AI Assisted Coding Lab - 2

P.Vishwateja
2303A52389
Batch-44

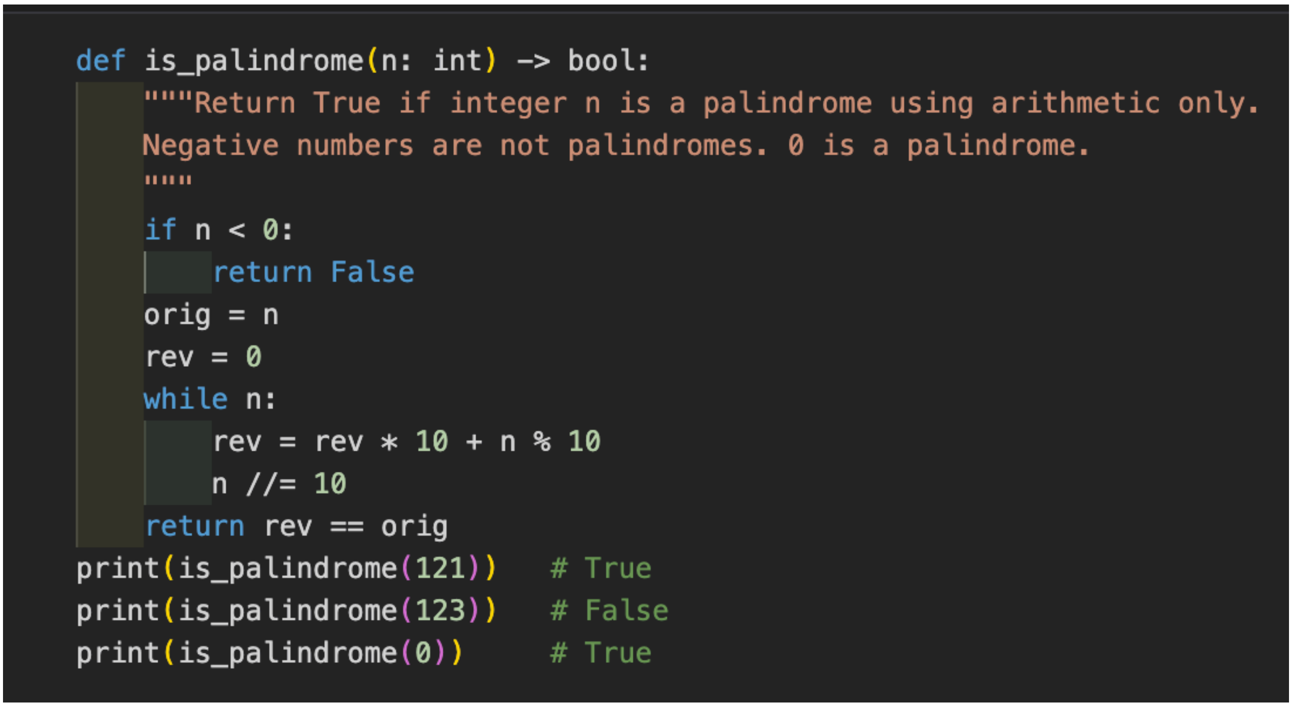
Task-1

Zero-Shot Prompting (Palindrome Number Program)

Prompt:

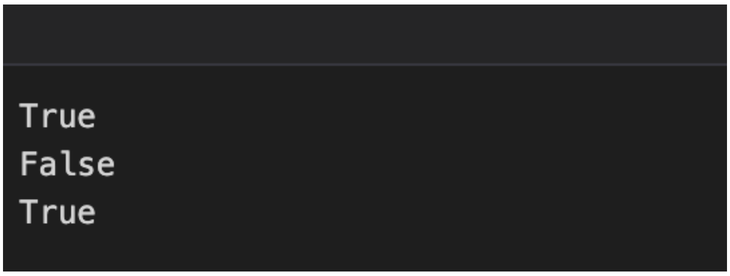
Write a Python function that checks whether a given integer is a palindrome. The function should return True if the number is a palindrome and False otherwise.Do not provide any examples.

