HOENN SAFARI ZONE RESEARCH

Analysis by Professor Rex/Rex_PHD

Forward:

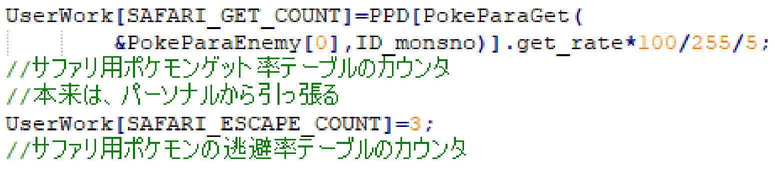
When looking through the Bulbapedia page for the Hoenn Safari Zone something jumped out at me… the guide didn't take into account the 15% chance of a Pokémon fleeing on the first turn. Something that can happen regardless of how skilled you are at throwing Pokéblocks. To rectify this, I took a dive into the source code of Pokémon Ruby & Sapphire. Read through for a full understanding of the Hoenn Safari Zone & learn how to guarantee a successful catch rate of at least 70% in any encounter!

TO SKIP TO THE CATCH GUIDE: CLICK HERE

The Catch Factor

The Hoenn Safari Zone calculates Catch Factors in the same way that Pokémon Firered and Leafgreen do in their Safari Zones. For more information see my Kanto Safari Zone Research.
The main thing to take away is that catch rates in the Safari zone always end up being less than or equal to their usual catch rates.

Snippet of code for getting catch and flee factors

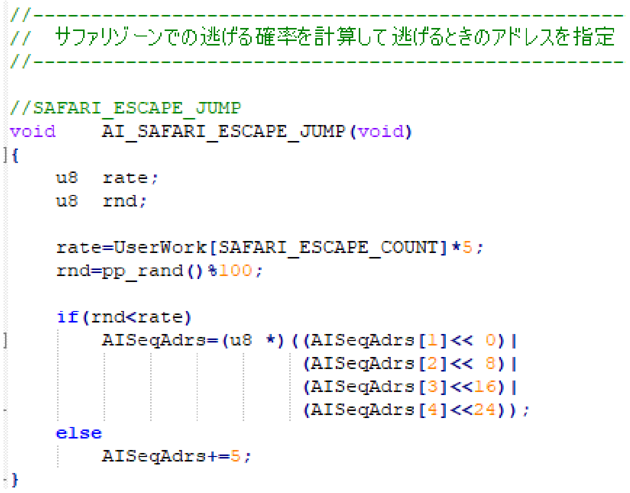


```
UserWork[SAFARI_GET_COUNT]=PPD[PokeParaGet(
        &PokeParaEnemy[0],ID_monsno)].get_rate*100/255/5;
//サファリ用ポケモンゲット率テーブルのカウンタ
//本来は、パーソナルから引っ張る
UserWork[SAFARI_ESCAPE_COUNT]=3;
//サファリ用ポケモンの逃避率テーブルのカウンタ
```

The Flee Factor

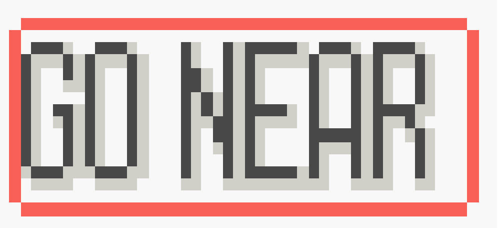
The Hoenn Flee factors are quite a bit different than the Flee Factors in Kanto though with all Pokémon encounters starting with a base flee factor of 3, which is equal to 15% per turn.
This flee factor can be modified by going near, or throwing Pokéblocks.
The flee factor cannot be reduced below 0, or increased above 20.
It should be noted that modifications to the Flee Factor do not affect the flee check of the current turn. Flee Factor changes do not take effect until the following turn's flee check.
A flee check is preformed at the start of each turn by multiplying the current flee factor by 5 and comparing it to a random number between 0-99 (inclusive). If the random number is less, the Pokémon will flee at the end of the turn unless it is caught first.

