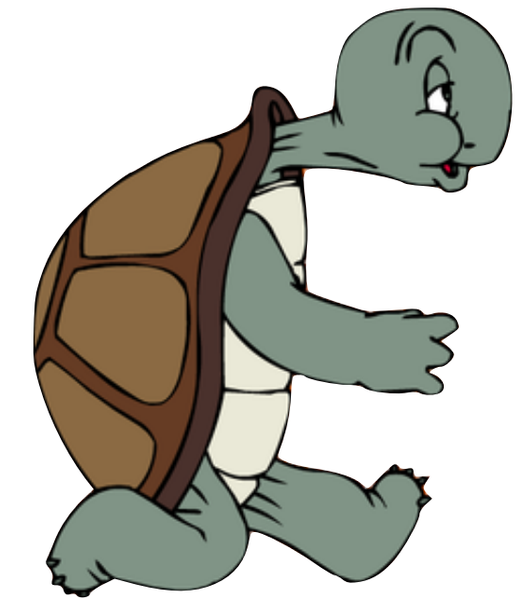
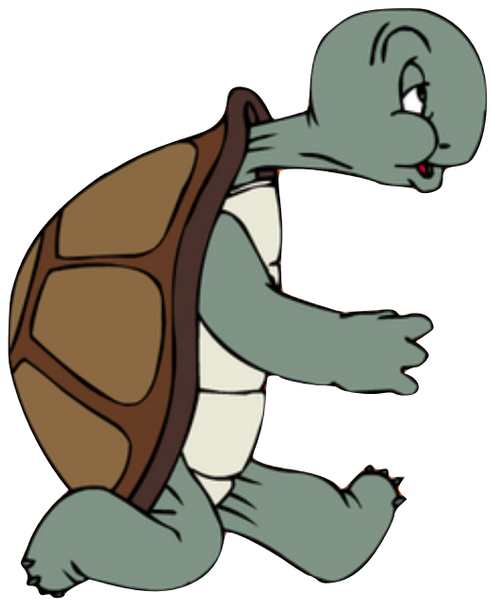


Tortoise v Hare

Basic Rule

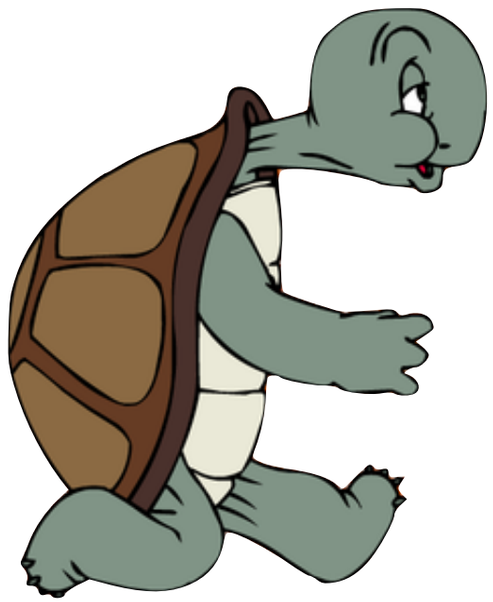


trumptonalia

Monte Carlo

Monte Carlo methods, or **Monte Carlo experiments**, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.