Code:



```python
def is_palindrome(n: int) -> bool:
    """Return True if integer n is a palindrome using arithmetic only.
    Negative numbers are not palindromes. 0 is a palindrome.
    """
    if n < 0:
        return False
    orig = n
    rev = 0
    while n:
        rev = rev * 10 + n % 10
        n //= 10
    return rev == orig
print(is_palindrome(121))   # True
print(is_palindrome(123))   # False
print(is_palindrome(0))     # True
```

Output:



```
True
False
True
```

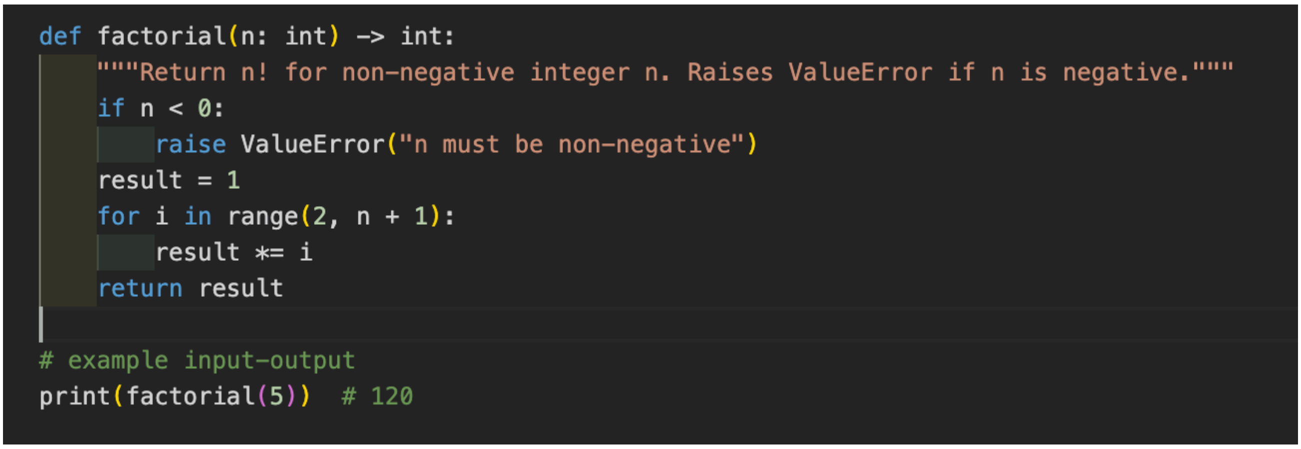
Task-2:

One-Shot Prompting (Factorial Calculation)

Prompt:

Write a Python function to calculate the factorial of a given number.
Example:
Input: 5
Output: 120

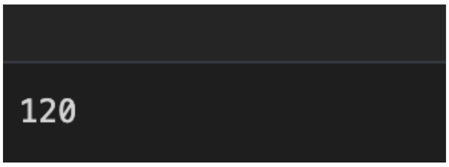
Code:



```python
def factorial(n: int) -> int:
    """Return n! for non-negative integer n. Raises ValueError if n is negative."""
    if n < 0:
        raise ValueError("n must be non-negative")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# example input-output
print(factorial(5))  # 120
```