Snippet of code for preforming flee checks



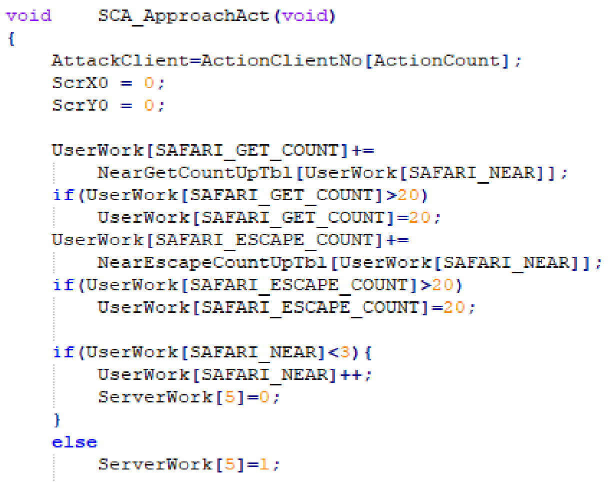
```
//---------------------------------------------------
//  サファリゾーンでの逃げる確率を計算して逃げるときのアドレスを指定
//---------------------------------------------------

//SAFARI_ESCAPE_JUMP
void    AI_SAFARI_ESCAPE_JUMP(void)
{
    u8  rate;
    u8  rnd;

    rate=UserWork[SAFARI_ESCAPE_COUNT]*5;
    rnd=pp_rand()%100;

    if(rnd<rate)
        AISeqAdrs=(u8 *)((AISeqAdrs[1]<< 0)|
                          (AISeqAdrs[2]<< 8)|
                          (AISeqAdrs[3]<<16)|
                          (AISeqAdrs[4]<<24));
    else
        AISeqAdrs+=5;
}
```

Go Near



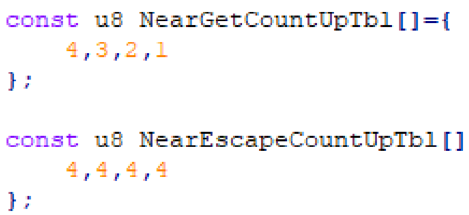
Going near a Pokémon is entirely useless, upon first going near a Pokémon the catch factor and the flee factor will both increase by 4. The second and third attempts to go near will increase the catch factor by 3 & 2 respectively while still increasing the flee factor by 4 each time. Any attempts to go near past the third will display the message "*PLAYER* can't get any closer!"; despite this the Flee Factor will still increase by 4 and the catch factor by 1. For a Pokémon such as Heracross this brings it's Catch Factor to 7 which is about 19.75% per ball; up from 8.09%. However, it's flee rate also increases by 20%, up to 35% per turn. This risk will never even out and you are better off chucking balls than attempting to get closer to the Pokémon.

Snippet of code that demonstrates how going near works



```
void     SCA_ApproachAct(void)
{
    AttackClient=ActionClientNo[ActionCount];
    ScrX0 = 0;
    ScrY0 = 0;

    UserWork[SAFARI_GET_COUNT]+=
        NearGetCountUpTbl[UserWork[SAFARI_NEAR]];
    if(UserWork[SAFARI_GET_COUNT]>20)
        UserWork[SAFARI_GET_COUNT]=20;
    UserWork[SAFARI_ESCAPE_COUNT]+=
        NearEscapeCountUpTbl[UserWork[SAFARI_NEAR]];
    if(UserWork[SAFARI_ESCAPE_COUNT]>20)
        UserWork[SAFARI_ESCAPE_COUNT]=20;

    if(UserWork[SAFARI_NEAR]<3){
        UserWork[SAFARI_NEAR]++;
        ServerWork[5]=0;
    }
    else
        ServerWork[5]=1;
```