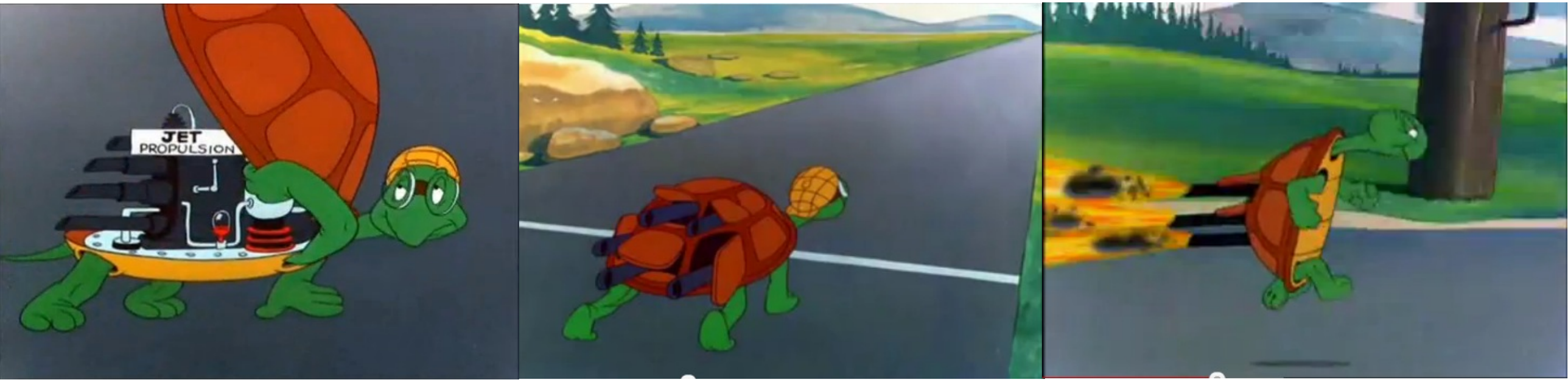
Basic Rule



Hare likes to nap

Hare

```
function updateHare(hare) {
    if (hare.missATurn) {
        hare.missATurn = false;
        hare.lastMissed = hare.space;
        return;
    }
    if (hare.isSleeping) {
        let r1 = roll(1);
        let r2 = roll(1);
        if (isEven(r1)) {
            hare.isSleeping = false;
            hare.space += r1 + r2;
        }
    } else {
        let r1 = roll(1);
        let r2 = roll(1);
        if (r1 == r2) {
            hare.isSleeping = true;
            hare.sleeps += 1;
        } else {
            hare.space += r1 + r2;
        }
    }
}
```



Tortoise has a jet engine

Tortoise

```
function updateTortoise(tortoise) {
    if (tortoise.missATurn) {
        tortoise.missATurn = false;
        tortoise.lastMissed = tortoise.space;
        return;
    }
    let r1 = roll(1);
    if (tortoise.isBrokenDown) {
        if (r1 == 6) tortoise.isBrokenDown = false;
        return;
    } else if (tortoise.isFlying) {
        if (r1 == 1) {
            tortoise.isFlying = false;
            tortoise.space += 1;
            return;
        }
        // else if (r1 == 6) {
        //  tortoise.isFlying = false;
        //  tortoise.isBrokenDown = true;
        // }
        tortoise.space += r1 * flySpeed;
    } else {
        if (r1 == 6) {
            tortoise.isFlying = true;
            tortoise.space += r1 * flySpeed;
            tortoise.flights += 1;
            return;
        }
        tortoise.space += r1;
    }
}
```

