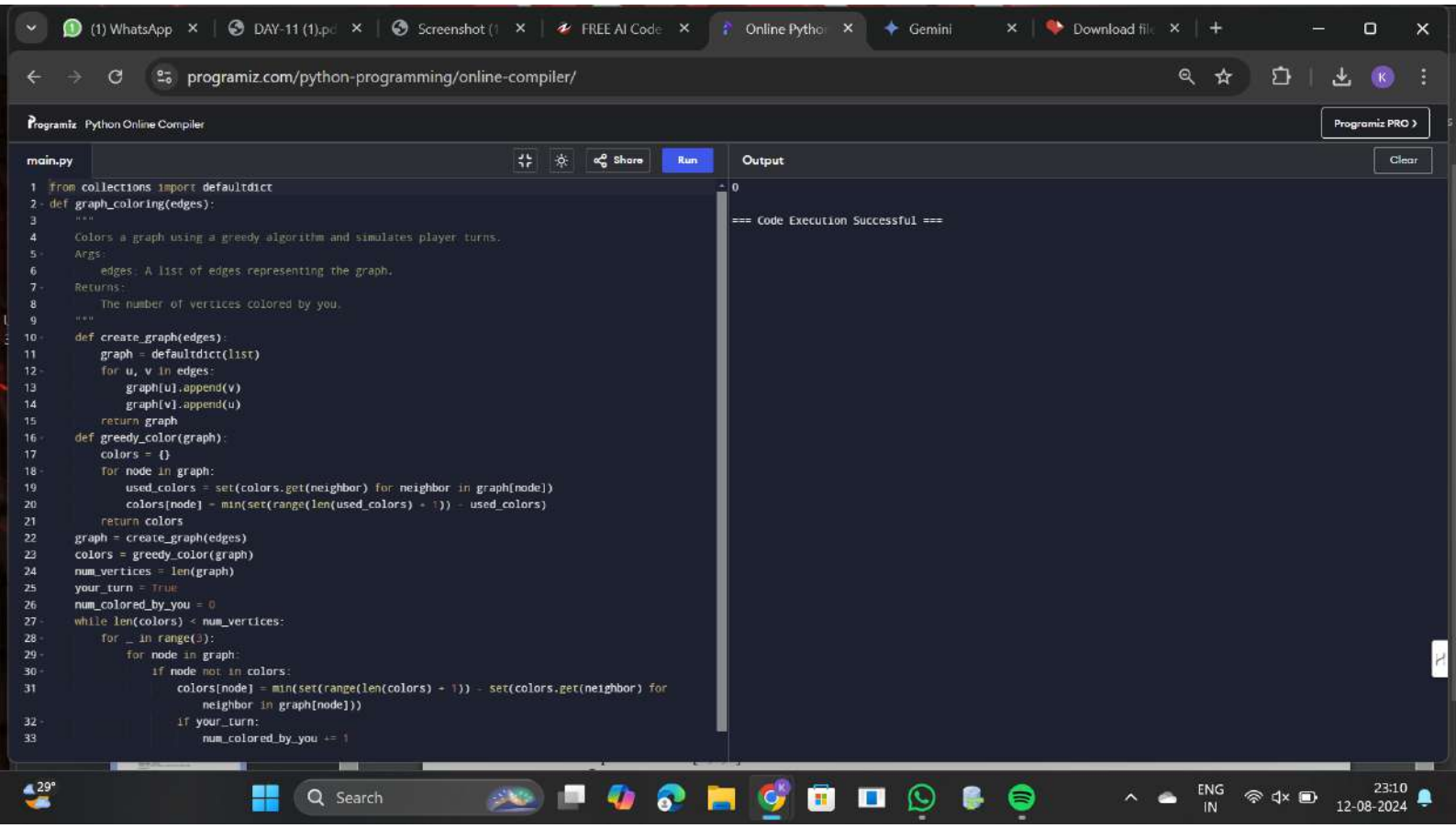
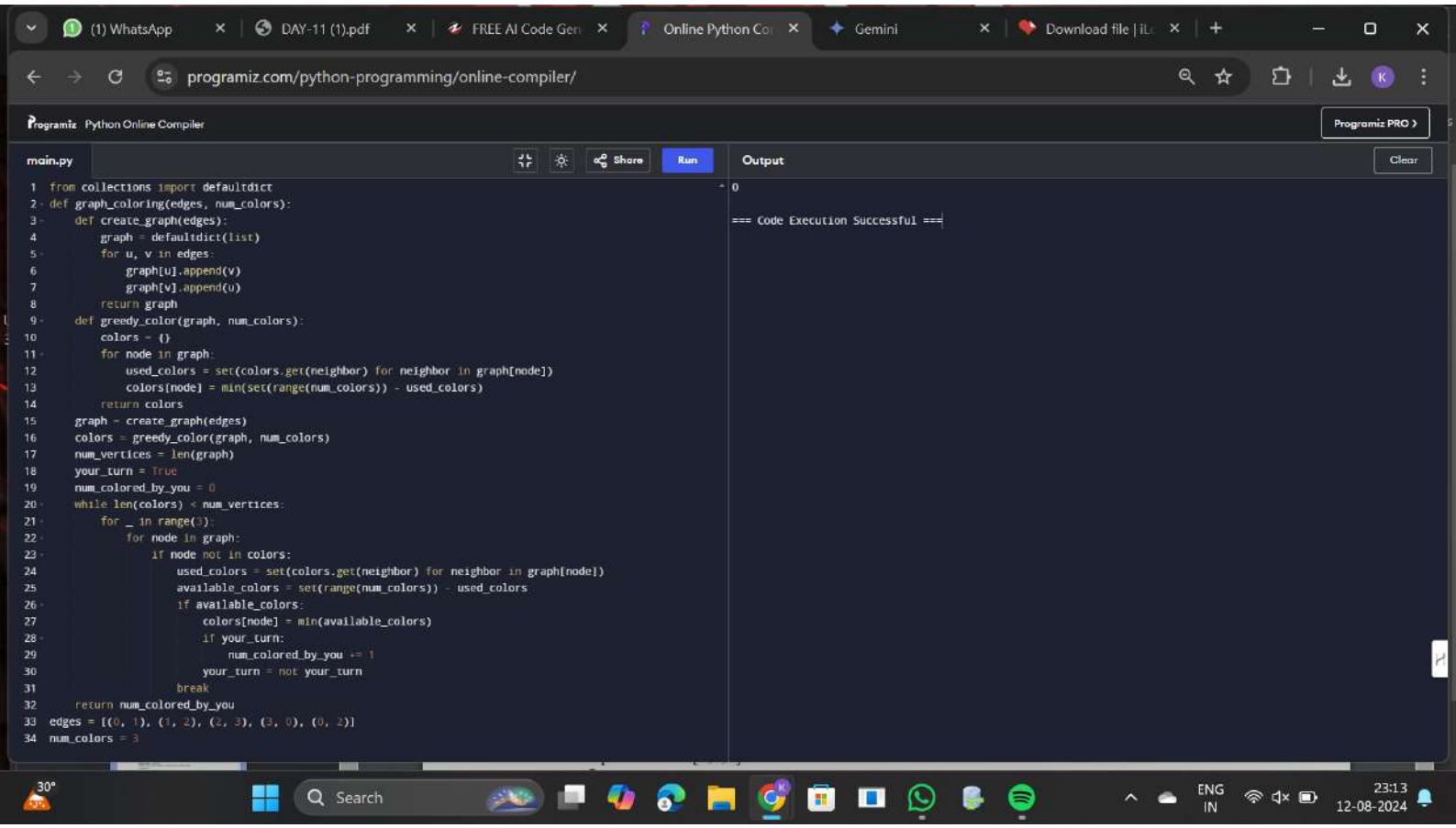
