

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 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, 3.1622776601683795, 3.3166247903554, 3.5, 3.605551275463989, 3.7797631000000002, 3.960784265005686, 4.16191254101746, 4.381780495308837, 4.617812101018222, 4.868413821546594, 5.131672236627107, 5.40832591417045, 5.697529437454273, 5.998016047537069, 6.30957344480193, 6.632244908493534, 6.966065957744684, 7.311188824514023, 7.6675213049360485, 8.035087610317549, 8.412883389073379, 8.800000000000001, 9.196431424501679, 9.60197617036057, 10.016742116159978, 10.440549450595578, 10.883285304264984, 11.334861471812185, 11.7952966251692, 12.264575131106459, 12.74267015516155, 13.229563051977869, 13.7252982336467, 14.229856476075072, 14.74322981513196, 15.264415887701708, 15.79341270189781, 16.3302153919173, 16.874824544122658, 17.427249998998703, 17.987491535925498, 18.555555555555557, 19.13144672151812, 19.715165751877647, 20.306722638051957, 20.906118880000002, 21.513352941521473, 22.12842261131691, 22.751327160496875, 23.38206679885977, 24.02064068811967, 24.667048939999998, 25.321291640625, 25.98336881198619, 26.653270560000002, 27.331006904499998, 28.016578750000002, 28.709985050917708, 29.41122578125, 30.120302734375, 30.837216796875, 31.56206787109375, 32.294856000000004, 33.0355712890625, 33.7842236328125, 34.5408154296875, 35.3053466796875, 36.07791748046875, 36.85852783203125, 37.647177734375, 38.4438681640625, 39.24859912109375, 40.06137158203125, 40.8821865234375, 41.7110450234375, 42.54795703125, 43.3929228515625, 44.24594248046875, 45.1060159375, 45.9731433203125, 46.847324609375, 47.728560890625, 48.6168521484375, 49.51220838125, 50.414629590625, 51.32411578125, 52.240666953125, 53.164283109375, 54.0949642578125, 55.032710390625, 55.977521609375, 56.92939796875, 57.8883395703125, 58.85434640625, 59.827418578125, 60.807556084375, 61.7947589375, 62.789027140625, 63.79036071875, 64.79876068125, 65.81422703125, 66.836760765625, 67.866361890625, 68.903030515625, 69.946767640625, 70.997573265625, 72.055447390625, 73.120389915625, 74.192391840625, 75.271453165625, 76.357573890625, 77.450754015625, 78.551093640625, 79.658592765625, 80.773251390625, 81.895069515625, 83.024047140625, 84.160284265625, 85.303780890625, 86.454537015625, 87.612552640625, 88.777827765625, 89.950362390625, 91.130156515625, 92.317209140625, 93.511520265625, 94.713089890625, 95.921918015625, 97.138004640625, 98.361349765625, 99.591953390625, 100.829815515625, 102.074936140625, 103.327314265625, 104.586950890625, 105.853846015625, 107.127999640625, 108.409411765625, 109.698082390625, 110.993911515625, 112.296899140625, 113.607045265625, 114.924349890625, 116.248813015625, 117.580434640625, 118.919214765625, 120.265153390625, 121.618250515625, 122.978506140625, 124.345920265625, 125.720492890625, 127.092224015625, 128.471113640625, 129.857161765625, 131.250368390625, 132.650733515625, 134.058257140625, 135.472939265625, 136.894780890625, 138.323782015625, 139.759942640625, 141.203262765625, 142.653742390625, 144.111381515625, 145.576180140625, 147.048138265625, 148.527255890625, 150.013533015625, 151.506969640625, 153.007565765625, 154.515321390625, 156.030236515625, 157.552311140625, 159.081545265625, 160.617938890625, 162.161492015625, 163.712204640625, 165.269976765625, 166.834808390625, 168.406709515625, 170.005680140625, 171.611720265625, 173.224830890625, 174.844912015625, 176.472063640625, 178.106285765625, 179.747578390625, 181.395941515625, 183.051375140625, 184.713879265625, 186.383453890625, 188.060099015625, 189.743814640625, 191.434590765625, 193.132427390625, 194.837324515625, 196.549282140625, 198.268299265625, 200.004375890625, 201.747512015625, 203.497707640625, 205.254962765625, 207.019277390625, 208.790651515625, 210.569085140625, 212.354578265625, 214.147130890625, 215.946743015625, 217.753414640625, 219.567145765625, 221.387936390625, 223.215786515625, 225.050696140625, 226.892665265625, 228.741693890625, 230.597782015625, 232.460929640625, 234.331136765625, 236.208403390625, 238.092729515625, 