Aufgabe 4.1

```
L = [13, 14, 2, 9, 16, 8, 7, 5]
print(L[7])
```

Aufgabe 4.2

```
L = [12, 7, 17, 5, 16, 3, 13, 2]
print(L[14 % 8])
```

Aufgabe 4.3

```
L = [18, 5, 10, 4, 12, 2, 11]
print(L[7])
```

Aufgabe 4.4

```
L = [11, 16, 4, 12, 7, 6, 10]
print(L[-6])
```

Aufgabe 4.5

```
L = [[7, 18, 16], [15, 14, 9, 5], [1, 17], [12, 2, 19]]
print(L[3][2])
```

Aufgabe 4.6

```
L = [6, 4, 3, 0, 2, 8, 1, 5, 7, 9]
print(L[L[L[0]]])
```

Aufgabe 4.7

```
L = [8, 5, 15, 3, 2, 6, 9, 19]
L[1] = L[2] + L[4]
print([8, 5, 15, 3, 2, 6, 9, 19])
```

Aufgabe 4.8

```
L = [10, 4, 11, 7, 13, 19, 9, 14]
print(L[2:7])
```

Aufgabe 4.9

```
L = [3, 13, 11, 8, 5, 19, 4, 14]
print(L[:2])
```

Aufgabe 4.10

```
L = [9, 16, 3, 17, 11, 13, 18, 14]
print(L[2:])
```

Aufgabe 4.11

```
A = [3, 18, 13, 9, 16]
B = A
B[1] = 22
print(A)
```

Aufgabe 4.12

```
A = [19, 14, 9, 15, 8]
B = A[:]
B[2] = 11
print(A)
```

Aufgabe 4.13

```
A = [16, 11, 7]
B = [8, 1, 9]
print(A + B)
```

Aufgabe 4.14

```
A = [8, 12]
print(2 * A)
```

Aufgabe 4.15

```
A = [14, 2, 4]
B = [8, 2]
print(A + 2 * B)
```

Aufgabe 4.16

```
A = [1, 4, 7, 18, 17, 2, 6, 14]
print(1 in A)
```

Aufgabe 4.17

```
A = [7, 6, 4, 16, 17, 12, 13, 1]
print(2 not in A)
```

Aufgabe 4.18

```
A = [4, 9, 3, 8, 2, 11, 5, 6, 7]
print(len(8*A))
```

Aufgabe 4.19

```
A = []
print(len(A))
```

Aufgabe 4.20

```
A = [2, 17, 15, 19, 10, 16]
A.sort()
print(A)
```

Aufgabe 4.21

```
A = [7, 9, 12, 16, 14, 10, 3, 17]
A.sort(reverse=True)
print(A)
```

Aufgabe 4.22

```
A = [7, 2, 6, 17, 16]
A.append(12)
print(A)
```

Aufgabe 4.23

```
A = [4, 15, 6, 1, 8, 11]
x = A.pop()
print(x)
print(A)
```

Aufgabe 4.24

```
A = [17, 10, 12, 13, 14, 11, 5]
y = A.pop(5)
print(y)
print(A)
```

Aufgabe 4.25

```
A = [15, 3, 11, 2, 9]
A.insert(1, 3)
print(A)
```

Aufgabe 4.26

```
A = [9, 5, 8, 16, 6]
A.append(A.pop(0))
print(A)
```

Aufgabe 4.27

```
L = [3, 2, 3, 2, 3, 3, 3, 1, 3]
print(L.count(2))
```

Aufgabe 4.28

```
L = [14, 3, 1, 7, 12]
print(sum(L))
```

Aufgabe 4.29

```
L = [6, 8, 18, 15, 7]
L.reverse()
print(L)
```

Aufgabe 4.30

```
L = [2, 13, 14, 19, 3]
for x in L:
    print(2*x)
```

Aufgabe 4.31

```
L = [3, 9, 6, 15, 18, 8, 16]
for x in L:
    if x > 10:
        break
    print(x)
```

Aufgabe 4.32

```
L = [1, 2, 16, 11, 5, 14, 8, 19]
for x in L:
    if x < 11:
        continue
    print(x)
```

Aufgabe 4.33

```
L = [18, 3, 5, 1]
for i in range(0, len(L)):
    print(i*L[i])
```

Aufgabe 4.34

```
L = ['g', 'k', 'm', 'f']
for i, x in enumerate(L):
    print(i, x)
```

Aufgabe 4.35

```
s = 0
for k in range(2,7):
    s += k
print(s)
```

Aufgabe 4.36

```
L = [1, 12, 14, 11]
```

```
for x in L:
    if x % 2 == 0:
        print('g')
    else:
        print('u')
```

Aufgabe 4.37

```
L = [4, 16, 2, 10, 11]
s = 0
for x in L:
    s += x
print(s)
```

Aufgabe 4.38

```
L = [9, 16, 6, 14, 10]
s = 0
for x in L:
    s += x
m = s / len(L)
print(m)
```

Aufgabe 4.39

Schreibe ein Programm `reverse_list.py`, das *ohne* die Methode `reverse()` alle Elemente einer gegebenen Liste `A` in umgekehrter Reihenfolge in der zu Beginn leeren Liste `B` speichert. Danach sind die Listen `A` und `B` auszugeben.

