

# 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, 3.674231616129343, 3.87298334621, 4.123105625617661, 4.38178345048047, 4.64, 4.909077747051215, 5.196152422706632, 5.385164807134504, 5.590663526645728, 5.803761611785413, 6.022505624971978, 6.247020717173179, 6.477225575051661, 6.713111231958997, 6.954691653786461, 7.201961151979899, 7.45483397735467, 7.713311263341394, 7.977302751891159, 8.24680864191109, 8.52182894241099, 8.80236265617063, 9.088419733224915, 9.38000027061825, 9.677115107878583, 9.979764151782542, 10.28794728612123, 10.59166415191729, 10.90091444121658, 11.21570815328057, 11.53604520725999, 11.86192561126261, 12.19334926419917, 12.5303171545461, 12.87282925197422, 13.22088556211971, 13.57448607722329, 13.93363179727265, 14.2983227124669, 14.66855982271693, 15.04434312721526, 15.42567272629167, 15.81254961032737, 16.20497477913526, 16.60294833571417, 17.00647039016419, 17.41555094248433, 17.83018909267458, 18.25039484073493, 18.67616828665537, 19.10750943142591, 19.54441826504655, 19.98689474751729, 20.43493897883813, 20.88855095900907, 21.34773078898011, 21.81247856875125, 22.28279429821249, 22.75867797726393, 23.24012970580457, 23.72715048383541, 24.21973031225645, 24.71786929106759, 25.22156742026883, 25.73082470985027, 26.24564115981191, 26.76601687015365, 27.29195184087549, 27.82344607207743, 28.36049955375937, 28.90311228592141, 29.45128426856355, 30.00501559167569, 30.56430625524783, 31.12915625927997, 31.70056571386211, 32.27853461900425, 32.86306297470539, 33.45415088096653, 34.05179843803787, 34.65590564611931, 35.26657250513085, 35.88369911519249, 36.50728547621423, 37.13743170720937, 37.77413789821631, 38.41740404923515, 39.06723036026579, 39.72361673130743, 40.38656326236017, 41.05607005342391, 41.73213710449875, 42.41476441558459, 43.10395208668153, 43.80069911768957, 44.50499550860871, 45.21684125943895, 45.93623627028029, 46.66318055113283, 47.39767410199747, 48.13971691287421, 48.88930898376205, 49.64645031466099, 50.41114100557113, 51.18338105649247, 51.96317046742511, 52.75050923836895, 53.54539736932399, 54.34783485029023, 55.15782168125767, 55.97535786222531, 56.79944339319315, 57.63007827416119, 58.46726246512943, 59.31100595609787, 60.16129874706651, 61.01814083803535, 61.88153222900439, 62.75147292097363, 63.62796190294307, 64.51109927491271, 65.40088403688265, 66.29731618885289, 67.20040574082343, 68.11015279279427, 69.02655734476541, 69.94962049673685, 70.87935224870849, 71.81575270068033, 72.75882185265237, 73.70855970462461, 74.66496635659715, 75.62804180856989, 76.59778616054283, 77.57419941251607, 78.55728166448951, 79.54704291646315, 80.54348326843699, 81.54660272041103, 82.55640127238527, 83.57287892435971, 84.59603567633435, 85.62587252830919, 86.66238948028423, 87.70558653225947, 88.75546368423491, 89.81202093621055, 90.87525838818629, 91.94517594016213, 93.02177359213807, 94.10505134411411, 95.19500929608925, 96.29164734806449, 97.39496550003983, 98.50496375201527, 99.62164210399081, 100.74500065596635, 101.87503940794199, 103.01175835991763, 104.15515741189327, 105.30523656386891, 106.46199571584455, 107.62543506782019, 108.79555451979583, 109.97235417177147, 111.15583392374711, 112.34579377572275, 113.54233362769839, 114.74545347967403, 115.95515333164967, 117.17143318362531, 118.39429303560095, 119.62373288757659, 120.85975273955223, 122.10235259152787, 123.35153244350351, 124.60729229547915, 125.86963214745479, 127.13855200943043, 128.41405186140607, 129.69613171338171, 130.98479156535735, 132.27993141733299, 133.58165126930863, 134.88985112128427, 136.20453097325991, 137.52569082523555, 138.85333067721119, 140.18745052918683, 141.52805038116247, 142.87513023313811, 144.22869008511375, 145.58872993708939, 146.95524978906503, 148.32824964104067, 149.70772949301631, 151.09368934499195, 152.48612919696759, 153.88504904894323, 155.29043890091887, 156.70239875289451, 158.12092860487015, 159.54602845684579, 160.97769830881643, 162.41493816079207, 163.85874801276771, 165.30902786474335, 166.76587771671899, 168.22929756869463, 