Indira College Of Engineering and Management, Pune

Department of MCA

Subject Name

Python Programming

Guided by

Prof. Hetal Thaker

Submitted by

Yash Chandrashekhar Khadke
(MC116)

Sr. No.	Practical No.	Date	Sign.
1	Practical No. 1	08-02-2025	
2	Practical No. 2	10-02-2025	
3	Practical No. 3	03-03-2025	
4	Practical No. 4	05-03-2025	
5	Practical No. 5	10-03-2025	
6	Practical No. 6	12-03-2025	
7	Practical No. 7	17-03-2025	
8	Practical No. 8	19-03-2025	

MODULE 1

# 1. Write a Python program that calculates the area of a circle based on the radius entered by the user.

```
import math
radius = float(input("Enter the radius of the circle: "))
 user area = math.pi * (radius ** 2)
print(f"r = {radius} Area = {area}")
```

OUTPUT

```
Enter the radius of the circle: 5
r = 5.0 Area = 78.53981633974483
```

# 2. Write a Python program that accepts the user's first and last name and prints them in reverse order with a space between them.

```
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
print(last_name + " " + first_name)
```

OUTPUT

```
Enter your first name: Yash
Enter your last name:Khad
Yash Khad
```

# 3. Write a Python program that accepts a sequence of comma-separated numbers from the user and generates a list and a tuple of those numbers.

```
numbers = input("Enter comma-separated numbers: ")
num_list = numbers.split(",")
num_tuple = tuple(num_list)
print("List:", num_list)
print("Tuple:", num_tuple)
```

OUTPUT

```
Enter comma-separated numbers: 1,2,3,4
List: ['1', '2', '3', '4']
Tuple: ('1', '2', '3', '4')
```

# 4. Write a Python program that determines whether a given number (accepted from the user) is even or odd, and prints an appropriate message to the user.

```
num = int(input("Enter a number: "))
input if num % 2 == 0:
   print(f"{num} is an Even
number") else:
   print(f"{num} is an Odd number")
```

OUTPUT

```
Enter a number: 7  7 is an Odd number
Enter a number: 8
8 is an Even number
```

# 5. Write a Python program to concatenate N strings.

```
n = int(input("Enter the number of strings you want to concatenate: "))
concatenated_string = ""
for _ in range(n):
    user_string = input("Enter a string: ")
    concatenated_string += user_string
print("Concatenated String:", concatenated_string)
```

OUTPUT

```
Enter the number of strings you want to concatenate: 3
Enter a string: Hello
Enter a string:
Enter a string: World!
Concatenated String: Hello
World!
```

MODULE 2

# 1. Write a Python program to do arithmetical operations addition and division.

```
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
addition = a + B
division = a / b if b != 0 else "Undefined (division by zero)”
print("Addition:", addition)
print("Division:", division)
```

OUTPUT

```
Enter first number: 10
Enter second number:
2 Addition: 12.0
Division: 5.0
```

# 2. Write a Python program to find the area of a triangle.

```
base = float(input("Enter base of the triangle: "))

height = float(input("Enter height of the triangle:

"))

area = 0.5 * base * height  # Formula for area of triangle: (1/2) * base * height
print("Area of the triangle:", area)
```

OUTPUT

```
Enter first variable: Hello
Enter second variable:
World
After swapping: x = World y = Hello
```

# 3. Write a Python program to swap two variables.

```
x = input("Enter first variable: ")
y = input("Enter second variable:
") x, y = y, x
print("After swapping: x =", x, "y =", y)
```

OUTPUT

```
Enter base of the triangle:
5 Enter height of the
triangle: 4 Area of the
triangle: 10.0
```

# 4. Write a Python program to generate a random number.

```
import random
rand_num = random.randint(1,
100)  print("Random  number:",
rand_num)
```

OUTPUT

```
Random number: 57
```

# 5. Write a Python program to convert kilometers to miles.

```
km = float(input("Enter distance in kilometers: "))
miles = km * 0.621371
print(f"{km} kilometers is equal to {miles} miles")
```

OUTPUT

```
Enter distance in kilometers: 10
10.0 kilometers is equal to 6.21371 miles
```

# 6. Write a Python program to display calendar.

```
import calendar
year = int(input("Enter year: "))
month = int(input("Enter month (1-12):
")) print(calendar.month(year, month))
```