Aufgabe 4.40

Schreibe ein Programm `find_max.py`, das *ohne* die Funktion `max()` das Maxium `m` der Werte in einer gegebenen Liste `L` findet und am Schluss dieses Maximum und die Liste auf der Shell ausgibt.

Aufgabe 4.41

Schreibe ein Programm `number_to_list.py`, das den Benutzer so lange auffordert einen Zahlstring einzugeben, bis die Eingabe der leere String ist.

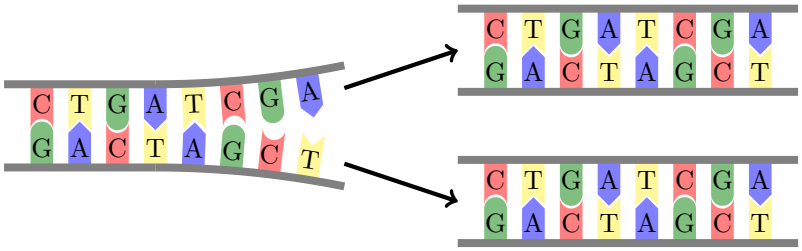
Jede eingegebenen Zahl soll ans Ende einer Liste `L` anfgefügt werden, die zu Beginn leer ist. Nach der letzten Eingabe soll die Liste ausgegeben werden.

Aufgabe 4.42

Schreibe ein Programm mit dem Namen `mean.py`, das den Mittelwert einer gegebenen Liste `L` von Zahlen berechnet und ausgibt.

Aufgabe 4.44

Die *DNA-Replikation* dient dazu, die Erbinformation einer Zelle oder eines Virus zu vervielfältigen. Dabei werden die beiden komplementären DNA-Stränge, auf denen die Erbinformation durch die Abfolge der Nukleotide *Adenin (A)*, *Cytosin (C)*, *Guanin (G)* und *Thymin (T)* codiert ist, durch biochemische Prozesse aufgetrennt und durch Komplementbildung wieder vervollständigt.



Beachte, dass nur die Nukleotidpaare Cytosin/Guanin sowie Adenin/Thymin jeweils komplementär sind (zueinander passen).

Schreibe ein Programm `dna_complement.py`, das aus einer Liste `N` mit Nukleotiden (`'A'`, `'C'`, `'G'`, `'T'`) eine Liste `K` mit den jeweiligen komplementären Nukleotiden produziert. *Beispiel:*

```
N = ['A', 'C', 'C', 'A', 'T', 'G', 'T']
```

⇒

```
K = ['T', 'G', 'G', 'T', 'A', 'C', 'A']
```

Aufgabe 4.45

Schreibe mit Hilfe der folgenden Anleitung ein Programm `eratosthenes.py`, das mit dem *Siebes des Eratosthenes* die Primzahlen von 2 bis und mit $N = 10^7$ ausgibt.

1. Erzeuge eine Liste `P` der Länge $N+1$, in der jedes Element den Wert `True` hat. Dies bedeutet, dass alle Indizes von 0 bis N zunächst Primzahlen sind.

0	1	2	3	4	5	6	7	8	9	···	N
T	T	T	T	T	T	T	T	T	T	···	T

2. Weise den Elementen von `P` mit den Indizes 0 und 1 den Wert `False` zu, da 0 und 1 keine Primzahlen sind.

0	1	2	3	4	5	6	7	8	9	···	N
F	F	T	T	T	T	T	T	T	T	···	T

3. Durchlaufe die Liste `P` von $k = 2$ bis und mit N. Falls die Liste an der Position k den Wert `True` hat, handelt es sich um eine Primzahl und k ist auszugeben. Danach ist die Variable `i` mit `i=2` zu initialisieren und in einer inneren Schleife den Elementen von `P` an den Positionen `2*k`, `3*k`, ..., `i*k` der Wert `False` zuzuweisen, so lange `i*k < N+1` gilt.

0	1	2	3	4	5	6	7	8	9	···	N
F	F	T	T	F	T	F	T	F	T	···	?

   Gib 2 aus und weise allen folgenden Vielfachen von 2 den Wert `False` zu.

0	1	2	3	4	5	6	7	8	9	···	N
F	F	T	T	F	T	F	T	F	F	···	?

   Gib 3 aus und weise allen folgenden Vielfachen von 3 den Wert `False` zu.

   Auf diese Weise wird jeder Index `k`, dessen Wert nicht schon in einem früheren Durchlauf auf `False` gesetzt wurde als Primzahl erkannt und ausgegeben. Danach sorgt der Algorithmus dafür, dass alle Vielfachen von `k` (`2*k`, `3*k`, ..., `i*k`, ...), die kleiner als $N + 1$ sind, von der Liste potenzieller Primzahlen „gestrichen“ werden.

Zusatzaufgabe: Erkundige dich im Internet, mit welcher einfachen Änderung am Code die Laufzeit des Algorithmus verkürzt werden kann.