Mögliche Lösungen zu den Aufgaben aus Block 1

Aufgabenblock 1: Grundlagen

1.1. Kreiert eine Liste mit den Zahlen von 1 bis 10. Speichert diese Liste unter dem Namen 'l_1' und lasst Python die Liste ausgeben (mit Hilfe der `print` Funktion).

```
l_1 = list(range(1,11))
print(l_1)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

1.2. Extrahiert das das dritte Element aus dieser Liste, assoziiert es in einer Variable und gebt diese aus.

```
l_1_e3 = l_1[2]
print(l_1_e3)
```

```
3
```

1.3. Extrahiert die letzten drei Elemente der Liste und gebt sie aus.

```
print(l_1[-3:])
```

```
[8, 9, 10]
```

1.4. Ersetzt den ersten Wert der Liste mit der Zahl 99.9.

```
l_1[0] = 99.9
```

1.5. Sortiert die Elemente der Liste in absteigender Reihenfolge. Gebt sie danach über den `print` Befehl aus.

```
l_1.sort(reverse=True)
print(l_1)
```

```
[99.9, 10, 9, 8, 7, 6, 5, 4, 3, 2]
```

1.6. Definiert die folgenden zwei Mengen:

$$m_1 = \{1, 4, 23, 95, 12\}$$
$$m_2 = \{0, 23, 80, 96, 95\}$$

```
m_1 = {1, 4, 23, 95, 12}
m_2 = {0, 23, 80, 96, 95}
```

1.7. Welche Elemente sind in m_1, aber nicht in m_2? Speichert diese Elemente *in einer Liste* und gebt diese aus.

```
l_diff = list(m_1 - m_2)
print(l_diff)
```

```
[1, 4, 12]
```

1.8. Speichert die Schnittmenge von m_1 und m_2 über eine Variable und lasst Python diese Variable ausgeben.

```
m_intersec = m_1 & m_2
print(m_intersec)
```

```
{95, 23}
```

1.9. Kreiert ein Wörterbuch mit den folgenden *key-value* Paaren:

- “Hello” and “Hola”
- 5 and 120.5
- “bla” and [10, 80]

Ruft dann den zu ‘bla’ gehörenden value auf.

```
dic_1 = {"Hello" : "Hola", 5 : 120.5, "bla" : [10, 80]}
dic_1["bla"]
```

```
[10, 80]
```

Aufgabenblock 2: Funktionen

2.1. Definiert eine Funktion, die folgende Gleichung berechnet:

$$f(x, y) = 10x + (1 - y)^2$$

```
def func_expl(x,y):
    result = 10 * x + (1-y)**2
    return result

func_expl(2,1)
```

```
20
```

2.2. Ergänzt die Funktion, sodass sie überprüft ob als Inputs nur ganze Zahlen eingegeben wurden (int). Wenn ein Input keine ganze Zahl ist soll eine Fehlermeldung ausgegeben werden.

```
def func_expl(x,y):
    assert isinstance(x, int), "Input x kein integer, sonder {}".format(type(x))
    assert isinstance(y, int), "Input y kein integer, sonder {}".format(type(y))
    result = 10 * x + (1-y)**2
    return result

func_expl(2,3.0)
```

```
AssertionError: Input y kein integer, sonder <class 'float'>
```

Aufgabenblock 3: Loops

3.1. Erstelle eine Liste mit den Wurzeln der Zahlen zwischen 5 und 15. Stelle die Lösung sowohl als `for loop` also auch als `list comprehension` dar.

```
l_loop = []
for i in range(5,16):
    l_loop.append(i**0.5)

l_comp = [x**0.5 for x in range(5, 16)]

print(l_loop)
print(l_comp)
```

```
[2.23606797749979, 2.449489742783178, 2.6457513110645907, 2.8284271247461903, 3.0, 3.1622776
[2.23606797749979, 2.449489742783178, 2.6457513110645907, 2.8284271247461903, 3.0, 3.1622776
```

3.2. Betrachte den folgenden `for loop`:

```
l_1 = [0, 2, "Land", "!"]
l_2 = ["Lame", "La", 1, 3]
for i in l_1:
    print(i * l_2[i], end=" ")
```

```
2
```

```
TypeError: list indices must be integers or slices, not str
```

Dieser loop funktioniert so nicht! Schreibe den loop so um, dass er nicht über die einzelnen Elemente von `l_1`, sondern über die Indices von `l_1` iteriert.

Gewünschtes Ergebnis: "LaLa Land!!!

```
l_1 = [0, 2, "Land", "!"]
l_2 = ["Lame", "La", 1, 3]
for i in range(len(l_1)):
    print(l_1[i] * l_2[i], end=" ")
```

```
LaLa Land !!!
```

3.3. Wie oft muss man 1.1 quadrieren bis das Ergebnis größer als 10 ist? Verwende einen `while loop` um diese Frage zu beantworten.

```
counter = 0
current_val = 1.1
while current_val <= 10:
    current_val = current_val**2
    counter += 1
print(current_val)
print(counter)
```

```
21.1137767453526
5
```

3.4. Definiert eine Funktion, welche die folgende Gleichung implementiert:

$$f(x) = \frac{1}{x} + \frac{1}{x^2}$$

Nehmt als Startwert $x_0 = 2$ und berechnet die Zeitreihe, welche durch diese Funktion für 7 Zeitschritte kreiert.

Erläuterung: Ihr könnt die von euch definierte Funktion in einen `for loop` einbauen, sodass sie in jedem Zeitschritt ihren Output aus dem Zeitschritt davor als Input erhält.

```
def func_ex_2(x):
    result = (1/x) + 1/x**2
    return result

ts = [2]
for t in range(7):
    ts.append(func_ex_2(ts[-1]))
ts
```

```
[2,
 0.75,
 3.1111111111111107,
 0.42474489795918374,
 7.897338780221661,
 0.1426588076214919,
 56.14607377836738,
 0.018127904872052552]
```

Hier nur als Illustration wie die Zeitreihe aussieht:

```
import matplotlib.pyplot as plt
plt.plot(range(8), ts)
```