Snippet of code that shows how the effects of going near change the more you do it

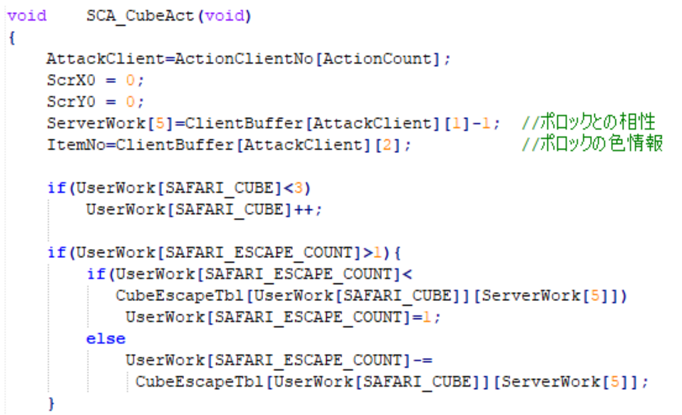


```
const u8 NearGetCountUpTbl[]={
    4,3,2,1
};

const u8 NearEscapeCountUpTbl[]
    4,4,4,4
};
```

Pokéblocks

It is well known that throwing Pokéblocks in the Hoenn Safari Zone can bring a Pokémon's flee factor down to 0. Doing so is quite easy yet slightly counter-intuitive. When you first encounter a Pokémon, the base flee factor is always 3. Our goal is to reduce this down as low as possible. One might assume that throwing a Pokéblock that the wild Pokémon likes would do the trick. However, this isn't the case. We actually want to throw a neutral Pokéblock to reduce the flee factor to 0. The reason for this is that on the first thrown Pokéblock will drop flee rate by 3 if it is neutral, 5 if they like it, or 0 if they don't like it. A reduction by 5 sounds like it should do the trick right? Unfortunately, not. If a Pokémon's flee rate is greater than 1, attempting to reduce the flee rate below 0 will instead change the flee rate to 1. Reducing it to exactly 0 will work. On the first turn our goal is actually to throw a neural Pokéblock. This will bring the Flee Rate down to exactly 0. If a Pokémon's flee factor is 1, it cannot be reduced further with Pokéblocks.

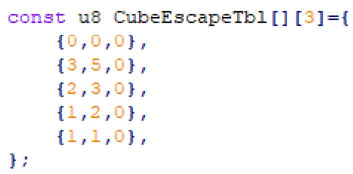
Snippet of code that demonstrates how Pokéblocks affect Flee Rates



```
void     SCA_CubeAct(void)
{
    AttackClient=ActionClientNo[ActionCount];
    ScrX0 = 0;
    ScrY0 = 0;
    ServerWork[5]=ClientBuffer[AttackClient][1]-1;  //ポロックとの相性
    ItemNo=ClientBuffer[AttackClient][2];           //ポロックの色情報

    if(UserWork[SAFARI_CUBE]<3)
        UserWork[SAFARI_CUBE]++;

    if(UserWork[SAFARI_ESCAPE_COUNT]>1){
        if(UserWork[SAFARI_ESCAPE_COUNT]<
           CubeEscapeTbl[UserWork[SAFARI_CUBE]][ServerWork[5]])
            UserWork[SAFARI_ESCAPE_COUNT]=1;
        else
            UserWork[SAFARI_ESCAPE_COUNT]-=
             CubeEscapeTbl[UserWork[SAFARI_CUBE]][ServerWork[5]];
    }
```

Pokéblock # ->	First	Second	Third	Fourth+
Neutral Pokéblock	-3	-2	-1	-1
Liked Pokéblock	-5	-3	-2	-1
Disliked Pokéblock	±0	±0	±0	±0

Table that stores how Pokéblocks effectiveness change as more are thrown



```
const u8 CubeEscapeTbl[][3]={
    {0,0,0},
    {3,5,0},
    {2,3,0},
    {1,2,0},
    {1,1,0},
};
```