Output:



```
120
```

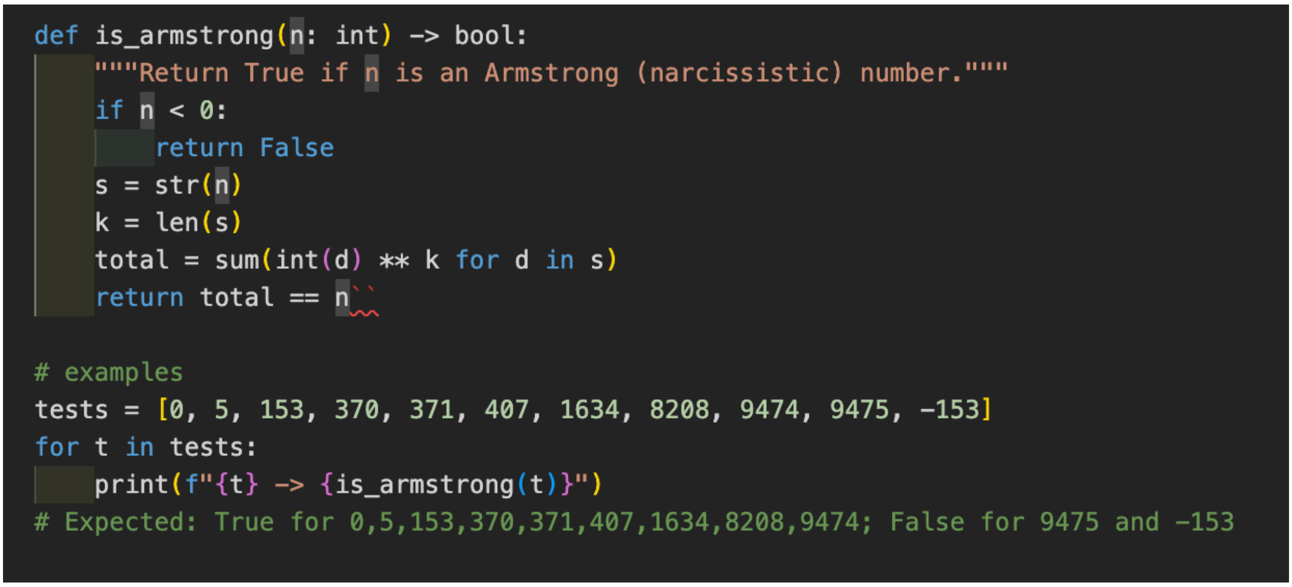
Task-3:

Few-Shot Prompting (Armstrong Number Check)

Prompt:

Write a Python function to check whether a number is an Armstrong number.

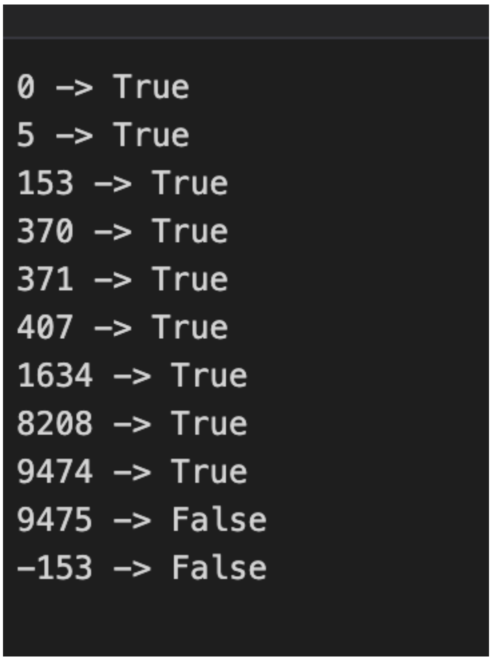
Code:



```python
def is_armstrong(n: int) -> bool:
    """Return True if n is an Armstrong (narcissistic) number."""
    if n < 0:
        return False
    s = str(n)
    k = len(s)
    total = sum(int(d) ** k for d in s)
    return total == n``

# examples
tests = [0, 5, 153, 370, 371, 407, 1634, 8208, 9474, 9475, -153]
for t in tests:
    print(f"{t} -> {is_armstrong(t)}")
# Expected: True for 0,5,153,370,371,407,1634,8208,9474; False for 9475 and -153
```

Output:



```
0 -> True
5 -> True
153 -> True
370 -> True
371 -> True
407 -> True
1634 -> True
8208 -> True
9474 -> True
9475 -> False
-153 -> False
```

Task-4:

Context-Managed Prompting (Optimized Number Classification)

Prompt:

You are an AI coding assistant.