240.004115140625, 241.932560265625, 243.868064890625, 245.810629015625, 247.760252640625, 249.716935765625, 251.680678390625, 253.651480515625, 255.629342140625, 257.614263265625, 259.606243890625, 261.605284015625, 263.611383640625, 265.624542765625, 267.644761390625, 269.671939515625, 271.706077140625, 273.747174265625, 275.795230890625, 277.850247015625, 279.912222640625, 281.981157765625, 284.057052390625, 286.139906515625, 288.229720140625, 290.326493265625, 292.430225890625, 294.540918015625, 296.658569640625, 298.783180765625, 300.914751390625, 303.053281515625, 305.198771140625, 307.351220265625, 309.510628890625, 311.676997015625, 313.850324640625, 316.030611765625, 318.217858390625, 320.412064515625, 322.613230140625, 324.821355265625, 327.036439890625, 329.258484015625, 331.487487640625, 333.723450765625, 335.966373390625, 338.216255515625, 340.473097140625, 342.736898265625, 345.007658890625, 347.285379015625, 349.569958640625, 351.861497765625, 354.159996390625, 356.465454515625, 358.777872140625, 361.097249265625, 363.423585890625, 365.756882015625, 368.097137640625, 370.444352765625, 372.798527390625, 375.159661515625, 377.527755140625, 379.902808265625, 382.284820890625, 384.673793015625, 387.069724640625, 389.472615765625, 391.882466390625, 394.299276515625, 396.723046140625, 399.153775265625, 401.591463890625, 404.036112015625, 406.487720640625, 408.946289765625, 411.411819390625, 413.884309515625, 416.363759140625, 418.850168265625, 421.343536890625, 423.843865015625, 426.351152640625, 428.865399765625, 431.386606390625, 433.914772515625, 436.449898140625, 438.991983265625, 441.541027890625, 444.097032015625, 446.660095640625, 449.230218765625, 451.807391390625, 454.391623515625, 456.982915140625, 459.581266265625, 462.186676890625, 464.799147015625, 467.418676640625, 470.045265765625, 472.678914390625, 475.319622515625, 477.967390140625, 480.622217265625, 483.284103890625, 485.953050015625, 488.629055640625, 491.312120765625, 493.992245390625, 496.679429515625, 499.373673140625, 502.074976265625, 504.783339890625, 507.498764015625, 510.221248640625, 512.950793765625, 515.687409390625, 518.431095515625, 521.181852140625, 523.939679265625, 526.704576890625, 529.476545015625, 532.255583640625, 535.041692765625, 537.834872390625, 540.635122515625, 543.442443140625, 546.256834265625, 549.078295890625, 551.906828015625, 554.742431640625, 557.585105765625, 560.434850390625, 563.291675515625, 566.155581140625, 569.026567265625, 571.904633890625, 574.789781015625, 577.681908640625, 580.581016765625, 583.487105390625, 586.400175515625, 589.320327140625, 592.247560265625, 595.181874890625, 598.123271015625, 601.071748640625, 604.027307765625, 606.990948390625, 609.962670515625, 612.941474140625, 615.927359265625, 618.920325890625, 621.920374015625, 624.927503640625, 627.941714765625, 630.963007390625, 633.991381515625, 637.026837140625, 640.069374265625, 643.119002890625, 646.175723015625, 649.239534640625, 652.310436765625, 655.388429390625, 658.473512515625, 661.565686140625, 664.664950265625, 667.771304890625, 670.884750015625, 673.905285640625, 676.932911765625, 679.967628390625, 683.009435515625, 686.058333140625, 689.114321265625, 692.177409890625, 695.247599015625, 698.324888640625, 701.409278765625, 704.500769390625, 707.599360515625, 710.705052140625, 713.817844265625, 716.937736890625, 720.064730015625, 723.198823640625, 726.339917765625, 729.488012390625, 732.643107515625, 735.805203140625, 738.974309265625, 742.150425890625, 745.333553015625, 748.523690640625, 751.720838765625, 754.924997390625, 758.136166515625, 761.354346140625, 764.579536265625, 767.811736890625, 771.050948015625, 774.297169640625, 777.550391765625, 780.810624390625, 784.077867515625, 787.352121140625, 790.633385265625, 793.920659890625, 797.214945015625, 800.516240640625, 803.824546765625, 807.139863390625, 810.462190515625, 813.791528140625, 817.127876265625, 820.471234890625, 823.821604015625, 827.178983640625, 830.543373765625, 833.914774390625, 837.293185515625, 840.678607140625, 844.071039265625, 847.470481890625, 