OUTPUT

```
Enter year: 2023
Enter month (1-12): 3
   March 2023
Mo Tu We Th Fr Sa
      Su 1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

# 7. Write a Python program to swap two variables without temp variable.

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
a = a + b
b = a -
b a =
a - b
print("After swapping: a =", a, "b =", b)
```

OUTPUT

```
Enter first number: 15
Enter second number:
25
After swapping: a = 25 b = 15
```

# 8. Write a Python program to check if a number is positive, negative, or zero.

```
num = float(input("Enter a number:
")) if num > 0:
   print("The number is
Positive") elif num < 0:
   print("The number is
Negative") else:
   print("The number is Zero")
```

OUTPUT

```
        Enter a number: -7
        The number is Negative
```

# 9. Write a Python program to check if a year is a leap year.

```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(year, "is a Leap Year")
else:
    print(year, "is not a Leap Year")
```

OUTPUT

```
        Enter a year: 2020

        2020 is a Leap Year
```

# 10. Write a Python program to check if a number is odd or even.

```
num = int(input("Enter a number:
")) if num % 2 == 0:
    print(f"{num} is
Even") else:
    print(f"{num} is Odd")
```

OUTPUT

```
        Enter a number:

        4 4 is Even
```

MODULE 3

# 1. Create a dictionary of your favourite books and their authors and print it.

```
books = {
    "1984": "George Orwell",
    "To Kill a Mockingbird": "Harper Lee",
    "The Great Gatsby": "F. Scott
    Fitzgerald", "Pride and Prejudice":
    "Jane Austen"
}
print("Favorite Books and Authors:", books)
```

OUTPUT

```
Favorite Books and Authors: {'1984': 'George Orwell', 'To Kill a Mockingbird': 'Harper Lee',
'The Great Gatsby': 'F. Scott Fitzgerald', 'Pride and Prejudice': 'Jane Austen'}
```

# 2. Add a new book to the dictionary and print the updated dictionary.

```
books["The Catcher in the Rye"] = "J.D. Salinger"  # Adding a new book
print("Updated Books Dictionary:", books)
```

OUTPUT

```
Updated Books Dictionary: {'1984': 'George Orwell', 'To Kill a
Mockingbird': 'Harper Lee', 'The Great Gatsby': 'F. Scott Fitzgerald',
'Pride and Prejudice': 'Jane Austen', 'The Catcher in the Rye': 'J.D.
Salinger'}
```

# 3. Remove a book from the dictionary and print the updated dictionary.

```
books.pop("1984")  # Removing a book by title
print("Dictionary after removing a book:", books)
```

OUTPUT

```
Dictionary after removing a book: {'To Kill a
Mockingbird': 'Harper Lee', 'The Great Gatsby': 'F.
Scott Fitzgerald', 'Pride and Prejudice': 'Jane
Austen', 'The Catcher in the Rye': 'J.D. Salinger'}
```

# 4. Use the keys() method to print a list of the book titles in the dictionary.

```
print("Book Titles:", list(books.keys())) # Extracting only book titles
```

OUTPUT

```
Book Titles: ['To Kill a Mockingbird', 'The Great Gatsby', 'Pride and Prejudice', 'The Catcher
in the Rye']
```

# 5. Use the values() method to print a list of the author names in the dictionary.

```
print("Author Names:", list(books.values()))  # Extracting only author names
```

OUTPUT

```
Author Names: ['Harper Lee', 'F. Scott Fitzgerald', 'Jane Austen', 'J.D. Salinger']
```

# 6. Create a set of your favourite colours and print it.

```
colors = {"Red", "Blue", "Green", "Black", "White"}
```

print("Favorite Colors Set:", colors)

OUTPUT

Favorite Colors Set: {'Blue', 'Green', 'White', 'Black', 'Red'}

# 7. Add a new colour to the set and print the updated set.

colors.add("Yellow") # Adding a new
color print("Updated Colors Set:",
colors) **OUTPUT**

Updated Colors Set: {'Blue', 'Green', 'White', 'Black', 'Red', 'Yellow'}

# 8. Remove a colour from the set and print the updated set.

colors.discard("White") # Removing a color
print("Set after removing a color:", colors)