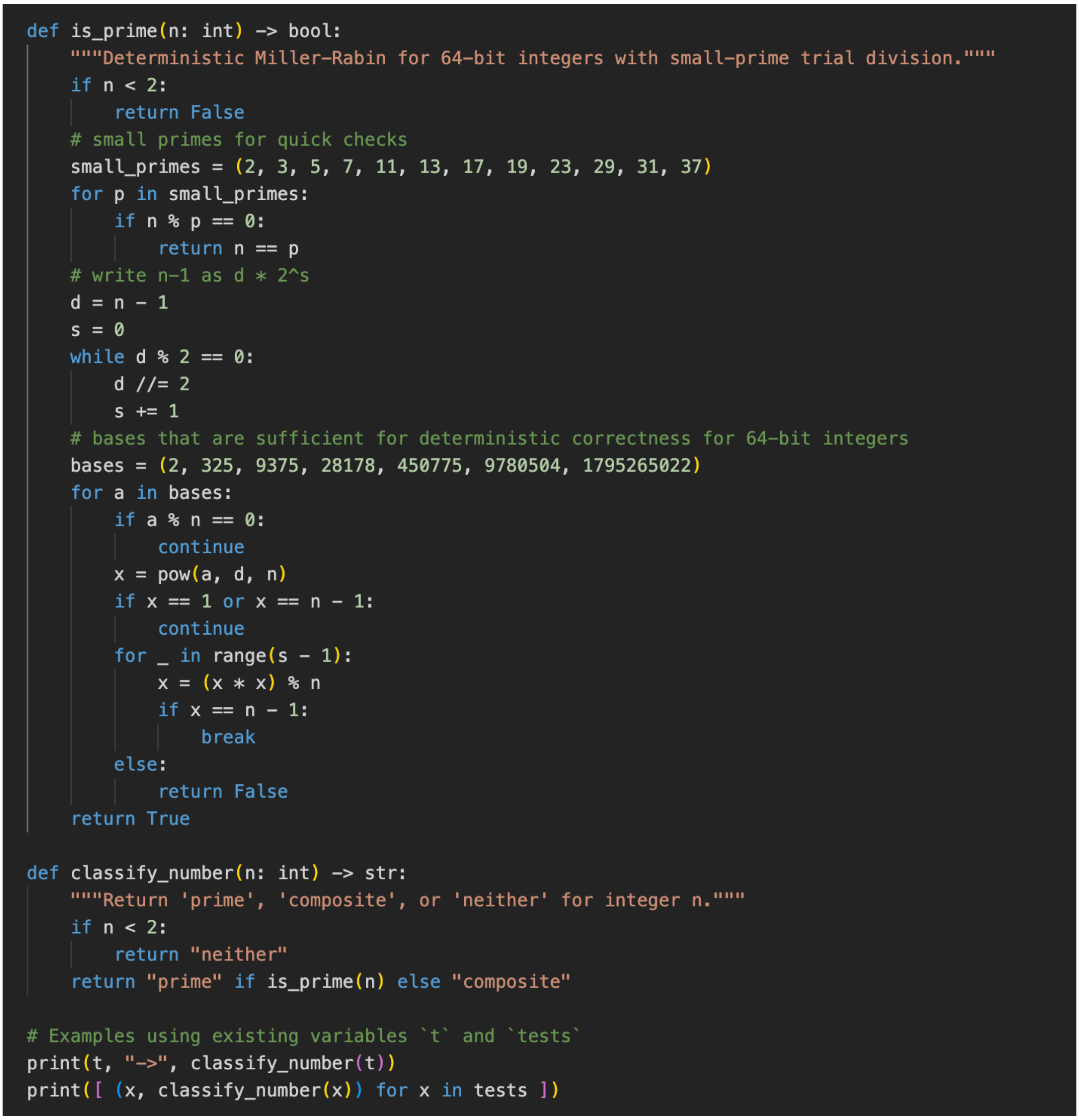
Generate an optimized Python program that classifies a number as:

- Prime
- Composite
- Neither Prime nor Composite

Constraints:

- Validate the input properly
- Handle edge cases like 0, 1, and negative numbers
- Use efficient logic for prime checking
- Return clear output messages

Code:

