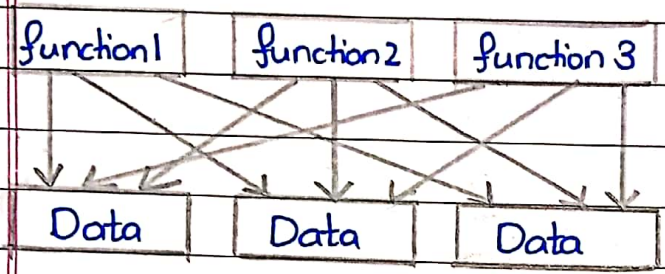
PAGE No.
DATE / /

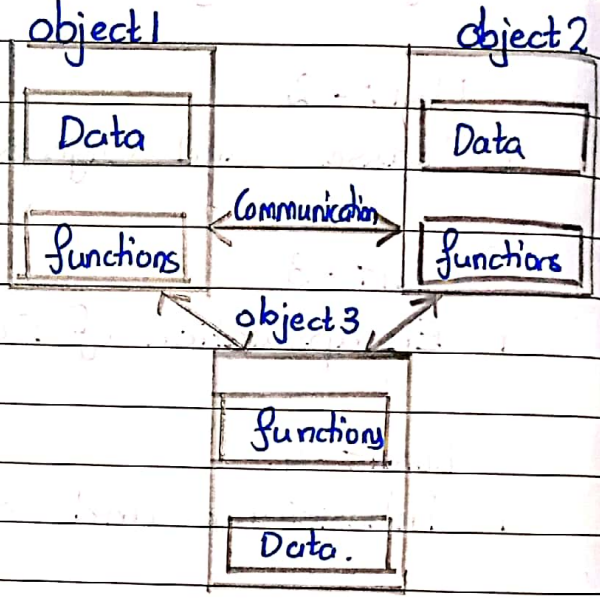
UNIT - 1

INTRODUCTION TO OOP.

POP



OOP.



Features.

1) emphasis is on doing things i.e. importance is given for developing anycode by appling certain algorithm.
2) Large program are divided into smaller programs known as function
3) Most of the functions share global data.
4) Data move openly around the system from function to function.
5) Functions transform data from 1 form to another.
6) ~~Employees~~ Implies or follows top down approach in program design

1) emphasis is on data rather than procedure or algorithm
2) Programs are divided into different objects
3) Data structures are designed such that they characterized the object
4) Functions that operate on the data of an object are tied together in the data structure.
5) Data is hidden and cannot be accessed by external functions.
6) Objects may communicate with each other through functions
7) New data and functions can be easily added whenever necessary
8) Follows bottom up approach in program design

Scanned by CamScanner

PAGE No.
DATE / /

* Definition

1) OOP : OOP is an approach that provides way of modularizing programs by creating partitioned memory area for both data & functions that can be used as templates for creating copies of such modules on demand.
Thus an object is considered to be a partitioned area of computer memory that stores data & set of operation (functions) that can access that data since the memory partition are independant the objects can be used in variety of different program without modification.

Application of OOPS

There are various areas of applications of OOPS

1) Real Time Systems (when data is generated, it is immediately resulted (current status)) Eg: live streaming, videocall, multipurpose gaming system
2) Simulations & Moduling
3) Object Oriented Databases
4) Hypertext & Hypermedia Eg: PHP
5) Artifical Intelligence
6) Neural Networks (Used in data science) & Parallel Programming
7) Desision Support (Java, python, C#, etc.) & Office Automation Systems
8) Communication Media
9) Online Transcation Processing, etc.

Scanned by CamScanner

PAGE No.
DATE / /

* Basic concept of OOP's:

1) Objects
2) Classes
3) Data abstraction & encapsulation
4) Inheritance
5) Polymorphism
6) Dynamic binding
7) Message passing

1) Objects :

Objects are basic runtime entity in an object-oriented system They may represent a person, place, a bank account, etc. Any programing problem can be analysed in terms of objed and the nature of communication between them. When a project is executed the object interacts by sending messages to one another. every info in Object-Oriented Program is stored in memory in terms of object and accessed by the name of object only at runtime That's why objects are called runtime entity

Eg: Class student
Data members
Roll no
name
address
class
div
Functions
add (....)
modify(..)

