Python-Programmieraufgaben

Prüfungsvorbereitung

Aufgabe 1

Schreibe eine Python-Funktion `list_max(L)`, die das grösste Element der Liste `L` zurückgibt.

Hinweis: Weise dem Maximum `m` den provisorischen Wert `float('-inf')`; d. h. $-\infty$ zu und überprüfe in einer Schleife über alle Listenelemente, ob sich das jeweils aktuelle Maximum durch ein noch grösseres Element ersetzen lässt. Falls ja, ersetze es.

Aufgabe 1

```
1  def list_max(L):
2      '''Gibt das Maximimum der Elemente von L zurück.'''
3      m = float('-inf')
4      for x in L:
5          if x > m:
6              m = x
7      return m
8
9  LoL = [ [], [14], [3,1,-4, 19, 12], [3, 3, 3, 3, 3]]
10 for L in LoL:
11     print(L, list_max(L))
```

Aufgabe 2

Schreibe eine Python-Funktion `list_find(item, L)`, die `True` zurückgibt, wenn `item` in der Liste `L` ist und `False` sonst.

Aufgabe 2

```
1  def list_find(item, L):
2      '''Gibt True zurück, wenn "item" Element der Liste
           "L" ist.'''
3      for x in L:
4          if x == item:
5              return True
6      return False
7
8  # Testcode
9  print(list_find(5, [3, 5]))
10 print(list_find('x', ['ab', 'xz', 'pq']))
11 print(list_find(False, [True, True, True]))
```

Aufgabe 3

Schreibe eine Python-Funktion `mean(L)`, welche das arithmetische Mittel der Elemente in der Liste `L` berechnet und zurückgibt.

Welcher Spezialfall muss berücksichtigt werden und wie soll man ihn verarbeiten?

Aufgabe 3

```
1  def mean(L): # pythonic
2      '''Gibt arithmetisches Mittel der Elemente in L
           zurück'''
3      return sum(L)/len(L)
4
5  def mean_classic(L):
6      '''Gibt arithmetisches Mittel der Elemente in L
           zurück'''
7      summe = 0
8      for i in range(len(L)):
9          summe = summe + L[i]
10     return summe/len(L)
11
12 # Was fehlt: Vorher überprüfen, ob die Liste leer ist
13 # und dann "False" oder "None" zurückgeben.
```

Aufgabe 4

Schreibe eine Python-Funktion `is_sorted(L)`, die `True` zurückgibt, wenn die Liste `L` aufsteigend sortiert ist; d. h. wenn mit Ausnahme des letzten Elements kein Element grösser als sein Nachfolger ist. Andernfalls soll die Funktion `False` zurückgeben.

Überlege auch, welcher Rückgabewert bei Listen der Länge 0 oder 1 sinnvoll ist.

Aufgabe 4

```
1  def is_sorted(L):
2      '''Gibt True zurück, wenn die Elemente in L
           aufsteigend sortiert sind'''
3      n = len(L)
4      if n < 2:
5          return True
6      for i in range(0, n-1):
7          if L[i] > L[i+1]:
8              return False
9      return True
10
11 LoL = [[0], [31], [-4, 3, 5, 7, 12, 29], [2, 6, 9, 15,
       11]]
12 for L in LoL:
13     print(L, is_sorted(L))
```

Aufgabe 5

Schreibe eine Python-Funktion `random_list(n, a, b)`, die eine Liste mit `n` Elementen zurückgibt, wobei jedes Element `r` eine ganze Zufallszahl mit $\texttt{a} \leq \texttt{r} \leq \texttt{b}$ ist.

Hinweis: Importiere dazu die Funktion `randint` aus dem `random`-Modul, die mit dem Aufruf `randint(a,b)` gleichverteilte ganze Zufallszahlen `r` mit $\texttt{a} \leq \texttt{r} \leq \texttt{b}$ zurückgibt.

Aufgabe 5

```
1  from random import randint
2
3  def random_list(n, a, b): # pythonic
4      '''Gibt eine Liste mit n zufälligen Zahlen a <= x <=
           b zurück'''
5      return [randint(a,b) for in range(n)]
6
7  def random_list_classic(n, a, b):
8      '''Gibt eine Liste mit n zufälligen Zahlen a <= x <=
           b zurück'''
9      L = []
10     for x in range(0, n):
11         L.append(x)
12     return L
13
14 print(random_list(5, 1, 9))
15 print(random_list_classic(1, 9))
```

Aufgabe 6

Schreibe eine Python-Funktion `gcd(a, b)`, welche den grössten gemeinsamen Teiler (*greatest common divisor*) der beiden ganzen Zahlen `a` und `b` berechnet und zurückgibt.