Pokéblock Feeder

The Pokéblock feeder in the Hoenn Safari Zone allows players to manipulate the natures of the Pokémon that are encountered. Once a Pokéblock is inserted into the feeder each encountered Pokémon has an 80% chance to be forced to have a nature which enjoys the same flavor as the Pokéblock that has been put into the feeder. When the Safari zone was first being designed this was planned to be done through a repeating random call that would run until such a nature was found. However, at some point in development this was changed to be done by "randomizing" the nature table and then using the first matching nature found after this randomization. Interestingly the randomization algorithm they used to make this randomized list of natures doesn't actually do a great job of randomizing the natures. Below are 5 lists of natures that have been created using this nature randomizer. You will notice that for something that should be random they all look eerily similar.

(Each number corresponds to a Nature ID as listed in the chart to the right)

[19, 24, 23, 22, 15, 21, 16, 17, 20, 11, 18, 7, 10, 12, 6, 14, 13, 4, 5, 3, 9, 1, 0, 2, 8]

[22, 23, 24, 12, 21, 15, 20, 18, 3, 19, 13, 16, 17, 14, 9, 11, 5, 6, 2, 10, 1, 0, 7, 8, 4]

[24, 22, 19, 23, 10, 18, 20, 21, 14, 12, 17, 15, 13, 1, 5, 16, 11, 9, 7, 2, 6, 0, 4, 8, 3]

[23, 22, 16, 24, 21, 17, 18, 12, 8, 5, 10, 19, 15, 14, 20, 0, 4, 11, 9, 1, 7, 13, 2, 3, 6]

[23, 13, 22, 24, 21, 20, 15, 18, 12, 11, 3, 19, 17, 10, 14, 9, 5, 8, 7, 16, 6, 4, 2, 1, 0]

The lists all follow a pattern that looks to have the high numbers frequently in early positions and lower numbers in later positions. As suspected, this creates an uneven distribution of what natures are selected when a Pokéblock has been put into the feeder. This can be a huge help when making decisions on which neutral Pokéblocks to throw. The following odds were derived from 10,000,000 trials and are the odds of getting a specific nature if the 80% forced nature check passes.

Spicy Pokéblock in feeder:
Lonely has a 22.00% chance
Brave has a 23.17% chance
Adamant has a 25.57% chance
Naughty has a 29.25% chance

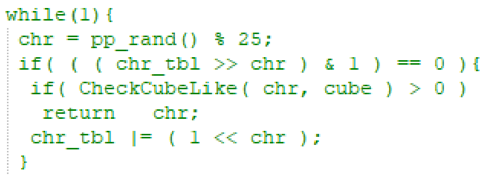
Sour Pokéblock in feeder:
Bold has a 14.01% chance
Relaxed has a 22.47% chance
Impish has a 28.28% chance
Lax has a 35.24% chance

Sweet Pokéblock in feeder:
Timid has a 12.93% chance
Hasty has a 17.36% chance
Jolly has a 30.37% chance
Naive has a 39.34% chance

Dry Pokéblock in feeder:
Modest has a 11.66% chance
Mild has a 16.95% chance
Quiet has a 24.16% chance
Rash has a 47.23% chance

Bitter Pokéblock in feeder:
Calm has an 8.71% chance
Gentle has a 15.61% chance
Sassy has a 27.66% chance
Careful has a 48.02% chance

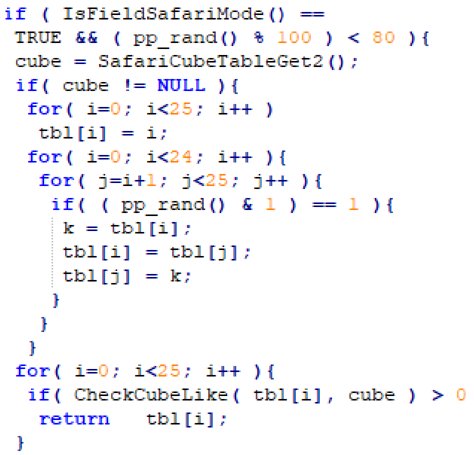
Original code for getting a random nature that matches flavor



