:", kuerzeste_distanz)
    print("Längste Distanz:", laengste_distanz)
    gesamtdistanz = sum(distanzen)
    durchschnitt_distanz = gesamtdistanz / anzahl_rennen
    print("Gesamtdistanz aller Rennen:", gesamtdistanz)
    print("Durchschnittliche Distanz:", durchschnitt_distanz)
```

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.

Übung 4: Grundlagen



```
telefonbuch = {
         'Max Mustermann': '123-456789',
         'Eva Beispiel': '987-654321',
         'Anna Musterfrau': '555-123456',
         'Peter Test': '789-456123',
         'Lisa Probe': '321-654987'
     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)

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

Telefonbuch2.py



Kurs: Objektorientierte Programmierung

Dozent: Michael Pfeuti Übung 4: Grundlagen

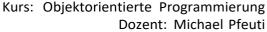
3. Wir erweitern das Programm durch ein Steuerung (if -Verzeigungen). Der Benutzer wird immer noch in einer while-Schleife um Eingabe gebeten. Nun soll aber der Benutzer zuerst sagen was gemacht werden soll. Dazu muss ein Buchstabe eingegeben werden und je nach Eingabe passiert etwas anderes. Schreibe 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 "I" Lösch-Modus: Es wird nach dem Namen gefragt, welcher aus dem Dictionary gelöscht werden soll (pop(key) Methode des Dictionaries).
- Buchstabe "a" Ändern-Modus: Es 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 Dictionaries) iteriert werden und geprüft werden, ob der erste Buchstabe mit der Eingabe übereinstimmt.

Ubung 4: Grundlagen



```
telefonbuch = {
    'Eva Beispiel': '987-654321',
    'Anna Musterfrau': '555-123456',
    'Peter Test': '789-456123',
    'Lisa Probe': '321-654987'
def telefonnummer modus():
    eingabe = input("Gib einen Namen ein: ")
    telefonnummer = telefonbuch.get(eingabe, "Der Name ist nicht im Telefonbuch.")
    print(f"Die Telefonnummer von {eingabe} ist: {telefonnummer}")
def loesch_modus():
    eingabe = input("Gib den Namen ein, der gelöscht werden soll: ")
    if eingabe in telefonbuch:
        del telefonbuch[eingabe]
        print(f"Der Eintrag für {eingabe} wurde gelöscht.")
        print(f"Der Name {eingabe} ist nicht im Telefonbuch.")
def aendern modus():
    eingabe_name = input("Gib den Namen ein: ")
    eingabe_telefonnummer = input("Gib die neue Telefonnummer ein: ")
    telefonbuch[eingabe_name] = eingabe_telefonnummer
    print(f"Der Eintrag für {eingabe_name} wurde geändert.")
def such_modus():
    eingabe = input("Gib einen Buchstaben ein: ")
    gefunden = False
    for name, telefonnummer in telefonbuch.items():
        if eingabe.lower() in name.lower():
            print(f"{name}: {telefonnummer}")
            gefunden = True
    if not gefunden:
        print(f"Keine Einträge gefunden, die den Buchstaben '{eingabe}' enthalten.")
```



.. Ubung 4: Grundlagen



```
print("\nWähle einen Modus:")
         print("t - Telefonnummer-Modus")
        print("l - Lösch-Modus")
         print("a - Ändern-Modus")
         print("? - Such-Modus")
         print("Enter - Beenden")
         modus = input("Gib einen Buchstaben ein: ")
         if modus.lower() == 'enter':
            break
50
         if modus.lower() == 't':
            telefonnummer modus()
         elif modus.lower() == 'l':
            loesch_modus()
         elif modus.lower() == 'a':
             aendern_modus()
         elif modus.lower() == '?':
            such modus()
             print("Ungültige Eingabe. Bitte wähle einen der Buchstaben t, l, a, ? oder 'Enter' zum Beenden.")
```

- 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)
```

Kurs: Objektorientierte Programmierung

Dozent: Michael Pfeuti Übung 4: Grundlagen



```
import os
telefonbuch = {
    'Max Mustermann': '123-456789',
    'Eva Beispiel': '987-654321',
'Anna Musterfrau': '555-123456',
    'Peter Test': '789-456123',
'Lisa Probe': '321-654987'
def telefonnummer_modus():
    eingabe = input("Gib einen Namen ein: ")
    telefonnummer = telefonbuch.get(eingabe, "Der Name ist nicht im Telefonbuch.")
    print(f"Die Telefonnummer von {eingabe} ist: {telefonnummer}")
def loesch modus():
    eingabe = input("Gib den Namen ein, der gelöscht werden soll: ")
    if eingabe in telefonbuch:
        del telefonbuch[eingabe]
        print(f"Der Eintrag für {eingabe} wurde gelöscht.")
        print(f"Der Name {eingabe} ist nicht im Telefonbuch.")
def aendern_modus():
    eingabe_name = input("Gib den Namen ein: ")
    eingabe_telefonnummer = input("Gib die neue Telefonnummer ein: ")
    telefonbuch[eingabe_name] = eingabe_telefonnummer
    print(f"Der Eintrag für {eingabe_name} wurde geändert.")
def such_modus():
    eingabe = input("Gib einen Buchstaben ein: ")
    gefunden = False
    for name, telefonnummer in telefonbuch.items():
        if eingabe.lower() in name.lower():
             print(f"{name}: {telefonnummer}")
             gefunden = True
```

.. Übung 4: Grundlagen



```
print(f"Keine Einträge gefunden, die den Buchstaben '{eingabe}' enthalten.")
def speichern():
         with open("telefonbuch.csv", "w", newline='') as file:
    writer = csv.writer(file, delimiter=';')
             for name, telefonnummer in telefonbuch.items():
                 writer.writerow([name, telefonnummer])
         print("Telefonbuch erfolgreich gespeichert.")
    except Exception as e:
         print(f"Fehler beim Speichern: {e}")
def oeffnen():
         telefonbuch.clear() # Löscht bestehende Einträge, um die Datei zu laden
         if os.path.exists("telefonbuch.csv"):
             with open("telefonbuch.csv", newline='') as file:
                  reader = csv.reader(file, delimiter=';')
                  for row in reader:
                      if len(row) == 2:
                           name, telefonnummer = row
                           telefonbuch[name] = telefonnummer
             print("Telefonbuch erfolgreich geladen.")
             print("Die Datei existiert noch nicht. Es wurden keine Einträge geladen.")
    except Exception as e:
         print(f"Fehler beim Laden: {e}")
# Speichern und Öffnen beim Programmstart
speichern()
oeffnen()
    print("\nWähle einen Modus:")
    print("t - Telefonnummer-Modus")
    print("l - Lösch-Modus")
  print("a - Ändern-Modus")
print("? - Such-Modus")
  # Optionen nur anzeigen, wenn bereits gespeichert und geladen wurde
  if telefonbuch:
      print("s - Speichern")
print("o - Öffnen")
  print("Enter - Beenden")
  modus = input("Gib einen Buchstaben ein: ").lower()
  if modus == 't':
      telefonnummer modus()
  elif modus == 'l':
      loesch_modus()
  elif modus == 'a':
      aendern_modus()
  elif modus == '?':
      such_modus()
  elif modus == 's':
      speichern()
  elif modus ==
      oeffnen()
  elif not modus:
      print("Ungültige Eingabe. Bitte wähle einen der Buchstaben t, l, a, ?, s, o oder drücke Enter zum Beenden.")
```