Game Simulation

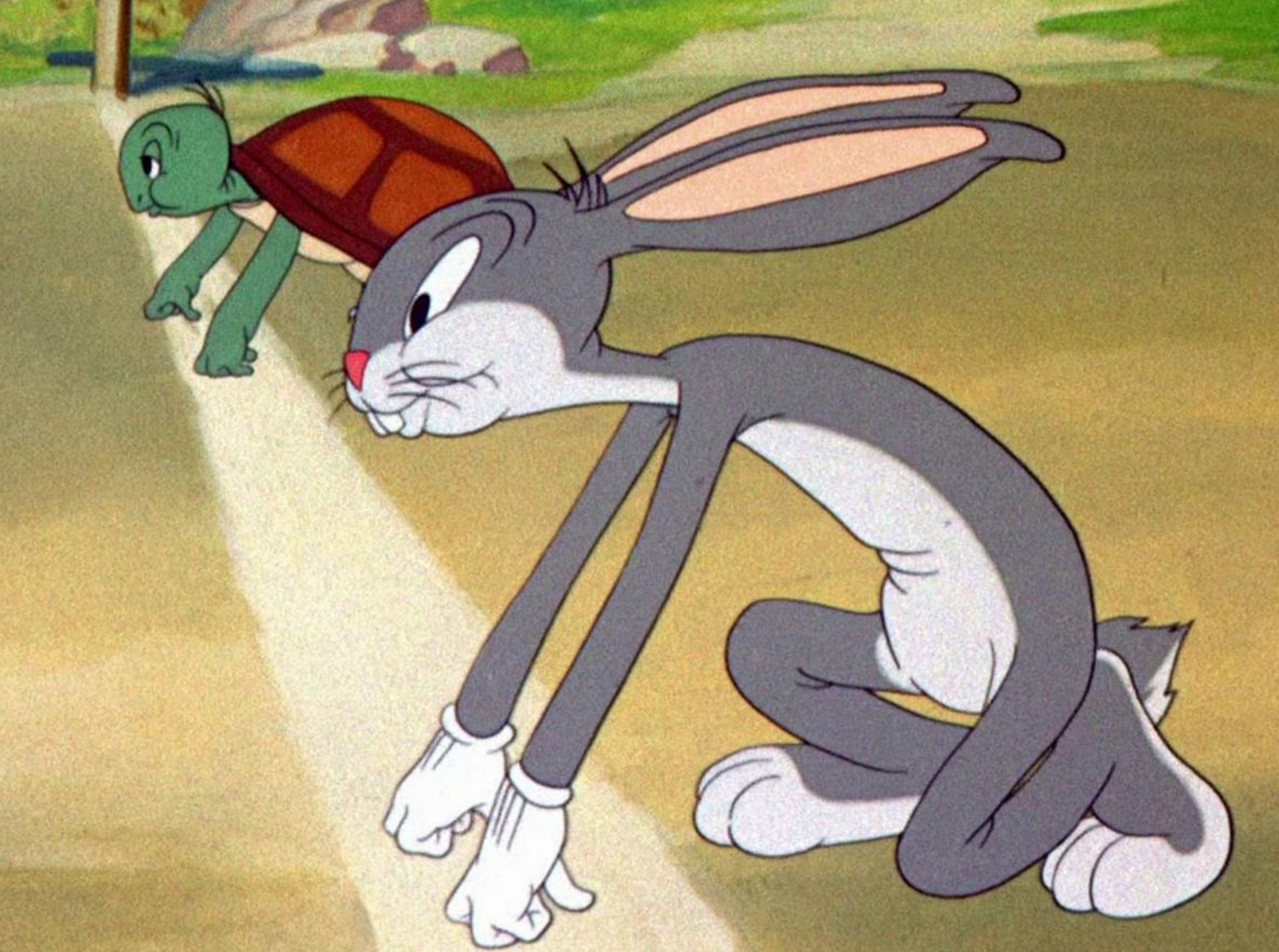
```
while (!hasWon(spaces, hare) && !hasWon(spaces, tortoise)) {
    let rollAgain = false;
    do {
        updateHare(hare);
        rollAgain = handleBoardSquare(board, hare);
    } while (rollAgain);
    do {
        updateTortoise(tortoise);
        rollAgain = handleBoardSquare(board, tortoise);
    } while (rollAgain);
    turns += 1;
}
```

Monte Carlo

```
for (i = 0; i < sims; i++) {
    const board = makeBoard(spaces);
    const [t, h, turns, sleeps, flights] = simulateGame(board);

    totTurns += turns;
    totSleeps += sleeps;
    totFlights += flights;

    if (t === h) {
        totDraws += 1;
        addBoardToDrawBoardsArray(drawBoards, board);
    } else {
        tortoiseWins += t;
        hareWins += h;
    }
}
```

Sims

```
const sims = 100000;
```



Demo 1

A Board

trumptonalia

Board

```
function handleBoardSquare(board, animal) {
    let square = board[animal.space];

    switch (square) {
        case "empty":
            return false;
        case "f3":
            animal.space += 3;
            return false;
        case "b3":
            animal.space -= 3;
            if (animal.space < 0) animal.space = 0;
            return false;
        case "mt":
            if (animal.lastMissed != animal.space) animal.missATurn = true;
        case "ra":
            return true;
        case "cr":
            if (animal.is == "tortoise") {
                animal.isFlying = false;
                // animal.isBrokenDown = true;
            }
            return false;
        case "zz":
            if (animal.is == "hare") animal.isSleeping = true;
            return false;
    }

    return;
}
```