Scanned by CamScanner

2) Classes:

Objects contain data and functions to manipulate that data. The entire set of data and function/code of an object can be made user defined type with the help of a class. Infact objects are variables of type class.

Once a class has been defined, we can create any no. of object belonging to that class.

Inshort a class is thus a collection of objects of similar type

3) Data abstraction & encapsulation.

Abstraction refers to the act of representing essential features without including bg details. Classes use the concept of abstraction and are defined as list of abstract attributes and functions or methods to operate on these attributes. They encapsulate all the essential property of the objects that are to be created. Attribute are also called as data members and functions are called as methods

4) Inheritance:

Inheritance is the process by which objects of 1 class acquire the properties of objects of another class. It supports the concept of ~~hierarc~~ hierarchial of maintaining the classification. The principle behind this sort of division is that each divide class share common characteristics with the class from which it is derived.

In OOP, the concept of inheritance provides the idea of reuseabilities ie we can add additional features, attributes, etc to an existing class

Scanned by CamScanner

without modifying it. This is possible by deriving new class from the existing one. The new class will have combined features of both the classes.
Every subclass defines only those features that are unique to it
The class from which these features are derived is called as parent class or base class and the class which derived this properties is called "derived class" or "subclass" or "child class."

5) Polymorphism:
It means the ability to take more than 1 form An operation may exhibit different behaviours in different instances. The behaviour depends upon the type of data used in the operation. For eg: In the operation of addition of two nos it will generate sum and if the operands are strings then the operation would produce 3rd String by concatenation. The process of making an operator to exhibit different behaviour in different instances in known as operator overloading.
A singe function name can be used to handle different nos and different types of arguments. It is called as function (method) overloading
A call to a specific function is decided by the compiler at compiled time depending upon the no & types of values passed so it is also called as

Scanned by CamScanner

compiled type polymorphism:
Some times there can be multiple functions with same name and no and type of arguments are present in different classes related with each other then which function to be called for execution is decided at runtime and it is implemented by using the concept of method overriding also called as runtime polymorphism.

6) Dynamic binding:
Binding refers to the linking of procedure call to the code to be executed in response to the call. Dynamic binding also known as "late binding" means that the code associated with the given procedure call is not known until the time of the call at runtime. It is associated with polymorphism and inheritance.

7) Message Passing:
An OOP consist of set of objects that communicate with each other. The process of programming in object-oriented language involves the foll. basic steps:
1) creating classes that define objects & their behaviour
2) creating objects from class definitions
3) destablishing communication among objects.
Objects communicate with one another by sending & receiving info with each other. A message for an object is a request for execution of a procedure and therefore invoke a method in the receiving object that generates the desired result. Message passing

Scanned by CamScanner

PAGE No.
DATE / /

involves specifying name of object, name of method (message) and info to be sent.
Objects have lifesycle, they can be created & destroyed.

Scanned by CamScanner

30/11/19.

PAGE No.
DATE / /

Objects Everywhere

* Recognizing Objects from nouns.

Q Develope an application to calculate areas & perimeters of square, rectangle, circle and ellipse.

⇒ The 4 nouns in the requirement represent real life objects square, rectangle, circle and ellipse

- We can design our application by following object-oriented paradigm instead of creating set of functions that perform a required task we can create software objects that represent the state & behaviours of different shapes.
- The required data for each shape can be represented as follows
  It can be encapsulated in each object

Shape	Required data.
Square	Length of side
Rectangle	width and height
Circle	Radius (usually labelled as r)
Ellipse	Semi-major axis and Semi-minor axis

