LAB-16

Permutations and combinations

CODE:

```
import itertools

def permutations(li):
    return list(itertools.permutations(li))

def combinations(li, r):
    return list(itertools.combinations(li, r))

li = [1, 2, 3]

perms = permutations(li)

print("Permutations:")

for p in perms:
    print(p)

r = 2

combs = combinations(li, r)

print("\nCombinations:")

for c in combs:
    print(c)
```

OUTPUT:

```
>>>
    = RESTART: C:/Users/bored/AppData/Local/Programs/Python/Python312/permutations.p
    y
    Permutations:
    (1, 2, 3)
    (1, 3, 2)
    (2, 1, 3)
    (2, 3, 1)
    (3, 1, 2)
    (3, 2, 1)

    Combinations:
    (1, 2)
    (1, 3)
    (2, 3)
>>>
```

Subset generation

CODE:

```
import itertools

def subsets(nums):
```

```
    subsets=[]
    for r in range(len(nums) + 1):
        subsets.extend(itertools.combinations(nums, r))
    return subsets
nums=[1, 2, 3]
subsets=subsets(nums)
print("Subsets")
for s in subsets:
    print(s)
```

OUTPUT:

```
>>>
    = RESTART: C:/Users/bored/AppData/Local/Programs/Python/Python312/subset generat
    ion.py
    Subsets
    ()
    (1,)
    (2,)
    (3,)
    (1, 2)
    (1, 3)
    (2, 3)
    (1, 2, 3)
>>>
```

Hamiltonian cycle

CODE:

```
def is_valid(vertex, graph, path, pos):
    if graph[path[pos - 1]][vertex] == 0:
        return False
    if vertex in path:
        return False
    return True
def hamiltonian_cycle_util(graph, path, pos):
    if pos == len(graph):
        if graph[path[pos - 1]][path[0]] == 1:
            return True
        else:
            return False
```

```
    for v in range(1, len(graph)):
        if is_valid(v, graph, path, pos):
            path[pos] = v
            if hamiltonian_cycle_util(graph, path, pos + 1):
                return True
            path[pos] = -1
    return False
def hamiltonian_cycle(graph):
    n = len(graph)
    path = [-1] * n
    path[0] = 0
    if not hamiltonian_cycle_util(graph, path, 1):
        print("No Hamiltonian cycle exists")
        return False
    print("Hamiltonian cycle exists:")
    print(path + [path[0]])
    return True
graph = [
    [0, 1, 0, 1, 0],
    [1, 0, 1, 1, 1],
    [0, 1, 0, 0, 1],
    [1, 1, 0, 0, 1],
    [0, 1, 1, 1, 0]
]
hamiltonian_cycle(graph)
```

OUTPUT:

```
>>>
= RESTART: C:/Users/bored/AppData/Local/Programs/Python/Python312/hamiltonian cy
cle.py
Hamiltonian cycle exists:
[0, 1, 2, 4, 3, 0]
>>>
```

Sudoku solver

CODE:

```
def print_board(board):
    for row in board:
        print(" ".join(map(str, row)))
def find_empty_location(board):
    for i in range(9):
        for j in range(9):
            if board[i][j] == 0:
                return (i, j)
    return None
def is_valid(board, num, pos):
    for i in range(9):
        if board[pos[0]][i] == num and i != pos[1]:
            return False
    for i in range(9):
        if board[i][pos[1]] == num and i != pos[0]:
            return False
    box_x = pos[1] // 3
    box_y = pos[0] // 3
    for i in range(box_y * 3, box_y * 3 + 3):
        for j in range(box_x * 3, box_x * 3 + 3):
            if board[i][j] == num and (i, j) != pos:
                return False
    return True
def solve_sudoku(board):
    empty = find_empty_location(board)
    if not empty:
        return True
    row, col = empty
    for num in range(1, 10):
        if is_valid(board, num, (row, col)):
            board[row][col] = num
```

```
            if solve_sudoku(board):
                return True
            board[row][col] = 0
    return False
board = [
    [5, 3, 0, 0, 7, 0, 0, 0, 0],
    [6, 0, 0, 1, 9, 5, 0, 0, 0],
    [0, 9, 8, 0, 0, 0, 0, 6, 0],
    [8, 0, 0, 0, 6, 0, 0, 0, 3],
    [4, 0, 0, 8, 0, 3, 0, 0, 1],
    [7, 0, 0, 0, 2, 0, 0, 0, 6],
    [0, 6, 0, 0, 0, 0, 2, 8, 0],
    [0, 0, 0, 4, 1, 9, 0, 0, 5],
    [0, 0, 0, 0, 8, 0, 0, 7, 9]
]
print("Sudoku puzzle to solve:")
print_board(board)
print("\nSolving...\n")
if solve_sudoku(board):
    print("Sudoku solved:")
    print_board(board)
else:
    print("No solution exists.")
```

OUTPUT:

```
>>>
= RESTART: C:/Users/bored/AppData/Local/Programs/Python/Python312/sudoku solver.
py
Sudoku puzzle to solve:
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9

Solving...

Sudoku solved:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
>>>
```