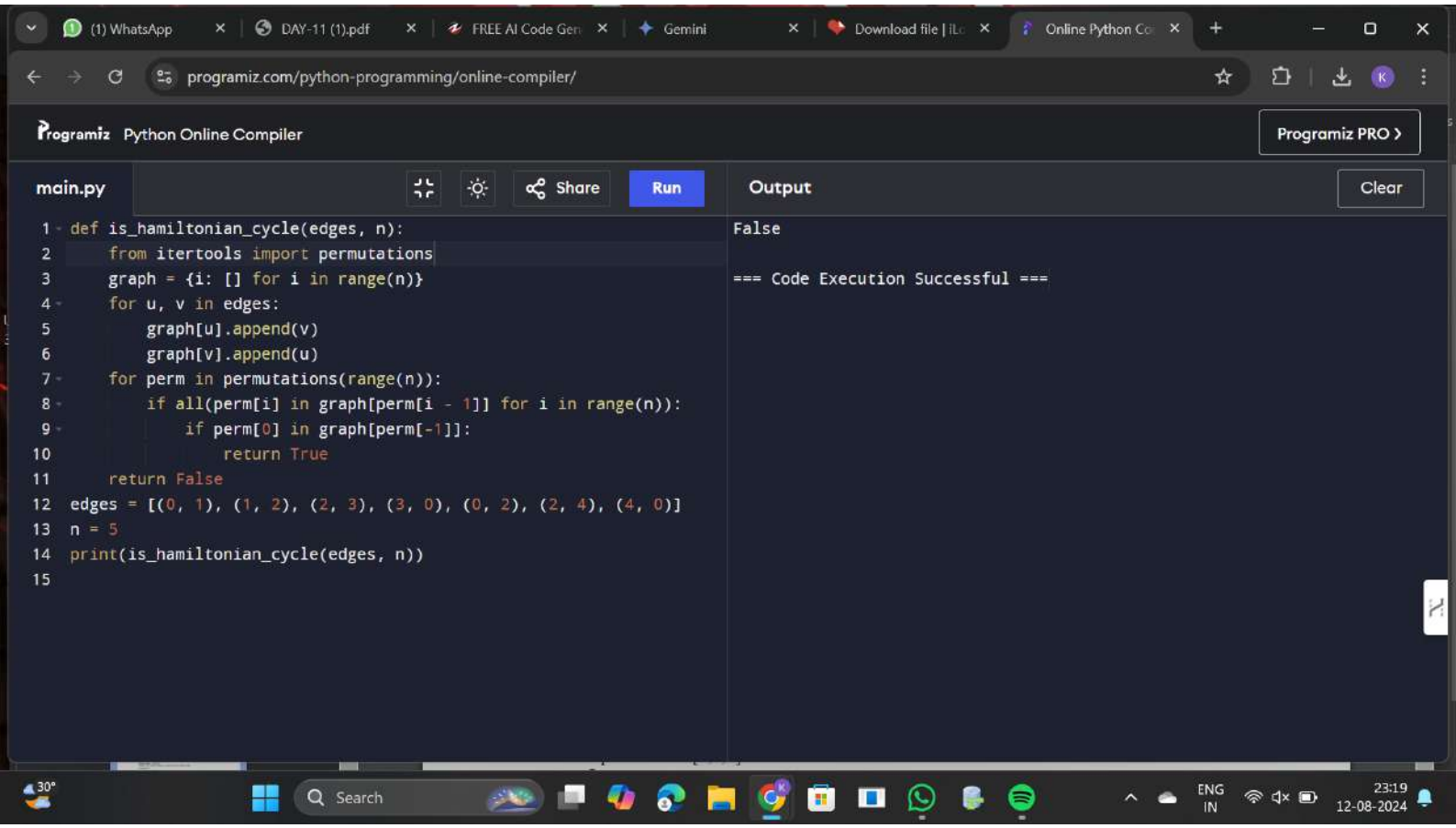
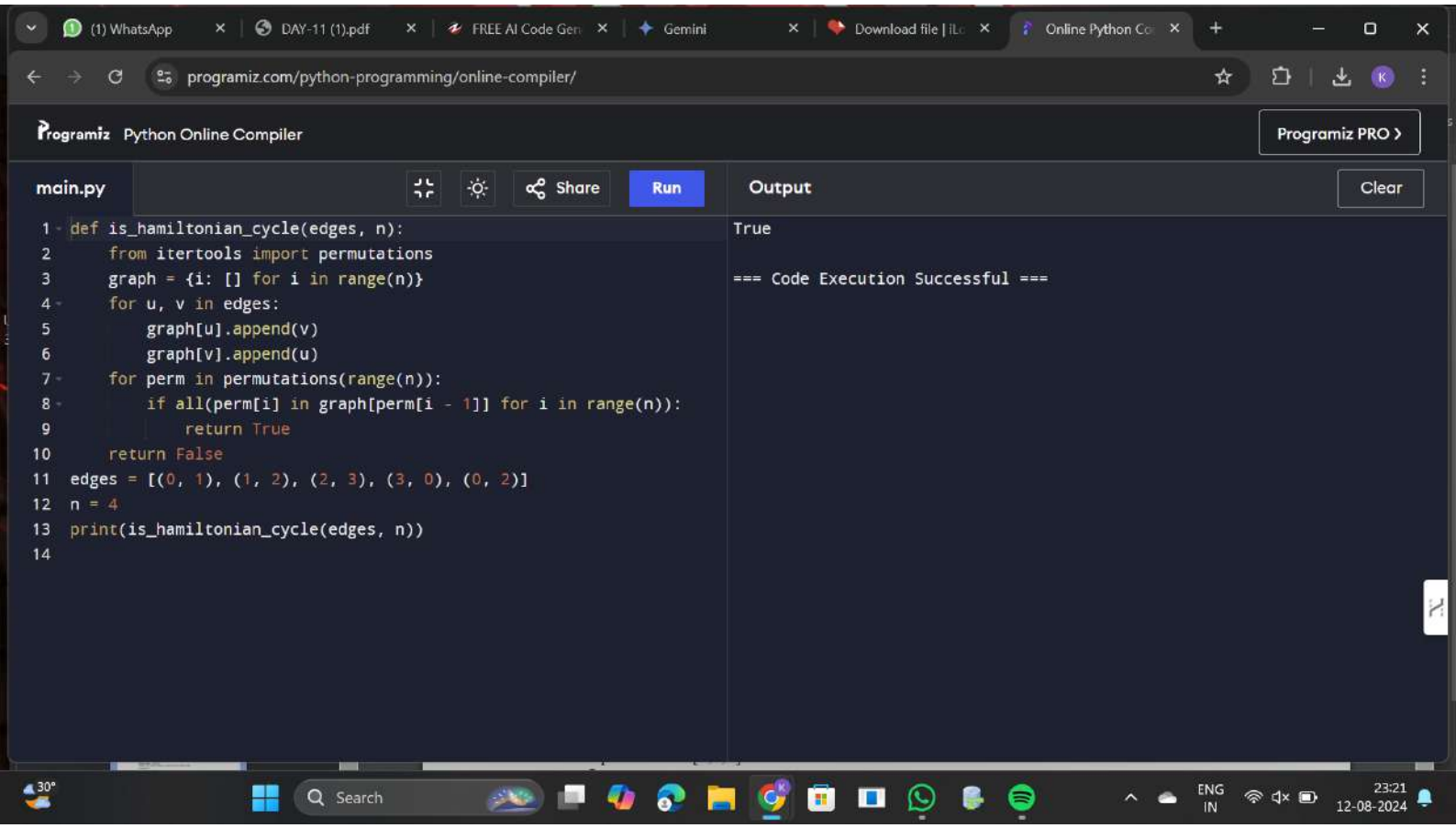