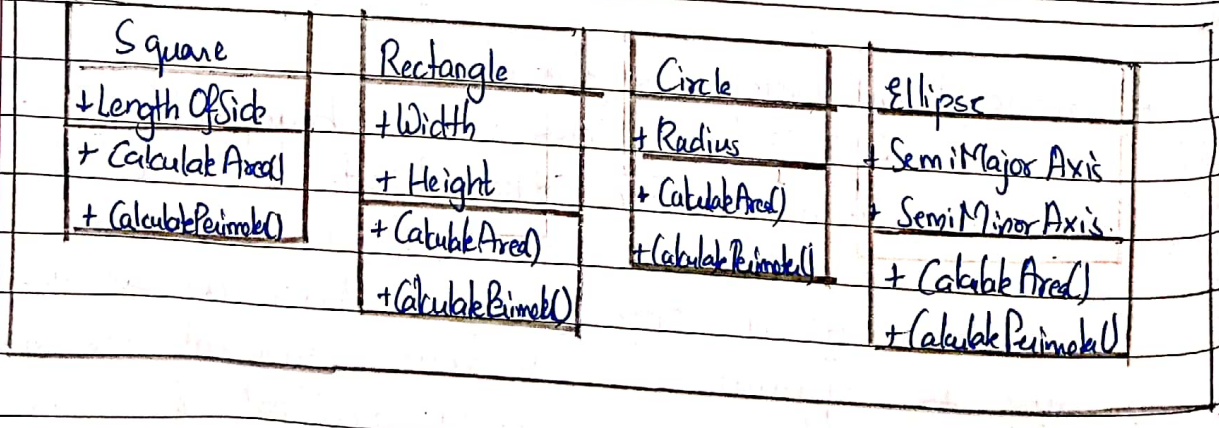
* Generating blueprints for objects:

In OOP a class is a blueprint or template definition from which the objects are created. Classes are models that define the state and behaviour of an object. After that, we can use this class to generate objects that represent the state and behavior of each real-wold rectangle & other shapes

Scanned by CamScanner

If we want to create 4 instances with their widths & heights specified: [Rect1, Rect2, Rect3, Rect4]
We can use rectangle class as blueprint to generate 4 different rectangle instances

* Recognizing attributes / fields
It is necessary to include attributes that provide the required data to each instance i.e. we need an encapsulated variable that allowed each instance of this class to specify the value of length of side. This encapsulated data for each instance of the class are known as attributes.
Each Instance has its own independent value for the attributes defined in the class
- The above table showing shape and required data can be represented as Class and attribute List.
- This info now can be represented using UML (Unified Modelling Language) diagram with the 4 classes and their attributes.