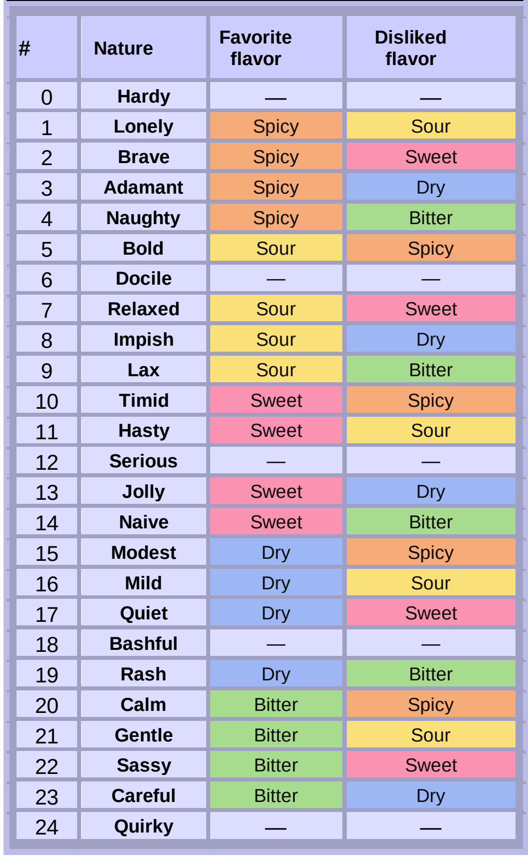
```
while(1){
 chr = pp_rand() % 25;
 if( ( ( chr_tbl >> chr ) & 1 ) == 0 ){
  if( CheckCubeLike( chr, cube ) > 0 )
   return   chr;
  chr_tbl |= ( 1 << chr );
 }
```

Production code for getting a random nature that matches flavor

Version with comments



```
if ( IsFieldSafariMode() ==
 TRUE && ( pp_rand() % 100 ) < 80 ){
 cube = SafariCubeTableGet2();
 if( cube != NULL ){
  for( i=0; i<25; i++ )
   tbl[i] = i;
  for( i=0; i<24; i++ ){
   for( j=i+1; j<25; j++ ){
    if( ( pp_rand() & 1 ) == 1 ){
     k = tbl[i];
     tbl[i] = tbl[j];
     tbl[j] = k;
    }
   }
  }
 for( i=0; i<25; i++ ){
  if( CheckCubeLike( tbl[i], cube ) > 0
   return   tbl[i];
 }
```



#	Nature	Favorite flavor	Disliked flavor
0	Hardy	—	—
1	Lonely	Spicy	Sour
2	Brave	Spicy	Sweet
3	Adamant	Spicy	Dry
4	Naughty	Spicy	Bitter
5	Bold	Sour	Spicy
6	Docile	—	—
7	Relaxed	Sour	Sweet
8	Impish	Sour	Dry
9	Lax	Sour	Bitter
10	Timid	Sweet	Spicy
11	Hasty	Sweet	Sour
12	Serious	—	—
13	Jolly	Sweet	Dry
14	Naive	Sweet	Bitter
15	Modest	Dry	Spicy
16	Mild	Dry	Sour
17	Quiet	Dry	Sweet
18	Bashful	—	—
19	Rash	Dry	Bitter
20	Calm	Bitter	Spicy
21	Gentle	Bitter	Sour
22	Sassy	Bitter	Sweet
23	Careful	Bitter	Dry
24	Quirky	—	—