OUTPUT

Set after removing a color: {'Blue', 'Green', 'Black', 'Red', 'Yellow'}

# 9. Create a new set that contains only the colours that start with the letter "B" and print it.

colors_starting_with_B = {color for color in colors if color.startswith("B")}
print("Colors starting with 'B':", colors_starting_with_B)

OUTPUT

Colors starting with 'B': {'Blue', 'Black'}

# 10. Use the len() function to find the number of colours in the set and print it.

print("Number of colors in the set:", len(colors))

OUTPUT

Number of colors in the set: 5

MODULE 4

# 1. Program to Find the GCD of Two Positive Numbers.

```
num1 = int(input("Enter first positive number: "))
num2 = int(input("Enter second positive number:
")) while num2:
   num1, num2 = num2, num1 %
num2 print(f"GCD of the numbers is
{num1}") OUTPUT
```

```
Enter first positive number: 48
Enter second positive number:
18 GCD of the numbers is 6
```

# 2. Write Python Program to Find the Sum of Digits in a Number.

```
num = int(input("Enter a number:
")) sum_of_digits = 0
while num:
   sum_of_digits += num % 10
   num //= 10
print(f"Sum of digits is {sum_of_digits}")
```

OUTPUT

```
Enter a number: 1234
Sum of digits is 10
```

# 3. Write a program that prints the first 10 multiples of 3.

```
multiples_of_3 = [3 * i for i in range(1, 11)]
print("First 10 multiples of 3:", multiples_of_3)
```

OUTPUT

```
First 10 multiples of 3: [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
```

LIST

# 1. Create a list of your favourite Hindi comedy movies and print the third movie in the list.

```
movies = ["Hera Pheri", "Andaz Apna Apna", "Dhamaal", "Chup Chup Ke", "Golmaal: Fun
Unlimited"] print("Third movie in the list:", movies[2])
```

OUTPUT

```
Third movie in the list: Dhamaal
```

# 2. Add a new movie to the list and print the updated list.

```
movies.append("Bhool Bhulaiyaa")
print("Updated Movie List:",
movies) OUTPUT
```

```
Updated Movie List: ['Hera Pheri', 'Andaz Apna Apna', 'Dhamaal', 'Chup Chup Ke', 'Golmaal:
Fun Unlimited', 'Bhool Bhulaiyaa']
```

# 3. Remove the second movie from the list and print the updated list.

```
movies.pop(1)
print("List after removing the second movie:", movies)
```

OUTPUT

```
List after removing the second movie: ['Hera Pheri', 'Dhamaal', 'Chup Chup Ke', 'Golmaal:
```

```
Fun Unlimited', 'Bhool Bhulaiyaa']
```

# 4. Sort the list in alphabetical order and print the sorted list.

```python
movies.sort()
print("Sorted Movie List:", movies)
```

OUTPUT

```
Sorted Movie List: ['Bhool Bhulaiyaa', 'Chup Chup Ke', 'Dhamaal', 'Golmaal: Fun Unlimited',
'Hera Pheri']
```

# 5. Create a new list that contains only the first and last movie in the original list and print it.

```python
first_last_movies = [movies[0], movies[-1]]
print("First and Last Movie:", first_last_movies)
```

OUTPUT

```
First and Last Movie: ['Hera Pheri', 'Bhool Bhulaiyaa']
```

TUPLE

# 1. Create a tuple of your favourite foods and print the second food in the tuple.

```python
foods = ("Pizza", "Biryani", "Pani Puri", "Chole Bhature", "Dosa")
print("Second food in the tuple:", foods[1])
```

OUTPUT

```
Second food in the tuple: Biryani
```

# 2. Try to change the second food in the tuple and see what happens.

```python
try:
   foods[1] = "Pasta"  # Tuples are immutable, so this will raise an error
except TypeError as e:
   print("Error:", e)
```

OUTPUT

```
Error: 'tuple' object does not support item assignment
```

# 3. Create a new tuple that contains only the first and last foods in the original tuple and print it.

```python
first_last_foods = (foods[0], foods[-1])
print("First and Last Food:", first_last_foods)
```

OUTPUT

```
First and Last Food: ('Pizza', 'Dosa')
```

# 4. Use the len() function to find the number of foods in the tuple and print it.

```python
print("Number of foods in the tuple:", len(foods))
```