169.69928742067027, 171.17574727264591, 172.65867712462155, 174.14807697659719, 175.64394682857283, 177.14628668054847, 178.65509653252411, 180.16937638450475, 181.68912623648039, 183.21434608845603, 184.74503594043167, 186.28129579240731, 187.82312654438295, 189.37052729635859, 190.92349794833423, 192.48203770030987, 194.04614755228551, 195.61582740426115, 197.19107725623679, 198.77189710821243, 200.35828696018807, 201.95024681216371, 203.54767666413935, 205.15057651611499, 206.75894636809063, 208.37268622006627, 210.00179607204191, 211.63527592401755, 213.27423577599319, 214.91857562796883, 216.56829547994447, 218.22339533192011, 219.88387518389575, 221.54973503587139, 223.22197488784703, 224.90059473982267, 226.58569459179831, 228.27627444377395, 230.00000000000003, 231.72597015174959, 233.46347590372523, 235.21252215570087, 236.97311890767651, 238.74526515965215, 240.52896091162779, 242.33420716360343, 244.15099391557907, 245.97932116755471, 247.81928891953035, 249.670896171506, 251.53404392348164, 253.40883117545728, 255.29425892743292, 257.19032617940856, 259.0970329313842, 261.01437818335984, 262.94236293533548, 264.88098718731112, 266.83025193928676, 268.7901571912624, 270.76070394323804, 272.74189219521368, 274.73372294718932, 276.73630520916496, 278.7495399611406, 280.78352621311624, 282.83726496509188, 284.90074621706752, 286.98407096904316, 289.0772482210188, 291.18027797299444, 293.29316022497008, 295.41589597694572, 297.54848422892136, 299.691025980897, 301.84351923287264, 304.00596503484828, 306.17936228682392, 308.36371103879956, 310.5580112907752, 312.76326204275084, 314.97846329472648, 317.20361504670212, 319.43871730867776, 321.6840710606534, 323.93947631262904, 326.20473206460468, 328.48083931658032, 330.74779706855596, 333.0306163205316, 335.32929607250724, 337.63383632448288, 339.95333707645852, 342.28779832842916, 344.6572710804048, 347.04175533238044, 349.43624108435608, 351.84072823633172, 354.25521693830736, 356.679707190278, 359.12419894225364, 361.58869129422928, 364.06218509620492, 366.55067934818056, 369.0541741001562, 371.57266935213184, 374.10616510409748, 376.65466135606812, 379.21815810803876, 381.7966563600094, 384.39015511198004, 386.99865436395068, 389.62215406592132, 392.260654217892, 394.90415486986264, 397.56265592183328, 400.23615747380392, 402.92466052577456, 405.6471641277452, 408.38466817971584, 411.13717273168648, 413.94567728365712, 416.76918183562776, 419.6086813875984, 422.47317593956904, 425.35267049153968, 428.24716504351032, 431.157659595481, 434.08415414745164, 437.02164870942228, 439.97014326139292, 442.92063781336356, 445.8831323653342, 448.85762691730484, 451.84412146927548, 454.84261602124612, 457.85311057321676, 460.8756051251874, 463.91909967715804, 466.98359422912868, 470.05908878109932, 473.14658333307, 476.24607788504064, 479.35757243701128, 482.48006698898192, 485.61456154095256, 488.7610560929232, 491.91955064489384, 495.09004519686448, 498.27254074883512, 501.46703630080576, 504.6735318527764, 507.89202740474704, 511.12252295671768, 514.36401850868832, 517.61751406065896, 520.8830096126296, 524.15950516460024, 527.44700071657088, 530.74649626854152, 534.05709182051216, 537.3787873724828, 540.71158292445344, 544.05547847642408, 547.40947402839472, 550.78356958036536, 554.167765132336, 557.56206068430664, 560.96645623627728, 564.38095178824792, 567.80554734021856, 571.2402428921892, 574.68503844415984, 578.14093409613048, 581.60792964810112, 585.08602520007176, 588.5752207520424, 592.07551630401304, 595.58691185598368, 599.10940740795432, 602.64300295992496, 606.1926985118956, 609.75349406386624, 613.32538961583688, 616.90838516780752, 620.50248071977816, 624.1076762717488, 627.72397182371944, 631.35136737569008, 634.99076292766072, 638.64125847963136, 642.302854031602, 645.97554958357264, 649.65934513554328, 653.35424068751392, 657.06023623948456, 660.7773317914552, 664.50552734342584, 668.24482289539648, 671.99551844736712, 675.75751400933776, 679.5308095613084, 683.31540511327904, 687.11130066524968, 