```
n = 8
target_deg = 3
vertices = list(range(n))

def all_cubic_multigraphs(n):
    result = []
    edgeset = [(i,j) for i in range(n) for j in range(i+1,n)]
    def backtrack(idx, degs, chosen_edges):
        if idx == len(edgeset):
            if all(d == target_deg for d in degs):
                G = Graph(multiedges=True, loops=True)
                G.add_vertices(range(n))
                for (u,v),m in chosen_edges.items():
                    for _ in range(m):
                        G.add_edge(u,v)
                if G.is_connected():
                    result.append(G)
            return
        u,v = edgeset[idx]
        max_mult = min(target_deg - degs[u], target_deg -
degs[v])
        for m in range(max_mult + 1):
            if m == 0:
                chosen_edges[(u,v)] = 0
                backtrack(idx+1, degs, chosen_edges)
                del chosen_edges[(u,v)]
            else:
                degs2 = list(degs)
                degs2[u] += m
```

```
                degs2[v] += m
                chosen_edges[(u,v)] = m
                backtrack(idx+1, degs2, chosen_edges)
                del chosen_edges[(u,v)]

    backtrack(0, [0]*n, {})
    return result

M = all_cubic_multigraphs(n)
print("Total connected cubic multigraphs (possibly
isomorphic):", len(M))

unique = []
for G in M:
    if not any(G.is_isomorphic(H) for H in unique):
        unique.append(G)
print("Distinct up to isomorphism:", len(unique))

planar_multis = [G for G in unique if G.is_planar()]
print("Planar cubic multigraphs (non-isomorphic):",
len(planar_multis))

for i, G in enumerate(planar_multis):
    show(G)
```

解释：

定义的函数backtrack(idx, degs, chosen_edges)是用找到的“点、点的度数、边”来在sagemath中生成multigraph，

并检查这个图是否符合要求，而这些“点、点的度数、边”是通过遍历每一条边(I, j)可能重复的次数（0，1，2，3次）得到的，也就是代码里“max_mult = min(target_deg - degs[u], target_deg - degs[v])
for m in range(max_mult + 1):”所做的事情