Sample Coding Questions

Loops

 # Assignment 1: Sum of First N Natural Numbers

 # Problem: Write a program to find the sum of the first

 # N natural numbers, where N is provided by the user.

```
# Problem 1 Solution
n = int(input("Enter a positive integer (N): "))
sum_of_numbers = 0

for i in range(1, n+1):
    sum_of_numbers += i  #sum_of_numbers = sum_of_numbers + i

print(f"The sum of first {n} natural numbers is: {sum_of_numbers}")
```

 # Write a Python program that calculates the sum of

 # even and odd numbers within a specified range.

```
# Get the range from the user
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))

# Initialize variables to hold sums
sum_even = 0
sum_odd = 0
```

```
# Loop through the range and calculate sums
for num in range(start, end + 1):
    if num % 2 == 0:
        #print(num)
        sum_even += num
    else:
        #print(num)
        sum_odd += num

# Print the results
print(f"Sum of even numbers between {start} and {end}: {sum_even}")
print(f"Sum of odd numbers between {start} and {end}: {sum_odd}")

# Write a Python program that
# displays all the even and odd numbers within a specified range.

start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))

# Display even and odd numbers
print(f"Even numbers between {start} and {end}:")
for num in range(start, end + 1):
    if num % 2 == 0:
        print(num, end=" ")
```

```
print(f"\nOdd numbers between {start} and {end}:")
for num in range(start, end + 1):
    if num % 2 != 0:
        print(num, end=" ")

# Assignment 2: Multiplication Table
# Problem: Write a program to print the
# multiplication table of a given number up to a specified range (e.g., 10).

# Problem 2 Solution
number = int(input("Enter a number for the multiplication table: "))
range_limit = 10

print(f"Multiplication table for {number}:")
for i in range(1, range_limit + 1):
    print(f"{number} x {i} = {number * i}")

# Assignment 3: Factorial Calculator
# Problem: Create a program that calculates the factorial of a given number.
# The factorial of a number is the multiplication of
# all the numbers between 1 and the number itself

# Problem 3 Solution
n = int(input("Enter a non-negative integer (N): "))
factorial = 1
```

```
for i in range(1, n + 1):
    factorial *= i

print(f"The factorial of {n} is: {factorial}")
```

 # Assignment 4: Fibonacci Sequence

 # Problem: Write a program that generates the Fibonacci

 # sequence up to a specified limit (e.g., less than 1000).

 # The sequence follows the rule that each number is equal to the sum of the preceding

 # two numbers. The Fibonacci sequence begins with the

 # following 14 integers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233

```
limit = int(input("Enter the number"))
num1 = 0
num2 = 1
num3 = 0
print(num1, end = " ")
print(num2, end = " ")

for i in range(3, limit+1):
    num3 = num1 + num2
    print(num3, end = " ")
    num1 = num2
    num2 = num3
```

 # Assignment 5: Prime Number Generator

 # Problem: Write a program that generates a list of

 # prime numbers between a specified range (e.g., 10 and 50).

 # A prime number is a positive integer that is divisible only by itself and 1.

 # Examples of prime numbers include 2, 3, 5, 7, and 11.

```
start = int(input("Enter starting number"))
end = int(input("Enter ending number"))

print("Prime numbers between", start, "and", end, "are:")

for num in range(start, end + 1):
  # all prime numbers are greater than 1
  #Prime numbers are defined as positive integers greater than 1.
  if num > 1:
     for i in range(2, num):
        if (num % i) == 0:
  #if statement checks if the current number num is divisible by i
  # (i.e., num modulo i is zero).
  # If this condition is true, it means num is not a prime number.
           break
   #If num is found to be divisible by i,
   # the break statement is executed, which exits the inner for loop.
     else:
   #If the inner loop completes without finding any factors of num,
   # the else block is executed. This means that num is a prime number.
```

```
        print(num)
```

The placement of "else" outside the "if" block is intentional. It is associated with the inner for loop and executes only if the loop completes without finding a divisor for num. This is a common construct used in Python to optimize the prime number-checking algorithm.