Nature table from Bulbapedia

Optimizing Pokéblocks

Knowing that some natures are more likely than others allows us to optimize our choice of Pokéblock for our first throw. In all cases the best option is to throw a Pokéblock that is disliked by the nature with the lowest chance to occur. For example, when a Bitter Pokéblock is in the feeder there is only an 8.71% chance that a forced roll will result in a Calm nature. Looking at the nature table we can see that in this case our best bet is to throw a Spicy Pokéblock. Overall, this would give us an 86.6% chance of our first Pokéblock being a neutral one and dropping the flee rate to 0. For reference the odds of getting a neutral Pokéblock when throwing a random flavor at a random nature is 68%.

Best-Case Scenario:

Spicy **Pokéblock in feeder:** throw a *sour* Pokéblock, 76.00% chance for neutral Pokéblock
Sour **Pokéblock in feeder:** throw a *spicy* Pokéblock, 82.39% chance for neutral Pokéblock
Sweet **Pokéblock in feeder:** throw a *spicy* Pokéblock, 83.25% chance for neutral Pokéblock
Dry **Pokéblock in feeder:** throw a *spicy* Pokéblock, 84.28% chance for neutral Pokéblock
Bitter **Pokéblock in feeder:** throw a *spicy* Pokéblock, 86.63% chance for neutral Pokéblock

Optimizing Patterns (with Pokéblock Feeder)

Below is the best pattern to use in the Safari Zone while a Pokéblock feeder is in effect. Different Species of Pokémon do best with different patterns. Refer to the charts below to determine which pattern works best for the Pokémon you are attempting to catch.

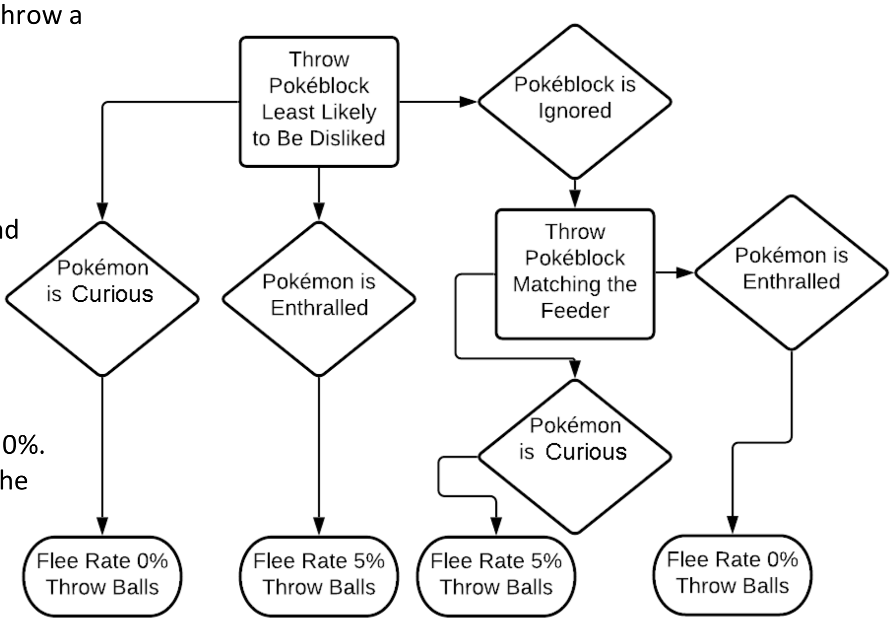
Optimal Pattern: *throwing a second Pokéblock sometimes*

On the first turn our goal is to throw a neural Pokéblock. This will bring the Flee Rate down to exactly 0. For the best chance to do this we want to throw a Pokéblock with a single flavour.

If your first Pokéblock **makes the wild Pokémon curious:** the flee rate is now 0%. Throw balls.