OUTPUT

```
Number of foods in the tuple: 5
```

# 5. Convert the tuple to a list and print the list.

```python
foods_list = list(foods)
print("Tuple converted to list:", foods_list)
```

OUTPUT

```
Tuple converted to list: ['Pizza', 'Biryani', 'Pani Puri', 'Chole Bhature', 'Dosa']
```

MODULE 5

# 1. Print all even numbers from 0 to the given number

```
num = int(input("Enter a number: "))
i = 0
while i <= num:
  if i % 2 == 0:
      print(i)
   i += 1
```

OUTPUT

```
Enter
a
numb
er: 10
0
2
4
6
8
10
```

# 2. Print each character of a string on a new line

```
text = input("Enter a string: ")
for char in text:
    print(char)
```

OUTPUT

```
Enter a string:
Hello
H
e
l
l
o
```

# 3. Print pattern

```
a) for  i in range(1, 6):
    for j in range(1, i + 1):
        print(j,
    end="") print()
```

```
b) ch = 65
for i in range(1,
    5): for j in
    range(i):
        print(chr(ch), end="
        ") ch += 1
    print()
```

c) for i in range(4, 0, -1):
```
    for j in range(i):
        print(i,
    end="") print()
```

d) for i in range(1, 6, 2):
```
    for j in range(5 - i, 0, -1):
        print(" ",
    end="") for j in
    range(i):
        print("*", end="
    ") print()
```

e) for i in range(1, 6, 2):
```
    for j in range(5 - i, 0, -1):
        print(" ",
    end="") for j in
    range(i):
        print("*", end="")
    print()
```

f) for i in range(1,
```
    5): for j in
    range(i):
        print((j + i) % 2, end="")
    print()
```

Output
```
1
12
123
1234
12345


A
B C
C D E
D E F G


4444
333
22
1
```

```
    *
  * * *
* * * * *


  *
 ***
*****

1
01
101
0101
```

# 6. Find ASCII value of a character

```
char = input("Enter a character: ")
print("ASCII value of", char, "is",
ord(char))
```

OUTPUT

```
Enter a character: A
ASCII value of A is 65
```

# 7. Simple calculator

```
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
op = input("Enter operation (+, -, *, /): ")
if op == "+":
   print("Result:", a +
b) elif op == "-":
   print("Result:", a -
b) elif op == "*":
   print("Result:", a *
b) elif op == "/":
   if b != 0:
      print("Result:", a /
   b) else:
      print("Division by zero
error") else:
   print("Invalid operation")
```

OUTPUT

```
Enter first number: 10
Enter second number: 5
Enter operation (+, -, *, /): *
Result: 50.0
```

# 8. Find the largest element in an array

```
n = int(input("Enter number of elements: "))
arr = []
```

```
for i in range(n):

    arr.append(int(input())

    )

largest=arr[0]
for num in arr:
    if num > largest:
        largest = num
print("Largest element:", largest)
```
OUTPUT

```
  Enter number of elements:
  5 1
  3
  7
  2
  5
  Largest element: 7
```

# 9. Add two matrices

```
r = int(input("Enter number of rows: "))
c = int(input("Enter number of columns: "))
mat1 = []
mat2 = []
result = []
for i in range(r):
    row = []
    for j in range(c):
        row.append(int(input()))
    mat1.append(row
) for i in range(r):
    row = []
    for j in range(c):
        row.append(int(input()))
    mat2.append(row
) for i in range(r):
    row = []

    for j in range(c):

        row.append(mat1[i][j] + mat2[i][j])
    result.append(row
) for row in result:
    for num in row:
        print(num, end=" ")
    print()
```
OUTPUT

```
        Enter number of rows: 2
```

```
Enter number of columns: 2
Enter elements of first matrix:
1
2
3
4
Enter elements of second matrix:
5
6
7
8
Resultant Matrix:
6 8
10 12
```