Board

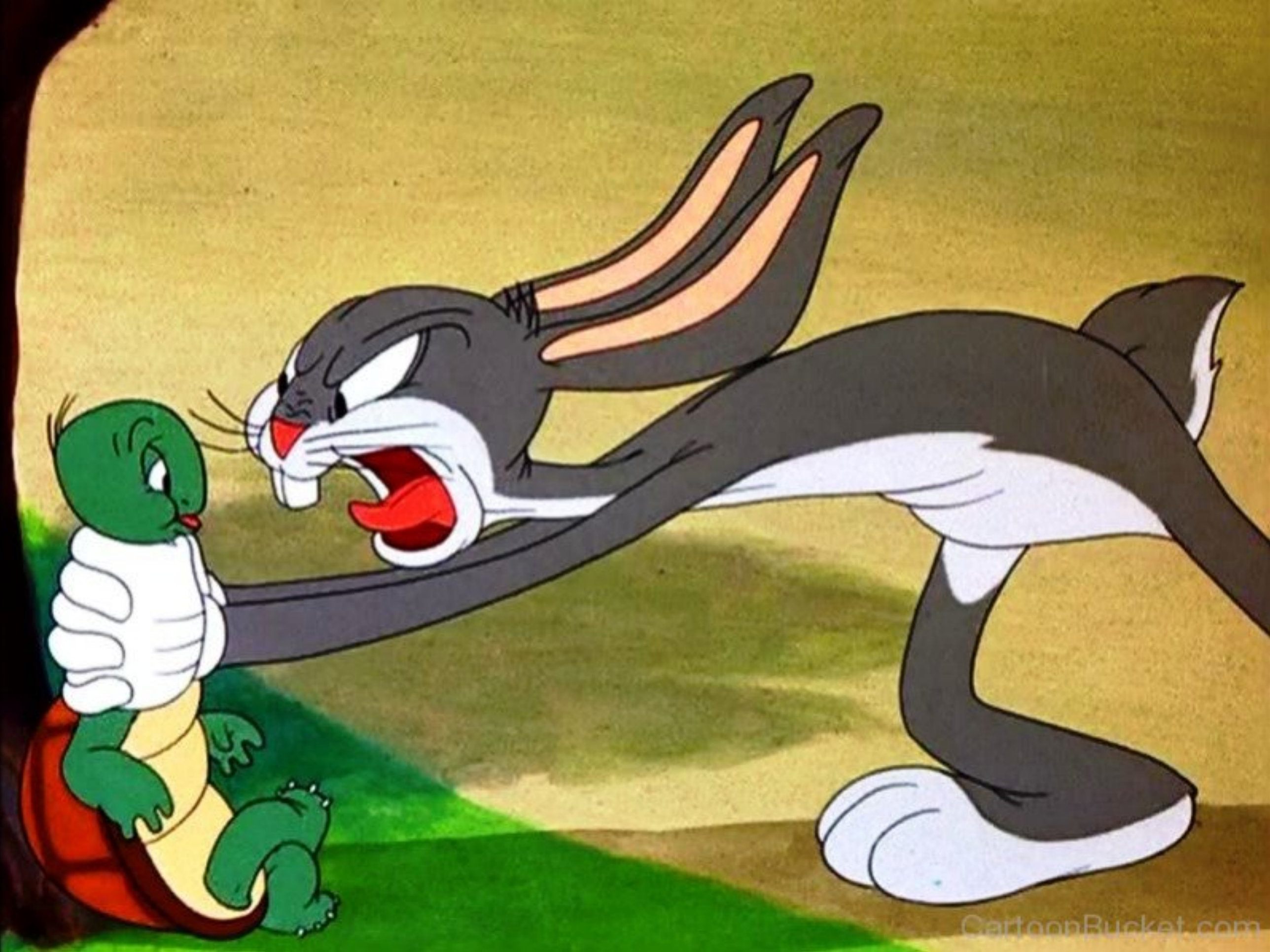
```
function makeBoard(n) {
    let board = new Array(spaces);
    cursor = 0;
    board.addTo = (item, number) => {
        board.fill(item, cursor, number + cursor);
        cursor += number;
    };
    board.fill("empty");

    board.addTo("f3", 5); // forward 3 spaces
    board.addTo("b3", 5); // back 3 spaces

    board.addTo("mt", 3); // miss a turn
    board.addTo("ra", 3); // roll again

    board.addTo("cr", 2); // tortoise crashes!
    board.addTo("zz", 3); // hare stops to eat and has a snooze

    shuffle(board);

    return board;
}
```

Board

```
board.fill("empty");

board.addTo("f3", 5); // forward 3 spaces
board.addTo("b3", 5); // back 3 spaces

board.addTo("mt", 3); // miss a turn
board.addTo("ra", 3); // roll again

board.addTo("cr", 2); // tortoise crashes!
board.addTo("zz", 3); // hare stops to eat and has a snooze

shuffle(board);
```

Monte Carlo

```
for (i = 0; i < sims; i++) {
    const board = makeBoard(spaces);
    const [t, h, turns, sleeps, flights] = simulateGame(board);

    totTurns += turns;
    totSleeps += sleeps;
    totFlights += flights;

    if (t === h) {
        totDraws += 1;
        addBoardToDrawBoardsArray(drawBoards, board);
    } else {
        tortoiseWins += t;
        hareWins += h;
    }
}
```


CartoonBucket.com

Demo 2


That's all Folks!