If your first Pokéblock is **ignored:** the second one you throw should attempt to be one the Pokémon likes. For the best chance to do this throw a Pokéblock that matches the flavor of the Pokéblock in the Pokéblock feeder. If the second Pokéblock enthralls the Pokémon the flee rate is now 0%. If the second Pokéblock does not enthrall the wild Pokémon, the flee rate is now 5%. In either case: throw balls.

If your first Pokéblock **enthralls the Pokémon:** the flee rate is now 5% per turn. Throw balls.



Optimizing Patterns (with Random Natures)

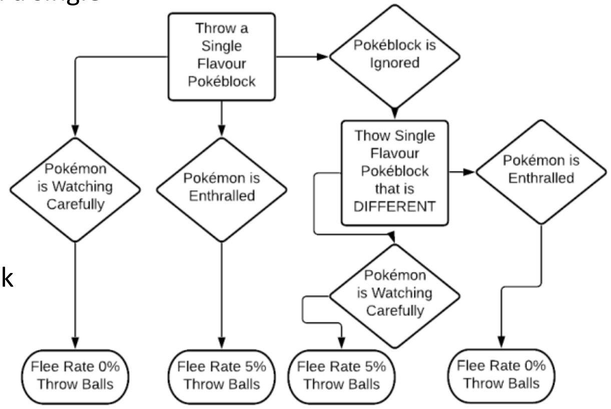
Below is the best pattern to use on randomly generated Pokémon in the Hoenn Safari Zone. Different Species of Pokémon do best with different patterns. Refer to the charts below to determine which pattern works best for the Pokémon you are attempting to catch.

Optimal Pattern: *throwing a second Pokéblock sometimes*
On the first turn our goal is to throw a neural Pokéblock. This will bring the Flee Rate down to exactly 0. For the best chance to do this we want to throw a Pokéblock with a single flavour.

If your first Pokéblock **makes the wild Pokémon curious:** the flee rate is now 0%. Throw balls.

If your first Pokéblock is **ignored:** Throw a second Pokéblock of a different flavor than the first. If the second Pokéblock enthralls the wild Pokémon the flee rate is now 0%. If the second Pokéblock does not enthrall the wild Pokémon, the flee rate is now 5%. In either case: throw balls.

If your first Pokéblock **enthralls the Pokémon:** the flee rate is now 5% per turn. Throw balls.



Calculating Success Rates

Below are the odds for catching each available species of Pokémon in any condition. Compare success rates to determine whether throwing a Pokéblock will be worth it or not.

DODRIO, PINSIR, AIPOM, WOBBUFFET, HERACROSS, STANTLER, & MILTANK

Base catch rate: 45
Base catch factor: 3
Modified catch rate: 38
Odds of capture per ball: 8.09%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only**: **36.94%**

Odds of capture with **random nature** & optimal pattern: **69.82%**
Odds of capture with *spicy* table & optimal pattern: 74.25%
Odds of capture with *sour* table & optimal pattern: 75.23%
Odds of capture with *sweet* table & optimal pattern: 75.36%
Odds of capture with *dry* table & optimal pattern: 75.52%
Odds of capture with ***bitter*** **table** & optimal pattern: **75.88%**

SEAKING, GIRAFARIG, & GLIGAR

Base catch rate: 60
Base catch factor: 4
Modified catch rate: 51
Odds of capture per ball: 10.66%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 44.27%**

Odds of capture with **random nature** & optimal pattern: **74.2%**
Odds of capture with *spicy* table & optimal pattern: 78.15%
Odds of capture with *sour* table & optimal pattern: 79.15%
Odds of capture with *sweet* table & optimal pattern: 79.28%
Odds of capture with *dry* table & optimal pattern: 79.44%
Odds of capture with ***bitter*** **table** & optimal pattern: **79.81%**

Code for these outputs can be found here

GOLDUCK, XATU, & OCTILLERY

Base catch rate: 75
Base catch factor: 5
Modified catch rate: 63
Odds of capture per ball: 12.33%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 48.36%**