850.876935015625, 854.290408640625, 857.710892765625, 861.138387390625, 864.572892515625, 868.014408140625, 871.462934265625, 874.918470890625, 878.381018015625, 881.850575640625, 885.327143765625, 888.810722390625, 892.301311515625, 895.800911140625, 899.308521265625, 902.824141890625, 906.347773015625, 909.879414640625, 913.419066765625, 916.966729390625, 920.522402515625, 924.086086140625, 927.657779265625, 931.237482890625, 934.825197015625, 938.420921640625, 942.024656765625, 945.636402390625, 949.256158515625, 952.883925140625, 956.519702265625, 960.163489890625, 963.815288015625, 967.475096640625, 971.142915765625, 974.818745390625, 978.502585515625, 982.194436140625, 985.894297265625, 989.602168890625, 993.318051015625, 997.041944640625, 1000.773849765625, 1004.513766390625, 1008.261694515625, 1012.017634140625, 1015.781585265625, 1019.553547890625, 1023.333522015625, 1027.121507640625, 1030.917504765625, 1034.721513390625, 1038.533533515625, 1042.353565140625, 1046.181608265625, 1050.017662890625, 1053.861729015625, 1057.713806640625, 1061.573895765625, 1065.441996390625, 1069.318108515625, 1073.202232140625, 1077.094367265625, 1080.994513890625, 1084.902672015625, 1088.818841640625, 1092.743022765625, 1096.675215390625, 1100.615419515625, 1104.563635140625, 1108.519862265625, 1112.484100890625, 1116.456351015625, 1120.436612640625, 1124.424885765625, 1128.421170390625, 1132.425466515625, 1136.437774140625, 1140.458093265625, 1144.486423890625, 1148.522766015625, 1152.567119640625, 1156.619484765625, 1160.679861390625, 1164.748249515625, 1168.824649140625, 1172.909060265625, 1176.991482890625, 1181.081917015625, 1185.180362640625, 1189.286819765625, 1193.401288390625, 1197.523768515625, 1201.654250140625, 1205.792733265625, 1209.939217890625, 1214.093704015625, 1218.256191640625, 1222.426680765625, 1226.605171390625, 1230.791663515625, 1234.986157140625, 1239.188652265625, 1243.399148890625, 1247.617647015625, 1251.844146640625, 1256.078647765625, 1260.321150390625, 1264.571654515625, 1268.830160140625, 1273.096667265625, 1277.371175890625, 1281.653686015625, 1285.944197640625, 1290.242710765625, 1294.549225390625, 1298.863741515625, 1303.186259140625, 1307.516778265625, 1311.855298890625, 1316.201821015625, 1320.556344640625, 1324.918869765625, 1329.289396390625, 1333.667924515625, 1338.054454140625, 1342.448985265625, 1346.851516890625, 1351.262049015625, 1355.680582640625, 1360.107117765625, 1364.541654390625, 1368.984192515625, 1373.434732140625, 1377.893273265625, 1382.359815890625, 1386.834359015625, 1391.316902640625, 1395.807446765625, 1400.306091390625, 1404.812836515625, 1409.327682140625, 1413.850628265625, 1418.381674890625, 1422.920822015625, 1427.468069640625, 1432.023417765625, 1436.586866390625, 1441.158415515625, 1445.738065140625, 1450.325815265625, 1454.921665890625, 1459.525617015625, 1464.137668640625, 1468.757820765625, 1473.386073390625, 1478.022426515625, 1482.666880140625, 1487.319434265625, 1491.980088890625, 1496.648844015625, 1501.325709640625, 1506.010685765625, 1510.703772390625, 1515.404969515625, 1520.114277140625, 1524.831695265625, 1529.557223890625, 1534.290863015625, 1539.032612640625, 1543.782472765625, 1548.540443390625, 1553.306524515625, 1558.080716140625, 1562.863018265625, 1567.653431890625, 1572.451947015625, 1577.258563640625, 1582.073281765625, 1586.896101390625, 1591.727022515625, 1596.566045140625, 1601.413169265625, 1606.268394890625, 1611.131722015625, 1616.003150640625, 1620.882680765625, 1625.770312390625, 1630.666045515625, 1635.569879140625, 1640.481813265625, 1645.401847890625, 1650.329983015625, 1655.266218640625, 1660.210554765625, 1665.162991390625, 1670.123528515625, 1675.092166140625, 1680.068904265625, 1685.053742890625, 1690.046682015625, 1695.047721640625, 1700.056861765625, 1705.074102390625, 1710.099443515625, 1715.132885140625, 1720.174427265625, 1725.224070890625, 1730.281816015625, 1735.347662640625, 1740.421610765625, 1745.503660390625, 1750.593811515625, 1755.
```

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)
```