Aufgabe 6

```
1  def gcd(a, b):
2      '''Gibt den grössten gemeinsamen Teiler von a und b
           zurück'''
3      while  b != 0:
4          a, b = b, a % b
5      return a
6
7  print(gcd(24,15))
8  print(gcd(17, 0))
```

Aufgabe 7

Schreibe eine Python-Funktion `reverse_string(string)`, welche den String `string` in umgekehrter Zeichefolge zurückgibt.

Beispiel: `reverse_string('ABC')` $\Rightarrow$ `'CBA'`

Aufgabe 7

```
1  def reverse_string(string): # pythonic
2      '''Gibt die Zeichen von "string" in umgekehrter
           Reihenfolge zurück'''
3      return string[::-1]
4
5  def reverse_string_classic(string):
6      new_string = ''
7      n = len(string)
8      for i in range(0, n):
9          new_string += string[n-i-1]
10     return new_string
11
12 string = 'ABCDEFG'
13 print(reverse_string(string))
14 print(reverse_string_classic(string))
```

Aufgabe 8

Schreibe eine Python-Funktion `factorial(n)`, welche die Fakultät von n berechnet:

$$n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$$

Es muss nicht geprüft werden, ob `n` eine gültige Eingabe ist.

Beachte, dass $0! = 1$ gilt. Vielleicht hilft die folgende Grafik um einzusehen, dass dies sinnvoll ist.

$$\begin{aligned} 3! &= 3 \cdot 2 \cdot 1 = 6 \\ 2! &= 2 \cdot 1 = 2 \\ 1! &= 1 = 1 \\ 0! &= 1 \end{aligned}$$

: 3
: 2
: 1

Aufgabe 8

```
1  def factorial(n):
2      '''Gibt n! = n*(n-1)*...*2*1 zurück'''
3      fact = 1
4      for k in range(n):
5          fact *= k
6      return fact
```

Aufgabe 9

Schreibe eine Python-Funktion `count_letters(string)`, welche die Häufigkeit aller Zeichen in der Zeichenkette `string` bestimmt und als Python-Dictionary zurückgibt.

Aufgabe 9

```
1  def count_letters(string):
2      '''Gibt Dictionary mit Zeichenhäufigkeiten von
           'string' zurück'''
3      D = dict()
4      for char in string:
5          if char not in D:
6              D[char] = 1
7          else:
8              D[char] += 1
9      return D
10
11 print(count_letters('MISSISSIPPI'))
```

Aufgabe 10

Schreibe eine Python-Funktion `count_vowels(string)`, welche die Anzahl der Vokale (A,E,I,O,U) in der Zeichenkette `string` zählt und zurückgibt.

Beispiel: `count_vowels('Eremit')` $\Rightarrow$ 3

Aufgabe 10

```
1  def reverse_string(string): # pythonic
2      '''Gibt die Zeichen von "string" in umgekehrter
           Reihenfolge zurück'''
3      return string[::-1]
4
5  def reverse_string_classic(string):
6      new_string = ''
7      n = len(string)
8      for i in range(0, n):
9          new_string += string[n-i-1]
10     return new_string
11
12 string = 'ABCDEFG'
13 print(reverse_string(string))
14 print(reverse_string_classic(string))
```

Aufgabe 11

Schreibe eine Python-Funktion `count_words(string)`, welche die Zeichenkette `string` mit der Methode `string.split()` in seine Wörter zerlegt und danach die Häufigkeit der Wörter als Python-Dictionary zurückgibt.

Aufgabe 11

```
1  def count_words(string):
2      '''Gibt Dictionary mit der Häufigkeit aller Wörter in
           'string' zurück'''
3      string = string.lower()
4      array = string.split() # trennt an Leerzeichen und
           Zeilenschaltungen
5      D = dict()
6      for word in array:
7          if word not in D:
8              D[word] = 1
9          else:
10             D[word] += 1
11     return D
12
13 print(count_words('Das ist wie es ist'))
```

Aufgabe 12

(a) Schreibe eine Python-Funktion `sum_of_n(n)`, welche die Summe der ersten natürlichen Zahlen von 1 bis `n` in $O(n)$ berechnet und zurückgibt.

(b) Schreibe eine Python-Funktion `sum_of_n_fast(n)`, welche die Summe der ersten natürlichen Zahlen von 1 bis bis `n` in $O(1)$ berechnet und zurückgibt.

Aufgabe 12

```
1  def sum_of_n(n):
2      '''Gibt 1 + 2 + ... + n in O(n) zurück'''
3      return sum([i for i in range(1,n+1)])
4
5  def sum_of_n_fast(n):
6      '''Gibt 1 + 2 + ... + n in O(1) zurück'''
7      return n*(n+1)//2
```

Aufgabe 13

Schreibe eine Python-Funktion `is_palindrome(string)`, die `True` zurückgibt, wenn `string` ein Palindrom ist; d. h. wenn `string` vorwärts und rückwärts gelesen das gleiche Wort ergibt.