690.91849621722032, 694.73689176919096, 698.5664873211616, 702.40728287313224, 706.25937842510288, 710.12277397707352, 714.09746952904416, 718.0834650810148, 722.09076063298544, 726.10935618495608, 730.13925173692672, 734.18044728889736, 738.232942840868, 742.29673839283864, 746.37183394480928, 750.45822949677992, 754.55592504875056, 758.6649206007212, 762.78521615269184, 766.91681170466248, 771.05970725663312, 775.21390280860376, 779.3793983605744, 783.55619391254504, 787.74428946451568, 791.94368501648632, 796.15438056845696, 800.3763761204276, 804.60967167239824, 808.85426722436888, 813.11016277633952, 817.37735832831016, 821.6558538802808, 825.94564943225144, 830.24674498422208, 834.55914053619272, 838.88283608816336, 843.217831640134, 847.56432719210464, 851.92172274407528, 856.29001829604592, 860.66921384801656, 865.0593094000472, 869.46030495201784, 873.87230050404848, 878.29529605601912, 882.72929160804076, 887.1742871600114, 891.63028271204204, 896.09727826401268, 900.57527381604332, 905.06426936801396, 909.56426492004456, 914.0752604720152, 918.59725602404584, 923.13025157601648, 927.67424712804712, 932.22924268001776, 936.7952382320484, 941.37223378401904, 945.96022933604968, 950.55922488802032, 955.16922044005096, 959.7902159920216, 964.42221154405224, 969.06520709602288, 973.71920264805352, 978.38419820002416, 983.0601937520548, 987.74718930402544, 992.44518485605608, 997.15418040802672, 1001.87417596005736, 1006.605171512028, 1011.34716706405864, 1016.10016261602928, 1020.86415816805992, 1025.63915372003056, 1030.4251492720612, 1035.22214482403184, 1040.03014037600248, 1044.83913592801312, 1049.65913148004376, 1054.4901270320144, 1059.33212258404504, 1064.18511813601568, 1069.04911368804632, 1073.924109240017, 1078.80910479204764, 1083.70410034401828, 1088.60909589604892, 1093.52409144801956, 1098.4490869999902, 1103.3840825519608, 1108.32907810393144, 1113.28407365590208, 1118.24906920787272, 1123.22406475984336, 1128.209060311814, 1133.20405586378464, 1138.20905141575528, 1143.22404696772592, 1148.24904251969656, 1153.2840380716672, 1158.32903362363784, 1163.38402912560848, 1168.44902467757912, 1173.52402022954976, 1178.6090157815204, 1183.70401133349104, 1188.80900688546168, 1193.92400243743232, 1199.04900299140296, 1204.1840035453736, 1209.32900409934424, 1214.48400465331488, 1219.64900520728552, 1224.82400576125616, 1229.9990063152268, 1235.18400686919744, 1240.38900742716808, 1245.60400798513872, 1250.82900854310936, 1256.06400910108, 1261.30900965905064, 1266.56401021702128, 1271.82901077499192, 1277.10401133296256, 1282.3890118909332, 1287.68401244890384, 1292.9890130648744, 1298.30401368084504, 1303.62901429681568, 1308.96401493278632, 1314.30901557875696, 1319.6640162647276, 1325.02901704668824, 1330.40401736864888, 1335.78901808061, 1341.18401879257064, 1346.58901951453128, 1351.99402024649192, 1357.40902092845256, 1362.8340216104132, 1368.26902229237384, 1373.71402297433448, 1379.16902375829512, 1384.63402454225576, 1390.1090253262164, 1395.59402609017704, 1401.08902685413768, 1406.59402762809832, 1412.10902840605896, 1417.63402908392, 1423.16902974588064, 1428.71403046980128, 1434.26903119172192, 1439.83403196968256, 1445.4090327436432, 1450.99403338150384, 1456.58903396536448, 1462.19403450922512, 1467.80903509288576, 1473.4340357213464, 1479.06903642880704, 1484.714037070228, 1490.3690379221696, 1496.0340387780912, 1501.80903964197184, 1507.59404051187344, 1513.38904138577504, 1519.1940421895888, 1525.00904299230544, 1530.8340438199416, 1536.66904447804, 1542.5140450778624, 1548.369045829984, 1554.234046174944, 1560.109046507072, 1566.004046964192, 1571.909047101344, 1577.824047412672, 1583.749047386368, 1589.684047497376, 1595.629047294096, 1601.58404692784, 1607.549046183936, 1613.524045145472, 1619.509043183696, 1625.504040057504, 1631.509036847328, 1637.524033482976, 1643.549030990576, 1649.5840284712, 1655.629025816992, 1661.684022931776, 1667.74902082704, 1673.824018553184, 1
```

**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)
```

