Guess and Check Algorithm

The **guess and check algorithm** is a methodical approach to problem-solving where you:

1. Make an initial guess for the solution.
2. Check if the guess satisfies the given condition(s).
3. Adjust the guess if it's incorrect.
4. Repeat the process until you find the solution or exhaust all possibilities.

Steps for the Guess and Check Algorithm

1. **Make an Initial Guess:** Start with a reasonable value or range.
2. **Check the Guess:** Verify if it meets the conditions of the problem.
3. **Adjust the Guess:** Modify the guess based on feedback.
4. **Repeat:** Continue until the correct solution is found.

This method is useful when:

- Direct calculation of the solution is not possible.
- Constraints need to be satisfied.
- Computational complexity makes other methods inefficient.

Example 1: Finding the Square Root (Approximation)

```
# Goal: Find the square root of 50 using guess and check
target = 50
guess = 0  # Start with an initial guess
tolerance = 0.01  # How close we want the result to be

while abs(guess**2 - target) > tolerance:  # Check if the guess is
close enough
    guess += 0.01  # Increment the guess in small steps

print(f"The approximate square root of {target} is {guess}")
```

Example 2: Solving a Puzzle

```
# Goal: Find a number between 1 and 100 divisible by both 3 and 7
for number in range(1, 101):  # Check numbers between 1 and 100
    if number % 3 == 0 and number % 7 == 0:  # Check divisibility
        print(f"The number {number} is divisible by both 3 and 7.")
        break  # Stop once the first solution is found
```

Exercises

1. **Guess the cube root:** Write a program to approximate the cube root of 27 using guess and check.
2. **Number puzzle:** Find the smallest number between 1 and 200 that is divisible by 5, 6, and 8.
3. **Word guessing game:** Implement a simple guessing game where the user tries to guess a predefined word.