Odds of capture with **random nature** & optimal pattern: **75.88%**
Odds of capture with *spicy* table & optimal pattern: 79.47%
Odds of capture with *sour* table & optimal pattern: 80.46%
Odds of capture with *sweet* table & optimal pattern: 80.59%
Odds of capture with *dry* table & optimal pattern: 80.75%
Odds of capture with ***bitter* table** & optimal pattern: **81.12%**

QUAGSIRE

Base catch rate: 90
Base catch factor: 7
Modified catch rate: 89
Odds of capture per ball: 19.75%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 62.11%**

Odds of capture with **random nature** & optimal pattern: **79.11%**
Odds of capture with *spicy* table & optimal pattern: 81.36%
Odds of capture with *sour* table & optimal pattern: 82.3%
Odds of capture with *sweet* table & optimal pattern: 82.43%
Odds of capture with *dry* table & optimal pattern: 82.58%
Odds of capture with ***bitter* table** & optimal pattern: **82.93%**

GLOOM, RHYHORN, TEDDIURSA, HOUNDOUR, & PHANPY

Base catch rate: 120
Base catch factor: 9
Modified catch rate: 114
Odds of capture per ball: 23.42%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 67.07%**

Odds of capture with **random nature** & optimal pattern: **79.81%**
Odds of capture with *spicy* table & optimal pattern: 81.58%
Odds of capture with *sour* table & optimal pattern: 82.51%
Odds of capture with *sweet* table & optimal pattern: 82.63%
Odds of capture with *dry* table & optimal pattern: 82.78%
Odds of capture with ***bitter* table** & optimal pattern: **83.12%**

PIKACHU, PYSDUCK, DODUO, NATU, MARILL, PINECO, SNUBBULL, SHUCKLE, & REMORAID

Base catch rate: 190
Base catch factor: 14
Modified catch rate: 178
Odds of capture per ball: 40.96%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 82.21%**

Odds of capture with random nature & optimal pattern: 81.46%
Odds of capture with *spicy* table & optimal pattern: 81.99%
Odds of capture with *sour* table & optimal pattern: 82.86%
Odds of capture with *sweet* table & optimal pattern: 82.98%
Odds of capture with *dry* table & optimal pattern: 83.12%
Odds of capture with ***bitter* table** & optimal pattern: **83.44%**

MAREEP & SUNKERN

Base catch rate: 235
Base catch factor: 18
Modified catch rate: 229
Odds of capture per ball: 50.28%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 87.07%**

Odds of capture with random nature & optimal pattern: 81.91%
Odds of capture with *spicy* table & optimal pattern: 82.1%
Odds of capture with *sour* table & optimal pattern: 82.95%
Odds of capture with *sweet* table & optimal pattern: 83.07%
Odds of capture with *dry* table & optimal pattern: 83.2%
Odds of capture with *bitter* table & optimal pattern: 83.52%

ODDISH, GEODUDE, GOLDEEN, MAGIKARP, HOOTHOOT, LEDYBA, SPINARAK, & WOOPER

Base catch rate: 255
Base catch factor: 20
Modified catch rate: 255
Odds of capture per ball: 50.28%
Odds of fleeing per turn: 15.01%
Odds of capture with **balls only: 87.07%**

Odds of capture with random nature & optimal pattern: 81.91%
Odds of capture with *spicy* table & optimal pattern: 82.1%
Odds of capture with *sour* table & optimal pattern: 82.95%
Odds of capture with *sweet* table & optimal pattern: 83.07%
Odds of capture with *dry* table & optimal pattern: 83.2%
Odds of capture with *bitter* table & optimal pattern: 83.52%

Questions can be directed to me on any of the following media platforms:
Discord: The Rex Men Twitch: BrofessorsLab Reddit: MineOSaurus_Rex
Instagram: rex_phd Youtube: Professor Rex
Email: the.professor.rex@gmail.com