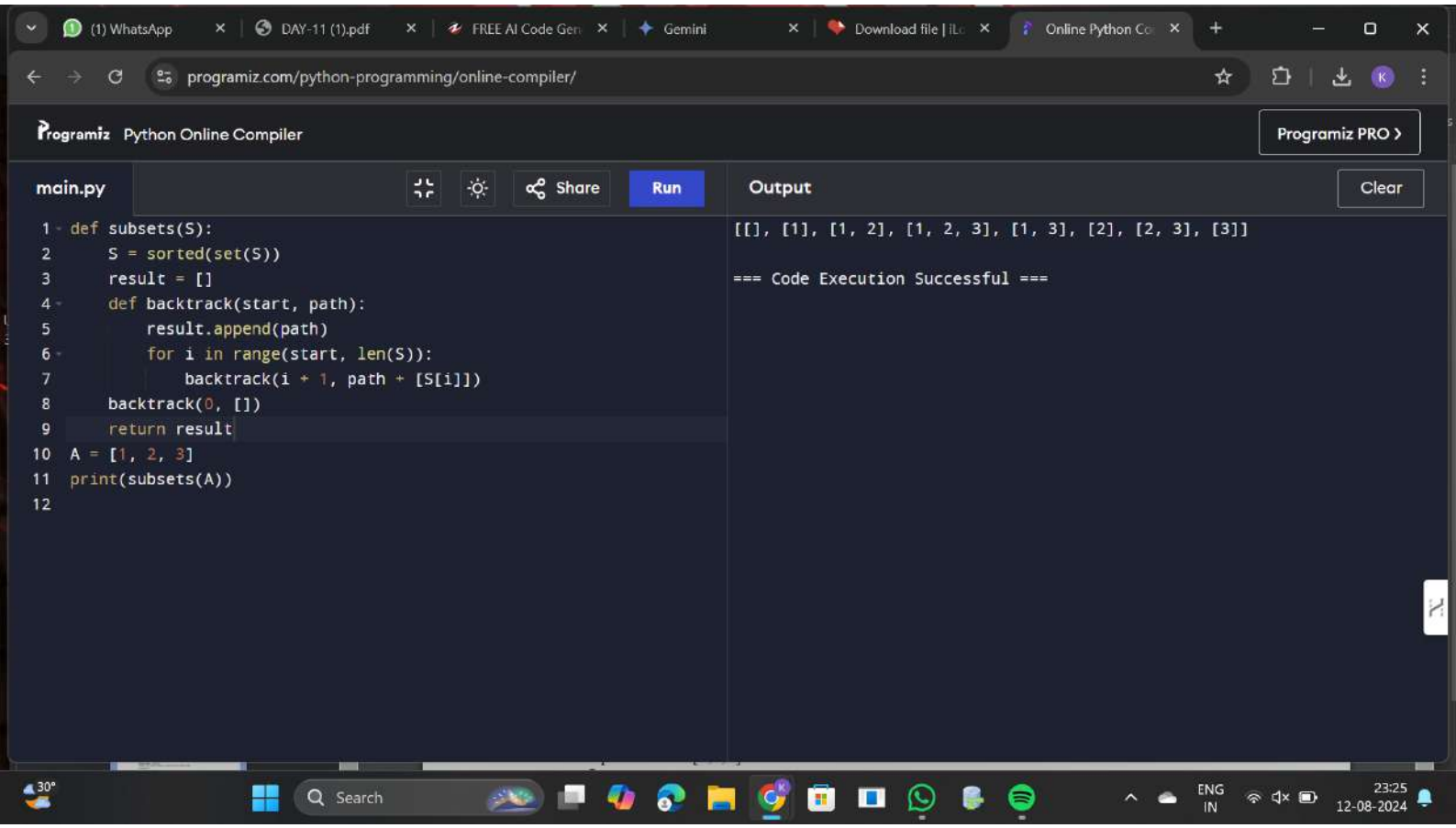
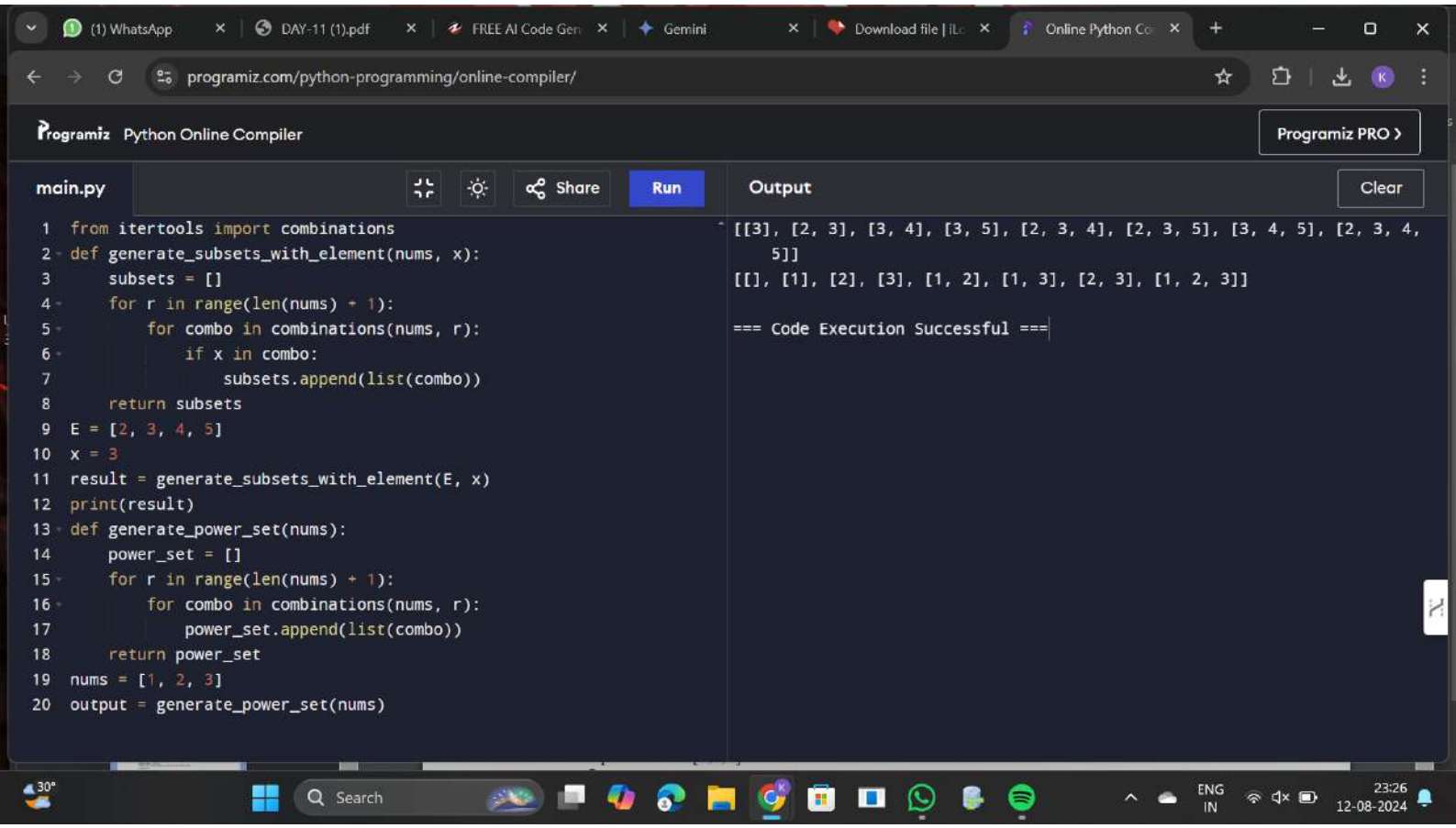

programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
from collections import defaultdict
def graph_coloring(edges):
"""
Colors a graph using a greedy algorithm and simulates player turns.
Args:
edges: A list of edges representing the graph.
Returns:
The number of vertices colored by you.
