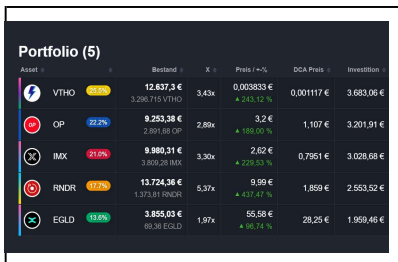
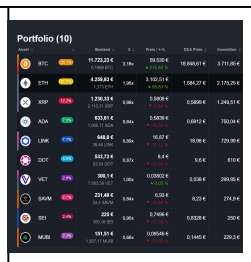
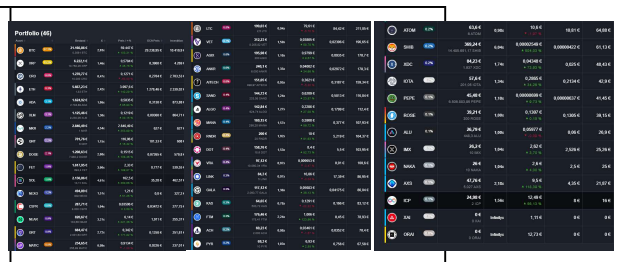
Diversifizierung Gleichgewichtung vs Ungleichgewichtung







5er Portfolio ungleich			5er Portfolio gleich		
16,0	0	>1000X	14,2	0	>1000X
16,0	24	100X -1000X	16,2	23	100X -1000X
17,0	41	40X - 100X	16,4	44	40X - 100X
19,0	30	21X - 40X	18,6	29	21X - 40X
20,0	2	0 X - 17X	21,6	3	0 X - 17X
21,0	3	17X - 21X	22,6	1	17X - 21X
21,0	Durchschnitt 5er Portfolios		23,2	Durchschnitt 5er Portfolios	
23,0	88,0		25,0	88,6	
24,0			25,2		
25,0			26,2		
25,0			26,6		
26,0			27,8		
27,0			28,0		
28,0			28,2		
29,0			28,2		
31,0			28,2		
31,0			28,6		
32,0			28,6		
32,0	Sicherheit ca wie ein		29,0		
33,0	4,5 Portfolio "gleich"		29,0		
33,0			29,0		
33,0			29,8		
33,0			34,2		
34,0	Performance 0,7% schlechter		36,0		
34,0			36,0		
35,0			36,4		
36,0			37,0		
37,0			37,6		
38,0			38,4		
38,0			38,4		

10er Portfolio ungleich			10er Portfolio gleich		
15,0	0	>1000X	22,7	0	>1000X
16,0	36	100X -1000X	26,7	34	100X -1000X
16,0	36	40X - 100X	28,5	52	40X - 100X
17,0	23	21X - 40X	31,7	14	21X - 40X
20,0	3	0 X - 17X	32,2	0	0 X - 17X
21,0	2	17X - 21X	32,8	0	17X - 21X
24,0	Durchschnitt 10er Portfolios		32,9	Durchschnitt 10er Portfolios	
26,0	90,3		34,1	108,9	
26,0			34,6		
26,0			35,2		
27,0			35,6		
27,0			35,7		
28,0			37,3		
29,0			37,3		
29,0			40,1		
29,0			41,5		
30,0			44,2		
32,0			46,2		
32,0	Sicherheit ca wie ein		48,3		
32,0	6 Portfolio "gleich"		48,5		
33,0			49,1		
34,0			49,1		
35,0			49,3		
35,0	Performance 17% schlechter		49,5		
36,0			50,1		
36,0			51,0		
37,0			51,7		
39,0			52,2		
40,0			53,4		
40,0			53,6		

20er Portfolio ungleich			20er Portfolio gleich		
15,0	0	>1000X	33,7	0	>1000X
16,0	37	100X -1000X	39,7	47	100X -1000X
24,0	47	40X - 100X	42,7	51	40X - 100X
25,0	14	21X - 40X	43,0	2	21X - 40X
29,0	2	0 X - 17X	43,8	0	0 X - 17X
29,0	0	17X - 21X	43,9	0	17X - 21X
30,0	Durchschnitt 20er Portfolios		47,2	Durchschnitt 20er Portfolios	
30,0	112,6		47,3	124,1	
31,0			47,4		
33,0			49,6		
34,0			52,5		
36,0			53,2		
38,0			54,3		
39,0			57,8		
39,0			58,1		
39,0			59,5		
41,0			59,7		
41,0			60,6		
44,0	Sicherheit ca wie ein		62,9		
44,0	10 Portfolio "gleich"		63,0		
45,0			63,6		
45,0			64,4		
45,0			65,9		
47,0	Performance 9,3% schlechter		66,6		
49,0			66,6		
50,0			67,5		
51,0			67,5		
51,0			68,9		
52,0			69,5		
56,0			72,7		

