Universität der Bundeswehr
Fakultät für Informatik – Institut 2
Prof. Dr. Wolfgang Hommel
Dr. Michael Grabatin, Kim Mönch

WT2025
Übungsblatt 3

Einführung in die Programmierung

29.01.2025

1 Berechnung und Speichern von Primzahlen

a) Schreiben Sie eine Funktion `is_prime(x)`, die bestimmt, ob ein gegebener Parameter vom Typ `Int` eine Primzahl ist oder nicht. Der Rückgabewert ist entsprechend `True` oder `False`.

b) Schreiben Sie eine Funktion `primes_up_to_n(n)`, die die in der vorherigen Aufgabe implementierte Funktion `is_prime` benutzt, um alle Primzahlen zwischen 0 und `n` zu berechnen. Speichern Sie außerdem alle gefundenen Primzahlen mittels Listenkomprehension in einer Liste und benutzen Sie diese Liste als Rückgabewert der Funktion.

c) Schreiben Sie eine einfache Funktion `has_number(x, lst)` zum Suchen einer bestimmten Zahl `x` in der Liste `lst` mit einer **`for`**-Schleife. Wenn die Liste die gesuchte Zahl enthält, dann geben Sie `True` zurück, andernfalls `False`.
Benutzen Sie dazu eine **`for-else`**-Schleife. Erläutern Sie in eigenen Worten, welchen Vorteil diese Variante im Gegensatz zu klassischen **`for`**-Schleifen bringen kann.

2 Listen und *slicing*

In dieser Aufgabe werden verschiedene Listenoperationen, insbesondere *slicing*, ausprobiert. Für alle Teilaufgaben ist dafür die Liste `zahlen = [1, 2, 3, 4, 5, 6, 7, 8, 9]` gegeben.

a) Geben Sie über Index-zugriffe die folgenden Positionen der Liste aus:

- Das erste Element.
- Die letzten Element.
- Das „mittlere“Element (wenn es kein echtes mittleres Element gibt, ist das links von der Mitte gemeint).

b) Verwenden Sie *slicing*, um die folgenden Positionen der Liste auszugeben:

- Die ersten drei Elemente.
- Die letzten drei Elemente.
- Jedes zweite Element, beginnend mit dem Ersten.
- Jedes zweite Element, beginnend mit dem Zweiten.
- Die Liste in umgekehrter Reihenfolge.

c) Iterieren Sie mit Hilfe einer **`for`**-Schleife über alle Elemente der Liste und geben Sie die Elemente einzeln aus. Verwenden Sie dazu drei verschiedene Varianten der **`for`**-Schleife:

- Über Index-basierte Zugriffe mit einer **`range`**. (Ausgabe über Index-Zugriff `print(zahlen[i])`)
- Nur mit dem Keyword **`in`**. (Ausgabe des Werts direkt: `print(x)`)
- Mit Hilfe von **`enumerate`**. (Ausgabe als Index und Wert: `print("{}: {}".format(i, x))`)

3 Wurzeltabelle

In dieser Aufgabe sollen Sie eine Tabelle zur schnellen Bestimmung der Quadratwurzel erstellen. Die Zellenwerte der Tabelle geben dabei die Quadratwurzel des Werts, der sich durch Konkatenation des Index der Zeile und der Reihe ergibt, an. Die Tabelle soll dabei wie folgt aussehen:

```
    0      1      2      3      4      5      6      7      8      9
 0  0.000  1.000  1.414  1.732  2.000  2.236  2.449  2.646  2.828  3.000
 1  3.162  3.317  3.464  3.606  3.742  3.873  4.000  4.123  4.243  4.359
 2  4.472  4.583  4.690  4.796  4.899  5.000  5.099  5.196  5.292  5.385
 3  5.477  5.568  5.657  5.745  5.831  5.916  6.000  6.083  6.164  6.245
 4  6.325  6.403  6.481  6.557  6.633  6.708  6.782  6.856  6.928  7.000
 5  7.071  7.141  7.211  7.280  7.348  7.416  7.483  7.550  7.616  7.681
 6  7.746  7.810  7.874  7.937  8.000  8.062  8.124  8.185  8.246  8.307
 7  8.367  8.426  8.485  8.544  8.602  8.660  8.718  8.775  8.832  8.888
 8  8.944  9.000  9.055  9.110  9.165  9.220  9.274  9.327  9.381  9.434
 9  9.487  9.539  9.592  9.644  9.695  9.747  9.798  9.849  9.899  9.950
10 10.000 10.050 10.100 10.149 10.198 10.247 10.296 10.344 10.392 10.440
```

Zum Beispiel ist der dritte Eintrag in der ersten Zeile (1.4142) der Wert von $\sqrt{(02)}$, der erste Wert in der letzten Zeile (10.0000) ist der Wert von $\sqrt{(100)}$.

a) Erstellen Sie eine Funktion, die eine verschachtelte Liste mit den Wurzelwerten berechnet. Entsprechend der Tabelle oben, sollen in der ersten Zeile die Wurzelwerte der Zahlen 0 bis 9 stehen, in der zweiten Zeile die von 10 bis 19. Die Funktion soll einen Parameter für die obere Grenze der Anzahl der Zeilen entgegen nehmen.
Um die Wurzel einer Zahl zu berechnen, müssen Sie zunächst das Modul `math` mit der Anweisung `import math` importieren. Danach können Sie über `math.sqrt( )` die Wurzel einer beliebigen Zahl berechnen.

b) Verwenden Sie die Funktion, die Sie in Teilaufgabe a) entwickelt haben um die Tabelle wie oben angegeben auszugeben.
Damit die Formatierung der Zellen stimmt, beachten Sie dazu die Dokumentation zu Formatstrings (`https://docs.python.org/3/library/string.html#format-string-syntax`) sowie die Beispiele zu **`format`** aus der Vorlesung. Um einzelne **`print`**-Befehle in einer Zeile zu kombinieren, können Sie den optionalen Parameter `end` verwenden, um keinen automatischen Zeilenumbruch einzufügen: **`print`**`("foo", end="")`.

c) Implementieren Sie nun eine Funktion `d33pc0py`, die eine *Deep Copy* von der verschachtelten Liste erstellt. Benutzen Sie dafür *nicht* die `deepcopy`-Funktion von Python selbst.
Testen Sie die Funktion der `d33pc0py`-Funktion geeignet.