"""
def create_graph(edges):
graph = defaultdict(list)
for u, v in edges:
graph[u].append(v)
graph[v].append(u)
return graph
def greedy_color(graph):
colors = {}
for node in graph:
used_colors = set(colors.get(neighbor) for neighbor in graph[node])
colors[node] = min(set(range(len(used_colors) + 1)) - used_colors)
return colors
graph = create_graph(edges)
colors = greedy_color(graph)
num_vertices = len(graph)
your_turn = True
num_colored_by_you = 0
while len(colors) < num_vertices:
for _ in range(3):
for node in graph:
if node not in colors:
colors[node] = min(set(range(len(colors) + 1)) - set(colors.get(neighbor) for
neighbor in graph[node]))
if your_turn:
num_colored_by_you += 1
0
=== Code Execution Successful ===

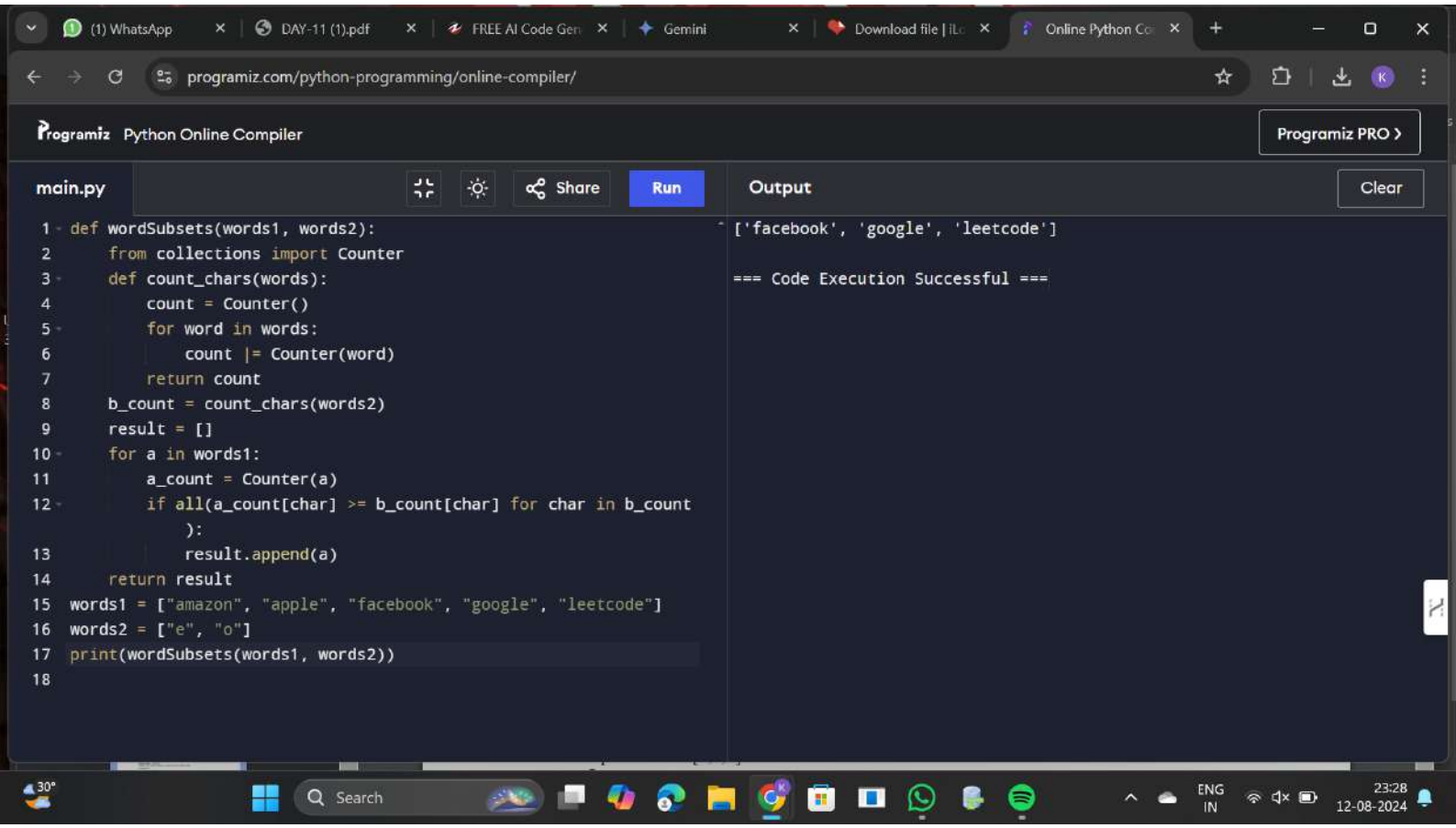

(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Online Python Cor
Gemini
Download file | iL
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
from collections import defaultdict
def graph_coloring(edges, num_colors):
def create_graph(edges):
graph = defaultdict(list)
for u, v in edges:
graph[u].append(v)
graph[v].append(u)
return graph
def greedy_color(graph, num_colors):
colors = {}
for node in graph:
used_colors = set(colors.get(neighbor) for neighbor in graph[node])
colors[node] = min(set(range(num_colors)) - used_colors)
return colors
graph = create_graph(edges)
colors = greedy_color(graph, num_colors)
num_vertices = len(graph)
your_turn = True
num_colored_by_you = 0
while len(colors) < num_vertices:
for _ in range(3):
for node in graph:
if node not in colors:
used_colors = set(colors.get(neighbor) for neighbor in graph[node])
available_colors = set(range(num_colors)) - used_colors
if available_colors:
colors[node] = min(available_colors)
if your_turn:
num_colored_by_you += 1
your_turn = not your_turn
break
return num_colored_by_you
edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]
num_colors = 3
0
=== Code Execution Successful ===
30°
Search
ENG
IN
23:13
12-08-2024


(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Gemini
Download file | iLo
Online Python Co
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def is_hamiltonian_cycle(edges, n):
from itertools import permutations
graph = {i: [] for i in range(n)}
for u, v in edges:
graph[u].append(v)
graph[v].append(u)
for perm in permutations(range(n)):
if all(perm[i] in graph[perm[i - 1]] for i in range(n)):
if perm[0] in graph[perm[-1]]:
return True
return False
edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2), (2, 4), (4, 0)]
n = 5
print(is_hamiltonian_cycle(edges, n))
False
=== Code Execution Successful ===
30°
Search
ENG
IN
23:19
12-08-2024


(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Gemini
