1) Write a Python function find_smallest_multiple(n: int) that uses a while loop to find the smallest positive integer x such that x is divisible by all numbers from 1 to n.
Conditions:

i. The function should keep incrementing x by 1 until it finds a number that satisfies the condition.

ii. Your solution should handle the edge case where n = 1 efficiently, returning 1 directly since 1 is divisible by itself.

```
def find_smallest_multiple(n: int) -> int:
    # Handle edge case efficiently
    if n == 1:
        return 1
    x = n  # Start checking from n, since the smallest multiple must
be at least n
    while True:
        divisible = True
        for i in range(1, n + 1):
            if x % i != 0:
                divisible = False
                break
        if divisible:
            return x
        x += 1  # Increment x until condition is satisfied
print(find_smallest_multiple(1))  # Output: 1
print(find_smallest_multiple(3))  # Output: 6  (since 6 % 1==0, 6 %
2==0, 6 % 3==0)
print(find_smallest_multiple(5))  # Output: 60
```

```
1
6
60
```

2) Write a function cubesum() that accepts an integer and returns the sum of the cubes of individual digits of that number. Use this function to make functions PrintArmstrong() and isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.

```
def cubesum(num: int) -> int:
    total = 0
    n = num
    while n > 0:
        digit = n % 10
        total += digit ** 3
        n //= 10
    return total

def isArmstrong(num: int) -> bool:
```

```
    return num == cubesum(num)

def PrintArmstrong(start: int, end: int) -> None:
    print(f"Armstrong numbers between {start} and {end}:")
    for i in range(start, end + 1):
        if isArmstrong(i):
            print(i, end=" ")
    print()

PrintArmstrong(1, 1000)
print(isArmstrong(153))
print(isArmstrong(370))
print(isArmstrong(123))
```

```
Armstrong numbers between 1 and 1000:
1 153 370 371 407
True
True
False
```

3) Why is operator precedence important? Give an example where neglecting precedence changes the result.

```
# Operator precedence means that some operators are evaluated before
others.
# For example:

# *, / have higher precedence than +, -.

# Expressions inside parentheses () are evaluated first
result = 10 + 5 * 2
print(result)
# How Python evaluates:

# * has higher precedence than +

# So it evaluates 5 * 2 = 10, then 10 + 10 = 20

result = (10 + 5) * 2
print(result)
# Now parentheses force addition first:

# (10 + 5) = 15, then 15 * 2 = 30
```

4) Write a program to input a decimal number and print its equivalent binary, octal, and hexadecimal using operators.

```
def decimal_to_binary(n: int) -> str:
    if n == 0:
        return "0"
```

```
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n //= 2
    return binary


def decimal_to_octal(n: int) -> str:
    if n == 0:
        return "0"
    octal = ""
    while n > 0:
        octal = str(n % 8) + octal
        n //= 8
    return octal


def decimal_to_hexadecimal(n: int) -> str:
    if n == 0:
        return "0"
    hex_chars = "0123456789ABCDEF"
    hexa = ""
    while n > 0:
        remainder = n % 16
        hexa = hex_chars[remainder] + hexa
        n //= 16
    return hexa

num = int(input("Enter a decimal number: "))
print(f"The entered number is: {num}")

print("Binary equivalent:", decimal_to_binary(num))
print("Octal equivalent:", decimal_to_octal(num))
print("Hexadecimal equivalent:", decimal_to_hexadecimal(num))
```

```
The entered number is: 2806
Binary equivalent: 101011110110
Octal equivalent: 5366
Hexadecimal equivalent: AF6
```

5) Write a Python function to create and print a list where the values are the squares of numbers between 1 and 30 (both included).

```
def print_square_list():
    squares = []
    for i in range(1, 31):
        squares.append(i ** 2)
    print(squares)
print_square_list()
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256,
289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900]
```

6) Write a program that takes a sentence as input and counts the frequency of vowels, consonants, digits, and special characters separately

```
def count_characters(sentence: str):
    vowels = "aeiouAEIOU"
    vowel_count = consonant_count = digit_count = special_count = 0

    for ch in sentence:
        if ch.isalpha():
            if ch in vowels:
                vowel_count += 1
            else:
                consonant_count += 1
        elif ch.isdigit():
            digit_count += 1
        elif not ch.isspace():
            special_count += 1

    print("Vowels:", vowel_count)
    print("Consonants:", consonant_count)
    print("Digits:", digit_count)
    print("Special Characters:", special_count)

sentence = input("Enter a sentence: ")
print(f"The entered sentence is {sentence}")
count_characters(sentence)
```

```
The entered sentence is Atul Dhiman Artificial Intelligence
Vowels: 14
Consonants: 18
Digits: 0
Special Characters: 0
```

8) Write a Python program to create a dictionary of students' names as keys and their marks as values. Then:
A. Print the student with the highest marks
B. Print the student with the lowest marks

```


students = {}

n = int(input("Enter the number of students: "))

for i in range(n):
    name = input(f"Enter name of student {i+1}: ")
    marks = float(input(f"Enter marks of {name}: "))
```

```
    students[name] = marks

print("\nStudent Marks Dictionary:")
print(students)

highest_student = max(students, key=students.get)
print(f"\nStudent with highest marks: {highest_student}
({students[highest_student]})")

lowest_student = min(students, key=students.get)
print(f"Student with lowest marks: {lowest_student}
({students[lowest_student]})")
```

```
Student Marks Dictionary:
{'Atul': 84.0, 'Reetik ': 85.0, 'Sahil': 82.0}

Student with highest marks: Reetik  (85.0)
Student with lowest marks: Sahil (82.0)
```