In [ ]:

```
(hash('Milan')%3)+1
```

Out[ ]:

```
3
```

Problem Description

Given a list of integers representing a range [0, n], the task is to find and return the missing numbers in the range. Handle cases where there are no missing numbers and return -1 in such instances.

# Create 2 new examples

Examples

Example 1:

Input: [3, 1, 0, 4, 6] Output: [2, 5]

Example 2:

Input: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] Output: -1

# Code the solution to the assigned problem in Python

In [ ]:

```
from typing import List

def missing_num(nums: List[int]) -> List[int]:
    if not nums: return [-1]

    n = max(nums)   # Get the maximum number in the list
    num_set = set(nums)

    missing_numbers = []

    for i in range(n+1):
        if i not in num_set:
            missing_numbers.append(i)

    return missing_numbers if missing_numbers else [-1]
```

```python
print(missing_num([0, 2]))  # Output: [1]
print(missing_num([5, 0, 1]))  # Output: [2, 3, 4]
print(missing_num([6, 8, 2, 3, 5, 7, 0, 1, 10]))  # Output: [4, 9]
print(missing_num([3, 1, 0, 4, 6]))
print(missing_num([9, 8, 7, 6, 5, 4, 3, 2, 1, 0]))
```

```
[1]
[2, 3, 4]
[4, 9]
[2, 5]
[-1]
```

RUN Test to verify

In [ ]:

```python
import ipytest
ipytest.autoconfig()


def test_example_1():
    nums_example_1 = [0, 2]
    result_example_1 = missing_num(nums_example_1)
    assert result_example_1 == [1]

def test_example_2():
    nums_example_2 = [5, 0, 1]
    result_example_2 = missing_num(nums_example_2)
    assert result_example_2 == [2, 3, 4]

def test_example_3():
    nums_example_3 = [6, 8, 2, 3, 5, 7, 0, 1, 10]
    result_example_3 = missing_num(nums_example_3)
    assert result_example_3 == [4, 9]

def test_example_4():
    nums_example_4 = [3, 1, 0, 4, 6]
    result_example_4 = missing_num(nums_example_4)
    assert result_example_4 == [2, 5]

def test_example_5():
    nums_example_5 = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
    result_example_5 = missing_num(nums_example_5)
    assert result_example_5 == [-1]

def test_edge_case_empty_list():
    nums_empty = []
    result_empty = missing_num(nums_empty)
    assert result_empty == [-1]

def test_edge_case_no_missing_numbers():
    nums_no_missing = [0, 1, 2, 3, 4]
    result_no_missing = missing_num(nums_no_missing)
    assert result_no_missing == [-1]

def test_edge_case_non_unique_numbers():
    nums_non_unique = [0, 1, 2, 3, 4, 0, 1, 2, 3, 4]
    result_non_unique = missing_num(nums_non_unique)
    assert result_non_unique == [-1]

def test_edge_case_non_unique_numbers_fail():
    nums_non_unique = [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 6]
    result_non_unique = missing_num(nums_non_unique)
    assert result_non_unique == [5, 1]
```

In [ ]:

```
ipytest.run('-vv')
```

```
======================================== test session starts ================
=========================
platform darwin -- Python 3.11.5, pytest-7.4.4, pluggy-1.3.0 -- /usr/local/b
in/python3
cachedir: .pytest_cache
rootdir: /Users/milzbhakta/Documents/PythonProjects/MLFoundations
plugins: mock-3.12.0, Faker-22.2.0
collecting ... collected 9 items

t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_example_1 PASSED
[ 11%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_example_2 PASSED
[ 22%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_example_3 PASSED
[ 33%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_example_4 PASSED
[ 44%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_example_5 PASSED
[ 55%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_edge_case_empty_list PASSED
[ 66%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_edge_case_no_missing_numbers PAS
SED                 [ 77%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_edge_case_non_unique_numbers PAS
SED                 [ 88%]
t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_edge_case_non_unique_numbers_fai
l FAILED            [100%]

============================================= FAILURES =====================
========================
______________________________ test_edge_case_non_unique_numbers_fail ______
________________________

    def test_edge_case_non_unique_numbers_fail():
        nums_non_unique = [0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 6]
        result_non_unique = missing_num(nums_non_unique)
>       assert result_non_unique == [5, 1]
E       AssertionError

/var/folders/1m/2rr7b3011gg02bt4wxfsr3hc0000gn/T/ipykernel_13473/2753363580.
py:48: AssertionError
===================================== short test summary info ==============
========================
FAILED t_5b9caf98afa9408c83bbf8da6b8b9f3e.py::test_edge_case_non_unique_numb
ers_fail - AssertionError
=================================== 1 failed, 8 passed in 0.07s
====================================
```

```
Out[ ]:  <ExitCode.TESTS_FAILED: 1>
```

Explain why the solution works

The solution works by iterating through the range [0, n] and checking for each number if it is present in the input list. If a number is not found, it is considered missing and added to the result list. The function returns the list of missing numbers or -1 if none are found.

Explain time and space complexity

Time Complexity: O(n) The function iterates through the range [0, n] once, checking for the presence of each number in the input list.

Space Complexity: O(n) The space complexity is mainly determined by the set created from the input list for efficient lookup. In the worst case, the set will contain all unique numbers in the input list, resulting in O(n) space.

Explain thinking for an alternative solution

An alternative solution could involve sorting the input list and then iterating through it to find missing numbers. By comparing adjacent elements and identifying gaps, we can determine the missing numbers. This approach might have a different time complexity but could be a viable alternative for larger datasets.