 # Assignment 6: Palindrome Checker

 # Problem: Write a program that checks if a given word is

 # a palindrome (reads the same backward as forward).

```
string=input(("Enter a letter:"))
if(string==string[::-1]):
#if statement that checks if the entered string string is equal to its
# reverse string[::-1].
# The expression string[::-1] uses slicing to create a reversed version
# of the string.
#the slice statement [::-1] means start at the end of the string and end at position 0,
# move with the step -1 ,
# negative one, which means one step backwards.
    print("The String is a palindrome")
else:
    print("The String is not a palindrome")
```

OR

```
string = input("Enter a word or phrase: ")

# Use the reversed() function to create a reversed version of the string
reversed_string = ''.join(reversed(string))
#This line uses the reversed() function to create a reversed version of the
# inputted string.
# The reversed() function returns an iterator, which needs to be joined into a
# string using ''.join().

if string == reversed_string:
    print(f"'{string}' is a palindrome.")
else:
    print(f"'{string}' is not a palindrome.")
```

 # Assignment 7: Pattern Printing

 # Problem: Create a program that prints a pattern of asterisks as shown below:

 # *

 # **

 # ***

 # ****

```
n = 5
for i in range(1, n+1):
    print('*' * i)
```

 # Problem: Create a program that prints a pattern of asterisks as shown below:

*****

****

***

**

*

```
n=6
for i in range(n,0,-1):
    print('*'* i)
```

 # Assignment 8: Reverse a Number

 # Problem: Write a program that reverses a given number (e.g., 12345 becomes 54321).

```
# Problem 8 Solution
number = int(input("Enter a number: "))
reversed_number = 0

while number > 0:
    digit = number % 10
    # print(digit)
    reversed_number = (reversed_number * 10) + digit
    # print(reversed_number)
    number = number // 10
    # print(number)
```

```
print(f"The reversed number is: {reversed_number}")
```

 # Assignment 9: Armstrong Number Checker

 # Problem: Build a program that checks if a given number is an Armstrong number

 # (a number that is equal to the sum

# of its own digits each raised to the power of the number of digits).

 # For example, 371 is an Armstrong number since 33 + 7**3 + 1**3 = 371.**

```
# Problem 9 Solution
number = int(input("Enter a number: "))
num_digits = len(str(number))
sum_of_cubes = 0
temp = number

while temp > 0:
    digit = temp % 10
    print(digit)
    sum_of_cubes += digit ** num_digits
    print(sum_of_cubes)
    temp //= 10
    print(temp)

if number == sum_of_cubes:
    print(f"{number} is an Armstrong number.")
else:
```

```
    print(f"{number} is not an Armstrong number.")
```

 # Assignment 10: GCD (Greatest Common Divisor)

 # Problem: Write a program that finds the greatest common divisor

 # (GCD) of two given numbers.

 # the greatest common divisor (GCD)

 # is the largest positive integer that divides each of the integers

 # For example, the GCD of 8 and 12 is 4

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

while num2:
    temp = num1
    num1 = num2
    num2 = temp % num2

gcd = num1
print(f"The GCD of the entered numbers is: {gcd}")
```

 # Assignment 11: Pattern Printing (Part 2)

 # Problem: Create a program that prints a pattern of numbers as shown below:

 # 1

 # 12

 # 123

 # 1234

 # 12345

```
# Problem 11 Solution
n = 5

for i in range(1, n+1):
    for j in range(1, i+1):
        print(j, end="")
    print()
```

 # Assignment 12: Square Number Generator

 # Problem: Write a program that generates and prints the square of numbers from 1 to 10.

```
# Problem 12 Solution
for i in range(1, 11):
    square = i**2
    print(f"The square of {i} is: {square}")
```

 # Assignment 13: Display and addition of all prime numbers within a range

 # Python program to calculate the sum of prime numbers within a specified #range and also display all those prime numbers within a specified range

```
start = int(input("Enter starting number"))
end = int(input("Enter ending number"))
sum_of_prime = 0
print("Prime numbers between", start, "and", end, "are:")
```

```
for num in range(start, end + 1):
  if num > 1:
     for i in range(2, num):
        if (num % i) == 0:
           break
     else:
        print(num)
        sum_of_prime+=num



print(f"\nSum of prime numbers between {start} and {end}: {sum_of_prime}")
```

#Assignment 14: Leap Year Checker

#find all the leap years within a specified range

#Leap Year Logic - Divisible by 4 AND (not divisible by 100 OR divisible by 400).

```
# Get the range from the user
start_year = int(input("Enter the start year: "))
end_year = int(input("Enter the end year: "))

# Find and display leap years
leap_years = []
for year in range(start_year, end_year + 1):
   if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
      leap_years.append(year)
```

```
if len(leap_years) > 0:
    print(f"The leap years between {start_year} and {end_year} are:")
    print(leap_years)
else:
    print(f"There are no leap years between {start_year} and {end_year}.")
```