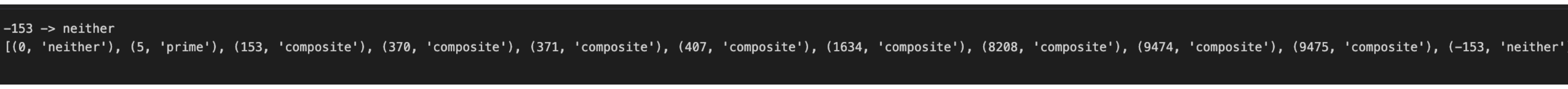


```python
def is_prime(n: int) -> bool:
    """Deterministic Miller-Rabin for 64-bit integers with small-prime trial division."""
    if n < 2:
        return False
    # small primes for quick checks
    small_primes = (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37)
    for p in small_primes:
        if n % p == 0:
            return n == p
    # write n-1 as d * 2^s
    d = n - 1
    s = 0
    while d % 2 == 0:
        d //= 2
        s += 1
    # bases that are sufficient for deterministic correctness for 64-bit integers
    bases = (2, 325, 9375, 28178, 450775, 9780504, 1795265022)
    for a in bases:
        if a % n == 0:
            continue
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue
        for _ in range(s - 1):
            x = (x * x) % n
            if x == n - 1:
                break
        else:
            return False
    return True

def classify_number(n: int) -> str:
    """Return 'prime', 'composite', or 'neither' for integer n."""
    if n < 2:
        return "neither"
    return "prime" if is_prime(n) else "composite"

# Examples using existing variables `t` and `tests`
print(t, "->", classify_number(t))
print([ (x, classify_number(x)) for x in tests ])
```

Output:



```
-153 -> neither
[(0, 'neither'), (5, 'prime'), (153, 'composite'), (370, 'composite'), (371, 'composite'), (407, 'composite'), (1634, 'composite'), (8208, 'composite'), (9474, 'composite'), (9475, 'composite'), (-153, 'neither'
```

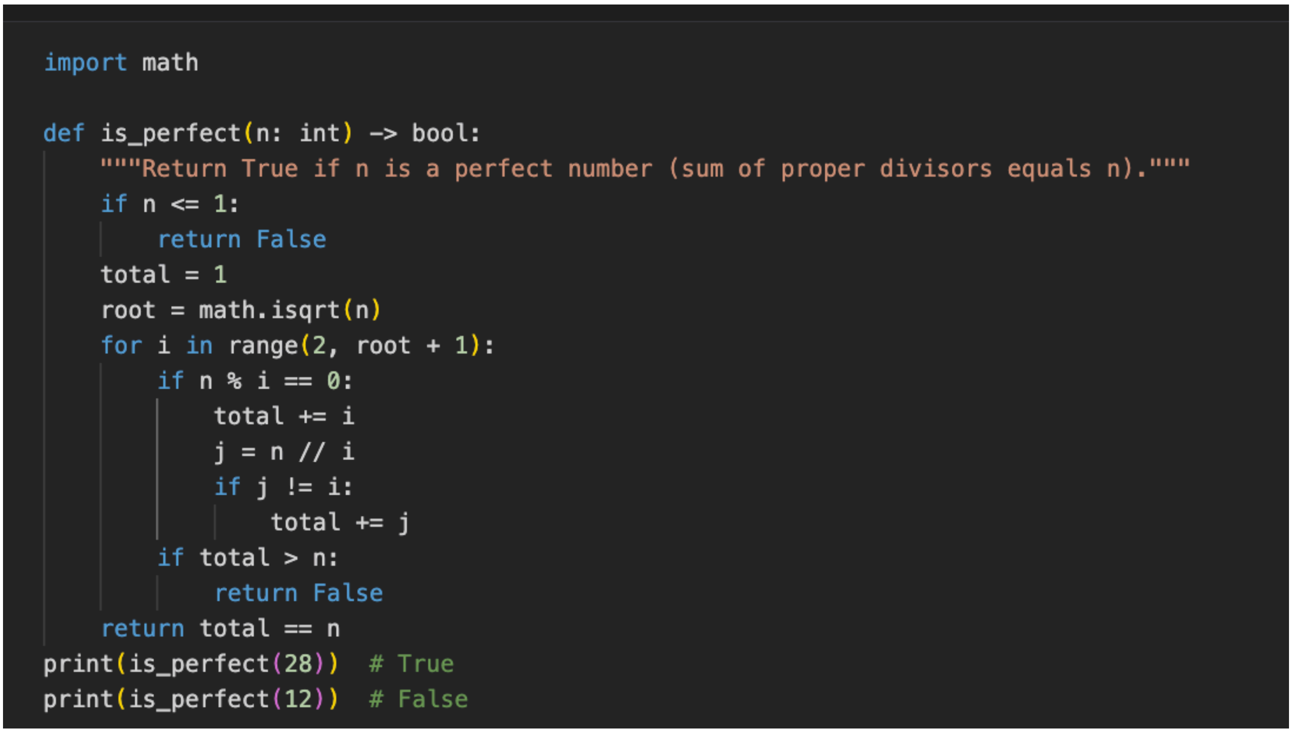
Task-5:

Zero-Shot Prompting (Perfect Number Check)

Prompt:

Write a Python function that checks whether a given number is a perfect number.The function should return True if it is a perfect number and False otherwise. Do not include any examples.