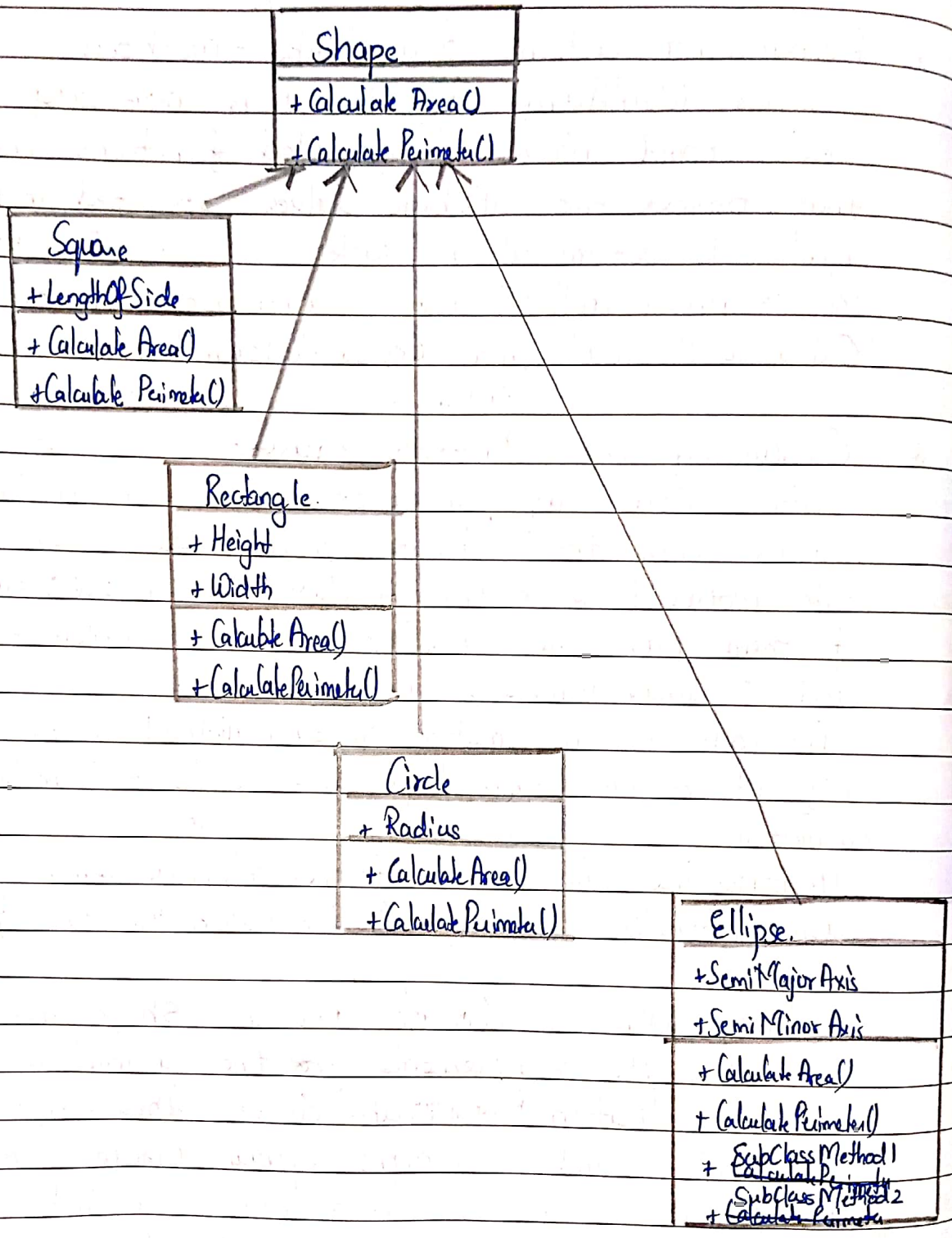
Scanned by CamScanner

PAGE No.
DATE / /

* Recognizing actions from verbs - methods
After identifying attributes it is necessary that each class should have necessary encapsulated functions / methods that process the attribute values specified in the objects to perform all the task
The required method for calculating area can be "Calculate Area()" and "Calcalate Perimeter()"

