1. (4 Punkte) Gegeben ist `s = 'Informatik'`. Gib an, welchen Wert die folgenden Ausdrücke haben.

```
s[2:5]
s[-4:]
s[::3]
s[::-1][:3]
s.upper().count('I')
s.lower().replace('i', '!')
s.find('mat')
len(s[1:-1])
```

Lösung:

`'for'`	Index 2 bis 4
`'atik'`	die letzten 4 Zeichen
`'Ioak'`	Index 0, 3, 6, 9
`'kit'`	rückwärts 'kitamrofnI', davon die ersten 3 Zeichen
2	'INFORMATIK' enthält zweimal 'I'
`'!nformat!k'`	nach lower() sind beide i klein
5	'mat' beginnt an Index 5
8	ohne das erste und das letzte Zeichen

2. (3 Punkte) Gegeben ist der folgende Code:

```
t = 'Mississippi'
print(t.replace('ss', 's').upper()[1:6].count('S'))
```

Gib die Zwischenergebnisse der einzelnen Schritte und die Ausgabe an.

Lösung:

`t.replace('ss', 's')`	`'Misisippi'`
`.upper()`	`'MISISIPPI'`
`[1:6]`	`'ISISI'`
`.count('S')`	2

Ausgabe: 2

3. (4 Punkte) Das folgende Programm soll aus `'Habicht'` den String `'KABICHT'` machen.

```
1 s = 'Habicht'
2 s[0] = 'K'
3 s.upper()
4 print(s)
```

   (a) Zeile 2 führt zu einem Fehler. Erläutere, warum.

   (b) Auch ohne Zeile 2 würde `Habicht` ausgegeben und nicht `HABICHT`. Erläutere, warum.

   (c) Gib ein korrigiertes Programm an.

Lösung:

(a) Strings sind unveränderlich. Man kann einzelne Zeichen nicht durch eine Zuweisung ändern (`TypeError`).

(b) `upper()` verändert den String nicht, sondern liefert einen neuen String. Dieser wird nicht gespeichert.

(c)
```
s = 'Habicht'
s = 'K' + s[1:]
s = s.upper()
print(s)
```

4. (4 Punkte) Schreibe eine Funktion `abkuerzung(text)`, die aus den Anfangsbuchstaben der Wörter von `text` eine Abkürzung in Großbuchstaben bildet und zurückgibt. Die Wörter sind durch Leerzeichen getrennt.

   Beispiele: `abkuerzung('portable network graphics')` liefert `'PNG'`,
   `abkuerzung('Hypertext Transfer Protocol')` liefert `'HTP'`.

Lösung:

```
def abkuerzung(text):
    ergebnis = ''
    for wort in text.split():
        ergebnis = ergebnis + wort[0].upper()
    return ergebnis
```

5. (3 Punkte) Ein Palindrom ist ein Wort, das vorwärts und rückwärts gelesen gleich ist, z.B. `'Rentner'` oder `'Anna'`. Groß- und Kleinschreibung soll keine Rolle spielen.

   Schreibe eine Funktion `ist_palindrom(wort)`, die `True` zurückgibt, wenn `wort` ein Palindrom ist, sonst `False`.

Lösung:

```
def ist_palindrom(wort):
    w = wort.lower()
    return w == w[::-1]
```

6. (4 Punkte) Gib an, wie die Liste `a` nach jeder Zeile aussieht. Gib in Zeile 4 auch den Wert von `x` an.

```
a = [5, 3, 8]          # Zeile 1
a.append(1)            # Zeile 2
a.insert(0, 9)         # Zeile 3
x = a.pop(2)           # Zeile 4
a.remove(8)            # Zeile 5
a[1] = a[1] * 2        # Zeile 6
a.sort()               # Zeile 7
```

Lösung:

```
Zeile 1: [5, 3, 8]
Zeile 2: [5, 3, 8, 1]
Zeile 3: [9, 5, 3, 8, 1]
Zeile 4: [9, 5, 8, 1]      x = 3 (das Element an Index 2)
Zeile 5: [9, 5, 1]
Zeile 6: [9, 10, 1]
Zeile 7: [1, 9, 10]
```

7. (4 Punkte) Gegeben ist der folgende Code:

```
a = [4, 2, 7]
b = sorted(a)
c = a.sort()
print(a, b, c)

namen = ['Zebra', 'affe', 'Apfel', 'Ähre', 'Biber']
print(sorted(namen))
```

(a) Gib die Ausgabe der ersten `print`-Anweisung an und erläutere den Unterschied zwischen `sorted(a)` und `a.sort()`.

(b) Gib die Ausgabe der zweiten `print`-Anweisung an und begründe die Reihenfolge.

Lösung:

(a) `[2, 4, 7] [2, 4, 7] None`
`sorted(a)` liefert eine neue, sortierte Liste und lässt `a` unverändert. `a.sort()` sortiert die Liste `a` selbst und liefert nichts zurück (`None`). Nach Zeile 3 ist deshalb auch `a` sortiert.

(b) `['Apfel', 'Biber', 'Zebra', 'affe', 'Ähre']`
Strings werden Zeichen für Zeichen nach den Zahlen von `ord()` verglichen. Großbuchstaben haben kleinere Zahlen als Kleinbuchstaben, und Umlaute haben größere Zahlen als alle Buchstaben von a bis z.

8. (5 Punkte) Eine Messstation speichert ihre Messwerte in einer Zeile, getrennt durch Semikolons: `zeile = '12;7;30;5'`
   Schreibe Python-Anweisungen, die

   - die Zeile in die einzelnen Werte zerlegen und daraus eine Liste `werte` mit ganzen Zahlen erzeugen,
   - die Summe und den größten Wert ausgeben,
   - die Werte mit `' + '` verbunden ausgeben, also `12 + 7 + 30 + 5`.

Lösung:

```
zeile = '12;7;30;5'
teile = zeile.split(';')       # ['12', '7', '30', '5']
werte = []
for t in teile:
    werte.append(int(t))
print(sum(werte), max(werte))      # 54 30

texte = []
for w in werte:
    texte.append(str(w))
print(' + '.join(texte))
```

(Für die letzte Ausgabe geht auch `print(' + '.join(teile))`, denn `teile` enthält schon Strings.)

9. (5 Punkte) Schreibe eine Funktion `laengstes_wort(satz)`, die das längste Wort eines Satzes zurückgibt. Die Wörter sind durch Leerzeichen getrennt. Gibt es mehrere längste Wörter, soll das erste zurückgegeben werden.

   Beispiel: `laengstes_wort('Heute lernen wir viele neue Methoden kennen')` liefert `'Methoden'`.

Lösung:

```
def laengstes_wort(satz):
    best = ''
    for wort in satz.split():
        if len(wort) > len(best):
            best = wort
    return best
```