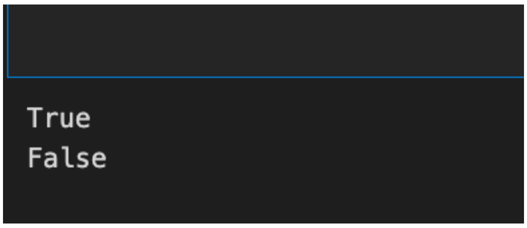
Code:



```python
import math

def is_perfect(n: int) -> bool:
    """Return True if n is a perfect number (sum of proper divisors equals n)."""
    if n <= 1:
        return False
    total = 1
    root = math.isqrt(n)
    for i in range(2, root + 1):
        if n % i == 0:
            total += i
            j = n // i
            if j != i:
                total += j
        if total > n:
            return False
    return total == n
print(is_perfect(28))  # True
print(is_perfect(12))  # False
```

Output:



```
True
False
```

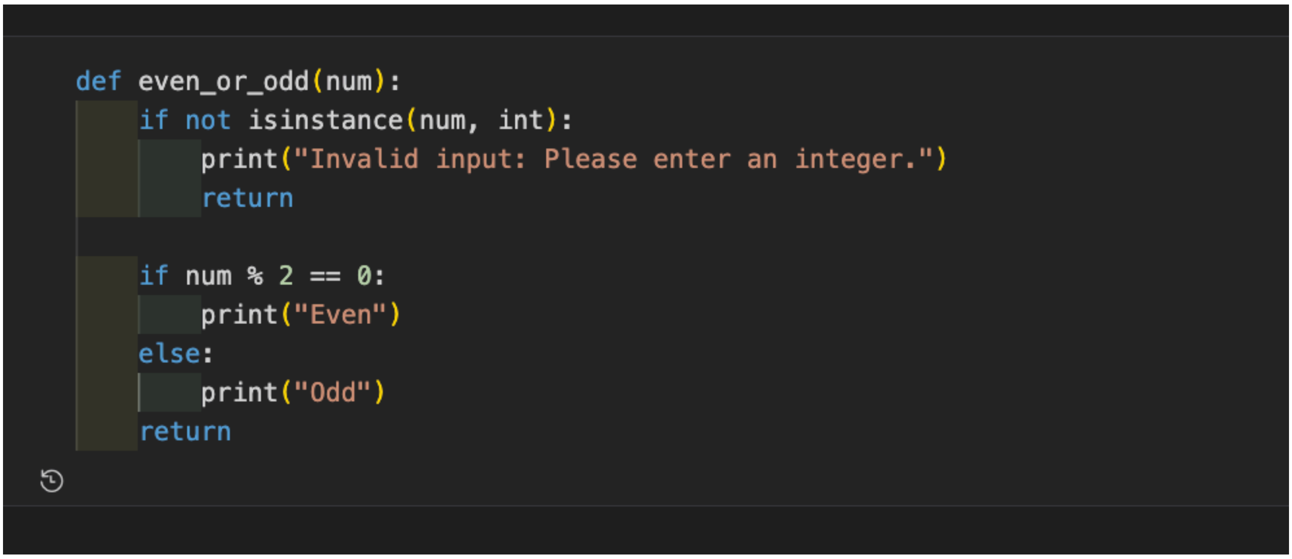
Task-6

Few-Shot Prompting (Even or Odd Classification with Validation)

Prompt:

Write a Python program to determine whether a number is even or odd. Include proper input validation.

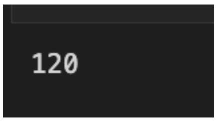
Code:



```python
def even_or_odd(num):
    if not isinstance(num, int):
        print("Invalid input: Please enter an integer.")
        return

    if num % 2 == 0:
        print("Even")
    else:
        print("Odd")
    return
```

Output:



```
120
```