46er Portfolio ungleich			50er Portfolio gleich		
32,0	1	>1000X	49,4	0	>1000X
34,0	36	100X -1000X	57,3	67	100X -1000X
34,0	54	40X - 100X	60,0	33	40X - 100X
36,0	9	21X - 40X	63,0	0	21X - 40X
36,0	0	0 X - 17X	65,8	0	0 X - 17X
37,0	0	17X - 21X	65,9	0	17X - 21X
37,0	Durchschnitt 46er Portfolios		68,8	Durchschnitt 50er Portfolios	
37,0	110,1		70,7	128,0	
38,0			71,6		
40,0			72,6		
40,0			73,2		
43,0			74,1		
43,0			75,1		
44,0			75,2		
45,0			78,7		
46,0			78,7		
47,0			79,1		
47,0			79,8		
47,0	Sicherheit ca wie ein		81,7		
48,0	15 Portfolio "gleich"		82,7		
48,0			83,8		
48,0			83,8		
51,0			84,5		
51,0	Performance 14% schlechter		84,5		
51,0			84,7		
53,0			85,2		
53,0			86,4		
54,0			86,7		
54,0			87,7		
55,0			92,7		

```
import random

# Provided coin rise data
data = {
2: 57,
3: 16,
5: 20,
6: 13,
7: 17,
8: 9,
9: 167,
10: 9,
11: 28,
12: 322,
13: 27,
14: 159,
15: 22,
16: 43,
17: 17,
19: 48,
20: 12,
21: 83,
22: 32,
23: 470,
24: 28,
25: 33,
26: 29,
27: 17,
28: 34,
30: 75,
31: 11,
32: 558,
33: 19,
34: 44,
36: 163,
39: 40,
40: 30,
41: 8,
42: 18,
43: 83,
44: 23,
45: 36,
46: 56,
47: 24,
48: 29,
49: 17,
50: 10,
51: 12,
52: 12,
53: 30,
```

```
54: 818,
55: 38,
56: 53,
57: 24,
58: 54,
59: 128,
60: 399,
61: 92,
62: 8,
63: 968,
64: 51,
65: 123,
66: 16,
67: 15,
68: 0,
69: 83,
70: 25,
71: 95,
72: 65,
73: 73,
74: 78,
75: 73,
76: 225,
77: 6,
79: 16,
80: 11,
81: 97,
82: 110,
83: 22,
84: 50,
85: 0,
86: 50,
87: 29,
88: 46,
89: 23,
90: 132,
91: 40,
92: 11,
93: 929,
94: 42,
95: 214,
96: 30,
97: 19,
98: 8,
99: 7,
100: 42,
101: 36,
102: 15,
103: 35,
104: 72,
105: 86,
108: 8,
109: 406,
110: 104,
111: 50,
112: 2595,
113: 87,
114: 28,
115: 44,
116: 27,
117: 13,
119: 28,
120: 32,
121: 16,
122: 4,
123: 19,
124: 6,
125: 4,
126: 0,
127: 26,
128: 33,
129: 7,
130: 0,
131: 12,
132: 354,
133: 20,
134: 71,
135: 32,
136: 19,
137: 27,
138: 87,
139: 18,
140: 496,
141: 38,
142: 11,
143: 46,
144: 0,
145: 130,
146: 0,
147: 41,
148: 31,
149: 22,
150: 5,
151: 19,
152: 192,
153: 27,
154: 15,
155: 14,
156: 44,
157: 67,
158: 36,
159: 35,
160: 119,
161: 36,
162: 74,
163: 51,
164: 7,
165: 0,
166: 83,
167: 33,
168: 8,
169: 0,
170: 30,
171: 71,
172: 275,
173: 59,
174: 131,
175: 14,
176: 144,
177: 36,
178: 23,
179: 108,
180: 50,
181: 8,
182: 30,
183: 0,
184: 16,
185: 24,
186: 154,
187: 36,
188: 3214,
189: 8,
190: 250,
191: 2626,
192: 315,
193: 36,
195: 288,
196: 82,
197: 18,
198: 90,
199: 5,
200: 48
}

# Money stacks for the 20 coins in a portfolio
money_stacks = [
    1000, 10, 10, 10, 10, 10, 10, 10, 10, 10,
    10, 10, 10, 10, 10, 10, 10, 10, 10, 10
]

# Function to calculate the weighted average rise for a portfolio
def calculate_weighted_average_rise(selected_keys):
    total_money = sum(money_stacks)
    weighted_rise_sum = sum(data[key] * money_stacks[i] for i, key in enumerate(selected_keys))
    return weighted_rise_sum / total_money if total_money != 0 else 0

# List to store the average rises for each portfolio
average_rises = []

# Generate 100 random portfolios and calculate their average rises
for _ in range(100):
    # Randomly select 20 unique keys (coins) from the data
    selected_keys = random.sample(list(data.keys()), 20)

    # Calculate the weighted average rise for the selected portfolio
    average_rise = calculate_weighted_average_rise(selected_keys)
    average_rises.append(average_rise)

# Calculate the overall average rise across all portfolios
overall_average_rise = sum(average_rises) / len(average_rises)

print(f"Overall Average Rise: {overall_average_rise}")

# Sort the average rises in ascending order and convert to integers for no decimals
sorted_average_rises = sorted(int(average_rise) for average_rise in average_rises)

# Print the sorted list
print(sorted_average_rises)
```

```
# Erstellen eines Strings mit allen Werten, getrennt durch Zeilenumbrüche
sorted_rises_str = "\n".join(map(str, sorted_average_rises))

# Drucken des Strings
print(sorted_rises_str)
```