MODULE 6

# 1. Program to find the sum of all odd and even numbers up to a number specified by the user.

```
def sum_odd_even(n):
    even_sum = sum(i for i in range(0, n+1, 2))  # Sum of even numbers
    odd_sum = sum(i for i in range(1, n+1, 2))  # Sum of odd numbers
    return even_sum, odd_sum

num = int(input("Enter a number: "))  # Taking input from the user
even_sum, odd_sum = sum_odd_even(num)  # Calling the function

print(f"Sum of even numbers up to {num} is: {even_sum}")
print(f"Sum of odd numbers up to {num} is: {odd_sum}")
```

OUTPUT

```
Enter a number: 10
Sum of even numbers up to 10 is: 30
Sum of odd numbers up to 10 is: 25
```

# 2. Program to check if a given string is a palindrome using slicing.

```
def is_palindrome(s):
    s = s.lower().replace(" ", "")  # Converting to lowercase and removing spaces
    return s == s[::-1]  # Checking if the string is equal to its reverse

user_string = input("Enter a string: ")  # Taking input from the user

if is_palindrome(user_string):  # Checking if palindrome
    print(f"'{user_string}' is a palindrome.")
else:
    print(f"'{user_string}' is not a palindrome.")
```

OUTPUT

```
Enter a string: Race car
'Race car' is a palindrome.
```

# 3. Count Vowels, Consonants, and Blanks in a String

```
def count_chars(s):
    vowels = "aeiouAEIOU"
    consonants = sum(1 for c in s if c.isalpha() and c not in vowels)
    vowel_count = sum(1 for c in s if c in vowels)
    blanks = s.count(' ')
    return vowel_count, consonants, blanks
```

OUTPUT

```
Enter a sentence: Hello World
Vowels, consonants, blanks: (3, 7, 1)
```

# 4. Print Characters Common in Two Strings

```
def common_chars(s1, s2):
    return set(s1) & set(s2)
```

OUTPUT

```
Enter first string: hello
```

```
        Enter second string: world
        Common characters: {'l', 'o'}
```

# 5. Calculate Percentage of Marks

```
def calculate_percentage(marks):
    total_marks = sum(marks)
    percentage = (total_marks / (len(marks) * 100)) * 100
    return percentage
```

OUTPUT

```
        Enter marks separated by space: 80 90 100
        Percentage: 90.0
```

# 6. Display Fibonacci Sequence up to n Terms

```
def fibonacci(n):
    a, b = 0, 1
    fib_seq = []
    for _ in range(n):
        fib_seq.append(a)
        a, b = b, a + b
    return fib_seq
```

OUTPUT

```
        Enter the number of Fibonacci terms: 7
        Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8]
```

# 7. Remove Duplicate Words from a Sentence and Sort Them

```
def remove_duplicates_sort(sentence):
    words = list(set(sentence.split()))
    return " ".join(sorted(words))
```

OUTPUT

```
        Enter a sentence: the quick brown fox jumps over the lazy dog
        Sorted unique words: brown dog fox jumps lazy over quick the
```

# 8. Implement Stack Operations using *args

```
def stack_operations(*args):
    stack = []
    for op in args:
        if op.startswith("push"):
            val = op.split()
            stack.append(val)
        elif op == "pop":
            if stack:
                stack.pop()
        elif op == "peek":
            if stack:
                return stack[-1]
    return stack
```

OUTPUT

```
        operations = ["push 10", "push 20", "pop", "peek"]
        Stack after operations: 10
```

MODULE 7

#1. OOP Concepts in Pharmaceuticals. Using Inheritance, Encapsulation, Abstraction, and Polymorphism

```
from abc import ABC, abstractmethod
class DrugFormulation(ABC):
    def __init__(self, name, dosage, manufacturer):
        self._name = name
        self._dosage = dosage
        self._manufacturer = manufacturer

    @abstractmethod
    def administer(self):
        pass

    def get_info(self):
        return f"{self._name} ({self._dosage}) by {self._manufacturer}"

# Tablet class inherits from DrugFormulation
class Tablet(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} tablet orally with water."
class Capsule(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} capsule with warm water."

class Injection(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} injection intravenously."

def prescribe_drug(drug: DrugFormulation):
    return drug.administer()

tablet = Tablet("Paracetamol", "500mg", "PharmaCorp")
capsule = Capsule("Amoxicillin", "250mg", "MediHealth")
injection = Injection("Insulin", "10ml", "BioCare")

print(prescribe_drug(tablet))
print(prescribe_drug(capsule))
print(prescribe_drug(injection))
```

