Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

9. Prime Number

```
[2] def sum_of_digits(n):
        return sum(int(digit) for digit in str(n))

    # Example usage
    num = int(input("Enter a number: "))
    print(sum_of_digits(num))
```

```
Enter a number: 35
8
```

10. Sum of Digits

```
import math

def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

num = int(input("Enter a number: "))
print(is_prime(num))
```

```
Enter a number: 22
False
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

11. LCM and GCD

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def lcm(a, b):
    return (a * b) // gcd(a, b)

# Example usage
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("GCD:", gcd(a, b))
print("LCM:", lcm(a, b))
```

```
Enter first number: 25
Enter second number: 55
GCD: 5
LCM: 275
```

12. List Reversal

```
[11] def reverse_list(lst):
        return lst[::-1]

# Example usage
lst = list(map(int, input("Enter numbers separated by space: ").split()))
print(reverse_list(lst))
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

[11]

```
Enter numbers separated by space: 2 5 8 9
[9, 8, 5, 2]
```

13. Sort a List

[4]

```
def sort_list(lst):
    return sorted(lst)

# Example usage
lst = list(map(int, input("Enter numbers separated by space: ").split()))
print(sort_list(lst))
```

```
Enter numbers separated by space: 2 5 6 7
[2, 5, 6, 7]
```

14. Remove Duplicates

```
def remove_duplicates(lst):
    return list(set(lst))

# Example usage
lst = list(map(int, input("Enter numbers separated by space: ").split()))
print(remove_duplicates(lst))
```

```
Enter numbers separated by space: 2 5 5 8 5
[8, 2, 5]
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

15. String Length

```
[9] def string_length(s):
        count = 0
        for _ in s:
            count += 1
        return count

    # Example usage
    s = input("Enter a string: ")
    print(string_length(s))
```

```
Enter a string: hi
2
```

Maze Generator and Solver

Maze Generator and Solver

```
[10] import random
     from collections import deque

     # Maze dimensions (small)
     ROWS, COLS = 9, 9

     # Directions for movement (Up, Down, Left, Right)
     DIRECTIONS = [(-2, 0), (2, 0), (0, -2), (0, 2)]

     def create_maze():
         """Creates a small grid filled with walls (1s)"""
         maze = [[1 for _ in range(COLS)] for _ in range(ROWS)]
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

```
    def dfs(x, y):
        maze[x][y] = 0
        random.shuffle(DIRECTIONS)
        for dx, dy in DIRECTIONS:
            nx, ny = x + dx, y + dy
            wx, wy = x + dx // 2, y + dy // 2
            if 1 <= nx < ROWS-1 and 1 <= ny < COLS-1 and maze[nx][ny] == 1:
                maze[wx][wy] = 0
                dfs(nx, ny)
    dfs(1, 1)
    return maze

def print_maze(maze):
    """Prints the maze using ▉ for walls and spaces for paths"""
    for row in maze:
        print("".join("▉" if cell == 1 else " " for cell in row))

def solve_maze(maze, start, end):
    """BFS-based solver"""
    queue = deque([(*start, [])])
    visited = set()
    while queue:
        x, y, path = queue.popleft()
        if (x, y) == end:
            return path + [(x, y)]
        if (x, y) in visited:
            continue
        visited.add((x, y))
        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            nx, ny = x + dx, y + dy
            if 0 <= nx < ROWS and 0 <= ny < COLS and maze[nx][ny] == 0:
                queue.append((nx, ny, path + [(x, y)]))
    return None
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

```
def solve_maze(maze, start, end):
    """BFS-based solver"""
    queue = deque([(*start, [])])
    visited = set()
    while queue:
        x, y, path = queue.popleft()
        if (x, y) == end:
            return path + [(x, y)]
        if (x, y) in visited:
            continue
        visited.add((x, y))
        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            nx, ny = x + dx, y + dy
            if 0 <= nx < ROWS and 0 <= ny < COLS and maze[nx][ny] == 0:
                queue.append((nx, ny, path + [(x, y)]))
    return None

# Generate and Solve the Maze
maze = create_maze()
start, end = (1, 1), (ROWS-2, COLS-2)
solution = solve_maze(maze, start, end)

print("Generated Maze:")
print_maze(maze)

if solution:
    for x, y in solution:
        maze[x][y] = 2
    print("\nSolved Maze:")
    for row in maze:
        print("".join("" if cell == 1 else ("*" if cell == 2 else " ") for cell in row))
else:
    print("\nNo solution found!")
```

✓ 9s completed at 10:52 PM

Untitled18.ipynb - Colab

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text

RAM Disk

Gemini

Generated Maze:

Solved Maze:

✓ 9s completed at 10:52 PM