Aufgabe 13

```
1  def is_palindrome(string):
2      '''Gibt True zurück, wenn string ein Palindrom ist
           und False sonst.'''
3      return string == string[::-1] # pythonic!
4
5  def is_palindrome_classic(string):
6      '''Gibt True zurück, wenn string ein Palindrom ist
           und False sonst.'''
7      n = len(string)
8      for i in range(n):
9          if string[i] != string[n-i-1]:
10             return False
11     return True
12
13 s1 = 'ANNA'
14 s2 = 'BEN'
15 print(s1, is_palindrome(s1)) # => True
16 print(s2, is_palindrome(s2)) # => False
17
18 print(s1, is_palindrome_classic(s1)) # => True
```

Aufgabe 14

Schreibe eine Python-Funktion `zero_matrix(m, n)`, welche die Nullmatrix mit `m` Zeilen und `n` Spalten zurückgibt.

Beispiel: `zero_matrix(2, 3)` gibt als Wert `[[0, 0, 0], [0, 0, 0]]` zurück.

Aufgabe 14

```
1  def zero_matrix(m, n):
2      '''Gibt die Nullmatrix mit m Zeilen und n Spalten
           zurück'''
3      return [[0 for i in range(n)] for j in range(m)]
```

Aufgabe 15

Schreibe eine Python-Funktion `matrix_add(A, B)`, welche die Summe der Matrizen `A` und `B` mit gleicher Dimension berechnet und zurückgibt.

Beispiel:

```
P = [[8,6], [4,2]]
Q = [[1,2], [5,3]]
matrix_add(P,Q)      gibt [[9,8], [9,5]] zurück
```

Aufgabe 15

```
1  def matrix_add(A, B):
2      '''Gibt die Summe der Matrizen A und B zurück'''
3      m, n = len(A), len(A[0])
4      return [[A[i][j]+B[i][j] for j in range(n)] for i in
           range(m)]
5
6  A = [[1,2], [3,4], [5,6]]
7  B = [[3,1], [0,2], [4,1]]
8  C = matrix_sum(A, B)
9  print(C)
```

Aufgabe 16

Schreibe eine Python-Funktion `transpose_matrix(A)`, welche die Transponierte $\mathtt{A}^\mathsf{T}$ der Matrix `A` zurückgibt.

Hinweis: Die Transponierte A^T einer Matrix A ist die Matrix, bei der die i-te Zeilen von A zur j-ten Spalte von A^T wird.

Beispiel: $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad \Rightarrow \quad A^\mathsf{T} = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$

Aufgabe 16

```
1  def transpose_matrix(A):
2      '''Gibt die Transponierte A^T von A zurück'''
3      m, n = len(A), len(A[0])
4      return [[A[i][j] for i in range(m)] for j in range(n)]
5
6  A = [[1,2], [3,4], [5,6]]
7  B = transpose_matrix(A)
8  print(B)
```

Aufgabe 17

Schreibe eine Python-Funktion `write_integers(n, filename)`, welche die ganzen Zahlen $1, 2, \ldots, n$ zeilenweise in die Datei mit dem Namen `filename` schreibt.

Aufgabe 17

```
1  def write_integers(n, filename):
2      '''Schreibt die ersten n natürlichen Zahlen in eine
           Datei'''
3      fd = open(filename, mode='w')
4      for i in range(1, n+1):
5          fd.write(f'{i}\n')
6      fd.close()
7
8  write_integers(50, 'integers.txt')
```

Aufgabe 18

Schreibe eine Python-Funktion `add_floats_in_file(filename)`, welche die Summe der Zahlen in dieser Datei zurückgibt, wobei in jeder Zeile der Datei eine Gleitkommazahl steht.

Aufgabe 18

```
1  def add_numbers_of_file(filename):
2      '''Addiert alle Gleitkommazahlen in der Datei
           "filename".'''
3      fd = open(filename, mode='r')
4      s = 0
5      for line in fd:
6          s += float(line)
7      fd.close()
8      return s
9
10 s = add_numbers_of_file('messwerte.txt')
11 print(s)
```

Aufgabe 19

Schreibe eine Python-Funktion `fibonacchi(n)`, welche die Liste mit den ersten `n` Fibonacchi-Zahlen (1, 1, 2, 3, 5, 8, 13, ...) erzeugt und zurückgibt.

Aufgabe 19

```
1  def fibonacci(n):
2      '''Gibt die Liste der ersten n Fibonacci-Zahlen
           zurück'''
3      F = []
4      if n > 0:
5          F.append(1)
6      if n > 1:
7          F.append(1)
8      for i in range(2,n+1):
9          F.append(F[-2] + F[-1])
10     return F
11
12
13 for k in range(1, 7):
14     print(k, fibonacci(k))
```