OUTPUT

```
Administer Paracetamol tablet orally with water.
Administer Amoxicillin capsule with warm water.
Administer Insulin injection intravenously.
```

#2. Program to find the sum of an array

```
def sum_of_array(arr):
    return sum(arr)
```

OUTPUT

```
arr = [1, 2, 3, 4, 5]
Sum of array: 15
```

#3. Program to find the largest element in an array

```
def largest_element(arr):
  return max(arr)
```

OUTPUT

```
arr = [1, 2, 3, 4, 5]
Largest element in array: 5
```

#4. Program to split the array and add the first part to the end

```
def split_and_add(arr, split_index):
  return arr[split_index:] + arr[:split_index]
```

OUTPUT

```
arr = [1, 2, 3, 4, 5]
split_index = 2
Array after splitting and adding: [3, 4, 5, 1, 2]
```

#5. Program to add two matrices

```
def add_matrices(matrix1, matrix2):
  return [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]
```

OUTPUT

```
matrix1 = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
]
matrix2 = [
  [9, 8, 7],
  [6, 5, 4],
  [3, 2, 1]
]
                              Sum of matrices: [[10, 10, 10], [10, 10, 10], [10, 10, 10]]
```

MODULE 8

# 1. Write a program to create point class with x,y,z coordinate and methods increment point, decrementpoint, add points , less than , greater than , equal to , check in which quadrant it lies,check whether the point is collinear and print point.

```
class Point:
  def __init__(self, x, y, z):
    self.x = x
    self.y = y
    self.z = z

  # Increment each coordinate by 1
  def increment(self):
    self.x += 1
    self.y += 1
    self.z += 1

  def decrement(self):
    self.x -= 1
    self.y -= 1
    self.z -= 1

  def add(self, other):
    return Point(self.x + other.x, self.y + other.y, self.z + other.z)

  def __lt__(self, other):
    return (self.x, self.y, self.z) < (other.x, other.y, other.z)

  def __gt__(self, other):
    return (self.x, self.y, self.z) > (other.x, other.y, other.z)

  def __eq__(self, other):
    return (self.x, self.y, self.z) == (other.x, other.y, other.z)

  def quadrant(self):
    if self.x > 0 and self.y > 0:
      return "First Quadrant"
    elif self.x < 0 and self.y > 0:
      return "Second Quadrant"
     elif self.x < 0 and self.y < 0:
       return "Third Quadrant"
     elif self.x > 0 and self.y < 0:
      return "Fourth Quadrant"
    else:
      return "On Axis"
    def __str__(self):
        return f"Point({self.x}, {self.y}, {self.z})"

p1 = Point(1, 2, 3)
p2 = Point(-1, -2, 0)

p1.increment()
p2.decrement()

p3 = p1.add(p2)

print(str(p1))
print(str(p2))
print(str(p3))
print(p1 < p2)
print(p1.quadrant())
print(p2.quadrant())
```

OUTPUT

```
    Point(2, 3, 4)
    Point(-2, -3, -1)
    Point(0, 0, 3)
    False
    First Quadrant
    Third Quadrant
```

#2. Create class watch with hr,min,sec,alarm,type and methods setalarm, stopalarm,showtime.

```
class Watch:
    def __init__(self, hr, min, sec, alarm=None, type="Digital"):
        self.hr = hr
        self.min = min
        self.sec = sec
        self.alarm = alarm
        self.type = type

    def set_alarm(self, alarm_time):
        self.alarm = alarm_time

     def stop_alarm(self):
         self.alarm = None

    def show_time(self):
        return f"{self.hr:02}:{self.min:02}:{self.sec:02}"
```

```
watch = Watch(9, 30, 45)
print(watch.show_time()) # Displays time in HH:MM:SS format

watch.set_alarm("10:00:00")
print("Alarm set for:", watch.alarm)
watch.stop_alarm()
print("Alarm:", watch.alarm)
```

