Week 2 Python Programming Tasks: Code and Outputs

July 3, 2025

Introduction

This document contains the Python code and sample outputs for the Week 2 Python Programming tasks. Each task includes the code snippet and a screenshot-like output of running the code with specific inputs, formatted to resemble a terminal.

1 Task 1: Prime Number Check and List

1.1 Code

```
import math

def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

def list_primes_up_to_n(n):
    primes = []
    for i in range(2, n + 1):
        if is_prime(i):
            primes.append(i)
    return primes

def main():
    try:
        n = int(input("Enter a positive integer: "))
        if n < 0:
            print("Please enter a non-negative number.")
            return

        if is_prime(n):
            print(f"{n} is a prime number.")
```

```
        else:
            print(f"{n} is not a prime number.")

        primes = list_primes_up_to_n(n)
        if primes:
            print(f"Prime numbers up to {n}: {primes}")
        else:
            print(f"There are no prime numbers up to {n}.")

    except ValueError:
        print("Please enter a valid integer.")

if __name__ == "__main__":
    main()
```

1.2 Sample Output

```
Enter a positive integer: 10
10 is not a prime number.
Prime numbers up to 10: [2, 3, 5, 7]
```

2 Task 2: First 30 Fibonacci Numbers

2.1 Code

```
def fibonacci_iterative(n):
    if n <= 0:
        return []
    fib = [0, 1] if n > 1 else [0] if n == 1 else []
    for i in range(2, n):
        fib.append(fib[i-1] + fib[i-2])
    return fib

def fibonacci_recursive(n):
    def fib_calc(k):
        if k <= 1:
            return k
        return fib_calc(k-1) + fib_calc(k-2)
    return [fib_calc(i) for i in range(n)] if n > 0 else []

def main():
    n = 30
    try:
        iterative_result = fibonacci_iterative(n)
        print(f"First {n} Fibonacci numbers (Iterative):
            {iterative_result}")
        recursive_result = fibonacci_recursive(n)
        print(f"First {n} Fibonacci numbers (Recursive):
            {recursive_result}")
```

```
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

2.2 Sample Output

```
First 30 Fibonacci numbers (Iterative): [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,
First 30 Fibonacci numbers (Recursive): [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,
```

3 Task 3: GCD and LCM

3.1 Code

```
def gcd(a, b):
    a, b = abs(a), abs(b)
    while b:
        a, b = b, a % b
    return a

def lcm(a, b):
    if a == 0 or b == 0:
        return 0
    return abs(a * b) // gcd(a, b)

def main():
    try:
        a = int(input("Enter the first integer: "))
        b = int(input("Enter the second integer: "))
        gcd_result = gcd(a, b)
        lcm_result = lcm(a, b)
        print(f"GCD of {a} and {b}: {gcd_result}")
        if lcm_result == 0:
            print("LCM is undefined when either number is zero.")
        else:
            print(f"LCM of {a} and {b}: {lcm_result}")
    except ValueError:
        print("Please enter valid integers.")

if __name__ == "__main__":
    main()
```

3.2 Sample Output

```
Enter the first integer: 48
Enter the second integer: 36
```

```
GCD of 48 and 36: 12
LCM of 48 and 36: 144
```

4 Task 4: Prime Factors

4.1 Code

```python
def prime_factors(n):
    factors = []
    if n <= 1:
        return factors
    while n % 2 == 0:
        factors.append(2)
        n = n // 2
    for p in range(3, int(n**0.5) + 1, 2):
        while n % p == 0:
            factors.append(p)
            n = n // p
    if n > 1:
        factors.append(n)
    return factors

def main():
    try:
        n = int(input("Enter a positive integer: "))
        if n <= 0:
            print("Please enter a positive integer.")
            return
        factors = prime_factors(n)
        if factors:
            print(f"Prime factors of {n}: {factors}")
        else:
            print(f"{n} has no prime factors.")
    except ValueError:
        print("Please enter a valid integer.")

if __name__ == "__main__":
    main()
```

4.2 Sample Output

```
Enter a positive integer: 100
Prime factors of 100: [2, 2, 5, 5]
```

5 Task 5: Maximum Sum Subarray (Kadane's Algorithm)

5.1 Code

```python
def kadanes_algorithm(arr):
    if not arr:
        return 0, []
    current_sum = max_sum = arr[0]
    start = temp_start = 0
    end = 0
    for i in range(1, len(arr)):
        if arr[i] > current_sum + arr[i]:
            current_sum = arr[i]
            temp_start = i
        else:
            current_sum += arr[i]
        if current_sum > max_sum:
            max_sum = current_sum
            start = temp_start
            end = i
    return max_sum, arr[start:end+1]

def main():
    try:
        input_str = input("Enter a list of numbers
           (comma-separated): ")
        arr = [int(x) for x in input_str.split(",")]
        if not arr:
            print("The list is empty.")
            return
        max_sum, subarray = kadanes_algorithm(arr)
        print(f"Maximum subarray sum: {max_sum}")
        print(f"Subarray with maximum sum: {subarray}")
    except ValueError:
        print("Please enter a valid list of integers (e.g., 1,
           -2, 3).")

if __name__ == "__main__":
    main()
```

5.2 Sample Output

```
Enter a list of numbers (comma-separated): -2, 1, -3, 4, -1, 2, 1, -5, 4
Maximum subarray sum: 6
Subarray with maximum sum: [4, -1, 2, 1]
```