Download file | iLo
Online Python Co
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def is_hamiltonian_cycle(edges, n):
from itertools import permutations
graph = {i: [] for i in range(n)}
for u, v in edges:
graph[u].append(v)
graph[v].append(u)
for perm in permutations(range(n)):
if all(perm[i] in graph[perm[i - 1]] for i in range(n)):
return True
return False
edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]
n = 4
print(is_hamiltonian_cycle(edges, n))
True
=== Code Execution Successful ===
30°
Search
ENG
IN
23:21
12-08-2024


(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Gemini
Download file | iLo
Online Python Co
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def subsets(S):
S = sorted(set(S))
result = []
def backtrack(start, path):
result.append(path)
for i in range(start, len(S)):
backtrack(i + 1, path + [S[i]])
backtrack(0, [])
return result
A = [1, 2, 3]
print(subsets(A))
[[], [1], [1, 2], [1, 2, 3], [1, 3], [2], [2, 3], [3]]
=== Code Execution Successful ===
30°
Search
ENG
IN
23:25
12-08-2024


(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Gemini
Download file | iLo
Online Python Co
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
from itertools import combinations
def generate_subsets_with_element(nums, x):
subsets = []
for r in range(len(nums) + 1):
for combo in combinations(nums, r):
if x in combo:
subsets.append(list(combo))
return subsets
E = [2, 3, 4, 5]
x = 3
result = generate_subsets_with_element(E, x)
print(result)
def generate_power_set(nums):
power_set = []
for r in range(len(nums) + 1):
for combo in combinations(nums, r):
power_set.append(list(combo))
return power_set
nums = [1, 2, 3]
output = generate_power_set(nums)
[[3], [2, 3], [3, 4], [3, 5], [2, 3, 4], [2, 3, 5], [3, 4, 5], [2, 3, 4, 5]]
[[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]
=== Code Execution Successful ===
30°
Search
ENG
IN
23:26
12-08-2024


(1) WhatsApp
DAY-11 (1).pdf
FREE AI Code Gen
Gemini
Download file | iLo
Online Python Co
programiz.com/python-programming/online-compiler/
Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def wordSubsets(words1, words2):
from collections import Counter
def count_chars(words):
count = Counter()
for word in words:
count |= Counter(word)
return count
b_count = count_chars(words2)
result = []
for a in words1:
a_count = Counter(a)
if all(a_count[char] >= b_count[char] for char in b_count
):
result.append(a)
return result
words1 = ["amazon", "apple", "facebook", "google", "leetcode"]
words2 = ["e", "o"]
print(wordSubsets(words1, words2))
['facebook', 'google', 'leetcode']
=== Code Execution Successful ===
Search
ENG
IN
23:28
12-08-2024