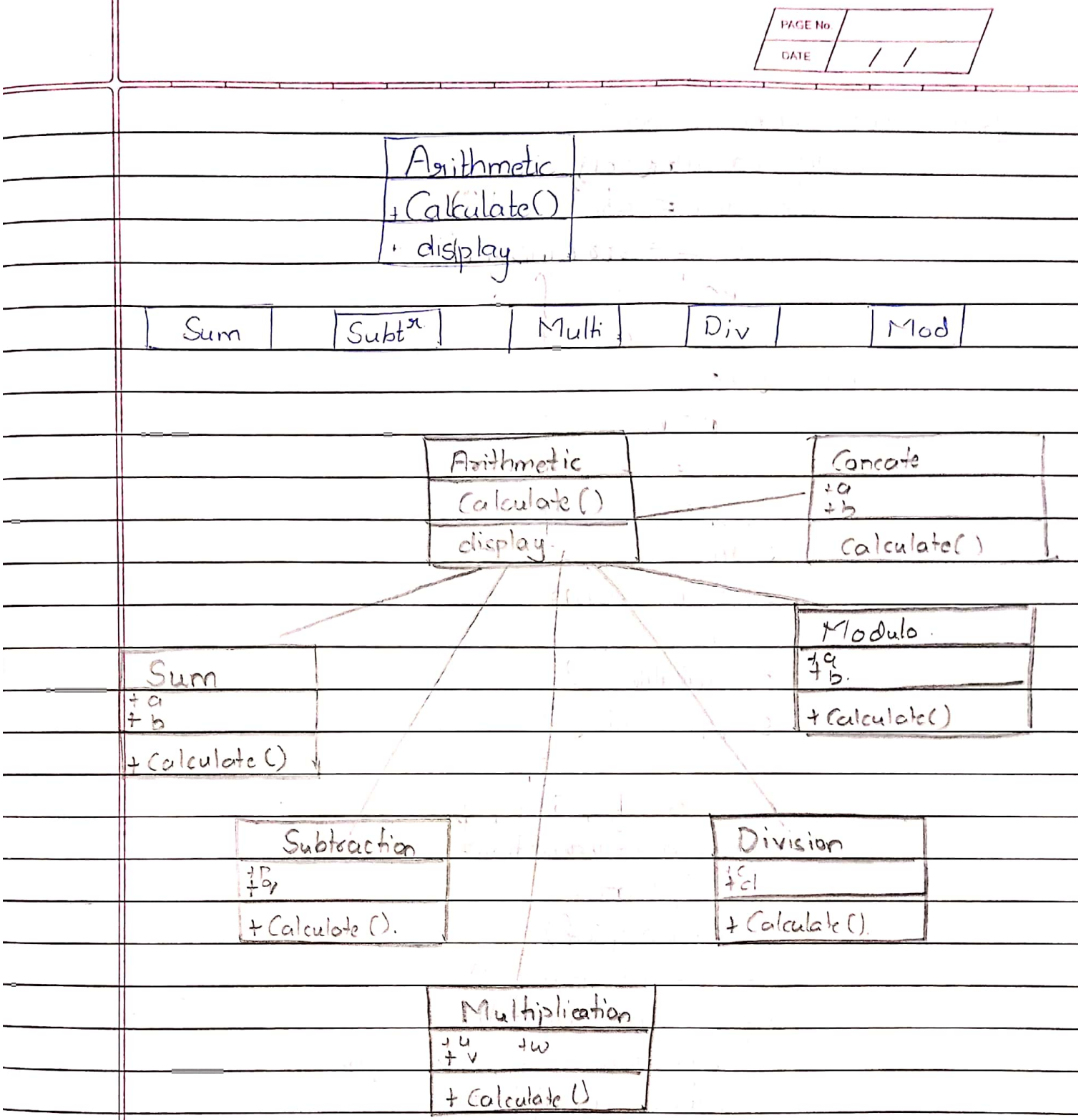
* Organizing the blueprints - classes
Uptill now our Object-oriented solution includes 4 classes with their attributes & methods however if we take another look at this 4 classes we notice that all of them have the same 2 methods "CalculateArea()" and "Calaulate Perimeter()"
The code for this methods in each method in each class is different because each shape uses different formula for calculation.
However the declaration of methods are same so we generalize the required behavior for the four shapes.
We can define a common class as Shape and it will generalize the requirements for the geometrical shapes in our application. We can declare this shape as parent class and rectangle, square, circle & ellipse as sub-classes of the shape class to inherit these two methods but this derived classes can have specific code for each of this methods.

Scanned by CamScanner



The updated UML class diagram shows the relation between shape as parent class and 4 different child classes. It also indicates that all 4 classes inherits the behavior (methods) from base class but they can have their own behavior or own methods as well as attributes.

Scanned by CamScanner



Scanned by CamScanner

Q.1)

```
class concatenation:
    def calculate(self):
        self.var1 = str(input())
        self.var2 = str(input())
        concatenation = self.var1 + self.var2
        return(concatenation)
class sum:
    def calculate(self)
        self.x = int(input())
        self.y = int(input())
        sum = self.x + self.y
        return(sum)
class subtraction:
    def calculate(self)
        self.p = int(input())
        self.q = int(input())
        subtraction=self.p - self.q
        return(subtraction)
class multiplication:
    def calculate(self)
        self.u = int(input())
        self.v = int(input())
        multiplication = self.u * self.v
        return multiplication
class division:
    def calculate(self)
        self.c = int(input())
        self.d = int(input())
        division = self.c / self.d
        return division
```

Scanned by CamScanner

```
class modulo:
    def calculate(self):
    self.a = int(input())
    self.b = int(input())
     modulo = self.a % self.b
        return modulo.
```

Q.2)

```
class Arithmetic
    def calculate(self):
    self.a = int(input())
    self.b = int(input())
    sum = self.a + self.b
     sub = self.a - self.b
     mul = self.a + self.b
     div = self.a/self.b
     print(sum)
     print(sub)
     print(mul)
     print(div)
a = Airthmetic()
    a.calculate().
```

Scanned by CamScanner