Comprehensive Revision Notes for the Live Class on Software Engineering Concepts

Introduction to Python Functions

Defining a Function

- **Syntax**: Functions are defined using the `def` keyword followed by the function name and parentheses `( )`. Parameters are specified within the parentheses.
- **Example**:

```
def add(a, b):
    return a + b
```

- **Execution**: Once defined, functions can be called using their name followed by arguments enclosed in parentheses.

Example Code for Function Definition and Use

- **Code**:

```
def add(a, b):
    print(a + b)
add(10, 20)  # Outputs: 30
```

  - **Explanation**: This function prints the sum of two numbers .

Difference between Print and Return

- **Print**: Outputs data to the console but does not save it for later use.
- **Return**: Outputs data and allows it to be used later in the program. For instance:

```
def add(a, b):
    return a + b
result = add(10, 20)  # 'result' now holds the value 30, which
```

The `while` Loop

- **Initial Condition**: Defines where the loop starts. Example: `i = 5`.
- **Break Condition**: Determines when the loop should terminate. E.g., `while i != 0`.
- **Movement Condition**: Controls how the loop progresses towards the break condition .

Introduction to Object-Oriented Programming (OOP)

Key Concepts

- **Encapsulation**: Restricts access to certain properties and methods of an object.
- **Inheritance**: Allows new objects to take on properties of existing objects.
- **Polymorphism**: Enables objects to be treated as instances of their parent class .

Example with Members and Premium Members

- **Class Implementation**:

```
class Member:
    def __init__(self, name):
        self.name = name
        self.books_borrowed = 0

    def borrow_book(self):
        self.books_borrowed += 1

class PremiumMember(Member):
    def borrow_book(self):
        self.books_borrowed += 2
```

  - **Explanation**: Premium members are allowed to borrow twice as many books as general members through method overriding .

Practical Session Example: GCD and LCM Calculations

- **Definition**: The largest number that divides two or more numbers without leaving a remainder.
- **Example Calculation**: For numbers 10 and 15, the GCD is 5 .

Least Common Multiple (LCM)

- **Definition**: The smallest positive integer that is divisible by both numbers.
- **Example Calculation**: For numbers 10 and 15, the LCM is 30 .

Relationship

- **Formula**: `GCD(a, b) * LCM(a, b) = a * b` .

Coding Practice: Batch Processing Orders

In the context of processing orders, a function might be used to handle orders in batches:

```
def process_orders(n):
    for i in range(1, n + 1):
        # Process logic in batches
        if i % 3 == 0 or i % 3 == 1 or i % 3 == 2:
            print("Processing order", i)
```

- **Explanation**: Orders are processed in batches of 3, with each batch printed according to the modulus operation .

Conclusion

This session covered the fundamentals of defining functions, the intricacies of loops and control structures, and gave a comprehensive introduction to object-oriented programming. Through practical lessons, concepts like GCD, LCM, and batch processing of orders were explored to give real-world coding applications.

---

programming guidelines and examples, you might want to refer to the course materials and praticed examples.