OUTPUT

```
09:30:45
Alarm set for: 10:00:00
Alarm: None
```

#3. Write Python Program to Simulate a Bank Account with Support for depositMoney, withdrawMoney and showBalance Operations.

```
class BankAccount:
  def __init__(self, balance=0):
    self.balance = balance

  def deposit_money(self, amount):
    self.balance += amount

  def withdraw_money(self, amount):
    if amount <= self.balance:
      self.balance -= amount
    else:
      print("Insufficient funds!")

  def show_balance(self):
    return f"Balance: {self.balance}"

account = BankAccount(100)
account.deposit_money(50)
print(account.show_balance())
account.withdraw_money(75)
print(account.show_balance())
account.withdraw_money(100)
print(account.show_balance())
```

OUTPUT

```
Balance: 150
Balance: 75
Insufficient funds!
Balance: 75
```

#4. Create class vehicle with attributes(color,capacity,enginpower,tyre) and behaviour (start,stop) Create class car which inherit vehicle class with

attributes(airbags,gear,speed,fuel,) and methods(accelerate ,fillfuel,playmusic(),onAC()) Create class electric car with attribute(battery) and behaviour(charging(),battery level().

```
class Vehicle:
  def __init__(self, color, capacity, engine_power, tyres):
    self.color = color
    self.capacity = capacity
    self.engine_power = engine_power
    self.tyres = tyres

  def start(self):
    print("Vehicle started.")

  def stop(self):
    print("Vehicle stopped.")

class Car(Vehicle):
  def __init__(self, color, capacity, engine_power, tyres, airbags, gear, speed, fuel):
    super().__init__(color, capacity, engine_power, tyres)
    self.airbags = airbags
    self.gear = gear
    self.speed = speed
    self.fuel = fuel

   def accelerate(self):
     self.speed += 10

  def fill_fuel(self, amount):
    self.fuel += amount

   def play_music(self):
    print("Playing music.")

  def on_ac(self):
    print("AC turned on.")

class ElectricCar(Car):
  def __init__(self, color, capacity, engine_power, tyres, airbags, gear, speed, battery):
    super().__init__(color, capacity, engine_power, tyres, airbags, gear, speed, fuel=0)
    self.battery = battery

  def charging(self):
    print("Car is charging.")

  def battery_level(self):
        return f"Battery level: {self.battery}%"

vehicle = Vehicle("Red", 5, "150 HP", 4)
vehicle.start()
vehicle.stop()

car = Car("Blue", 5, "200 HP", 4, airbags=6, gear="Automatic", speed=50, fuel=10)
car.start()
car.accelerate()
car.fill_fuel(20)
car.play_music()
car.on_ac()
print(f"Car color: {car.color}, Speed: {car.speed}, Fuel: {car.fuel}")

e_car = ElectricCar("Green", 5, "180 HP", 4, airbags=4, gear="Manual", speed=40,
battery=80)
e_car.start()
e_car.charging()
print(e_car.battery_level())
```

OUTPUT

```
Vehicle started.
Vehicle stopped.
Vehicle started.
Playing music.
AC turned on.
Car color: Blue, Speed: 60, Fuel: 30
Vehicle started.
Car is charging.
Battery level: 80%
```

#5. Define a class Person and its two child classes: Male and Female. All classes have a method "getGender" which can print "Male" for Male class and "Female" for Female class

```
class Person:
    def get_gender(self):
        pass

class Male(Person):
    def get_gender(self):
        return "Male"

class Female(Person):
    def get_gender(self):
        return "Female"
```

```
man = Male()
woman = Female()
print(man.get_gender())
print(woman.get_gender())
```

OUTPUT

```
    Male
    Female
```

#6. Implement Object-Oriented Programming (OOP) concepts in Python, including Polymorphism, Encapsulation, Inheritance, and Abstraction, using a pharmaceutical-related example

```
from abc import ABC, abstractmethod

class Medicine(ABC):
    @abstractmethod
    def use(self):
        pass

class Tablet(Medicine):
    def use(self):
        return "Swallow with water."

class Syrup(Medicine):
    def use(self):
        return "Take with a spoon."

class Capsule(Medicine):
    def use(self):
        return "Take with warm water."

class Doctor:
    def prescribe(self, medicine: Medicine):
        return medicine.use()

doc = Doctor()
tab = Tablet()
print(doc.prescribe(tab))

doc = Doctor()
tablet_medicine = Tablet()
syrup_medicine = Syrup()
capsule_medicine = Capsule()

print(doc.prescribe(tablet_medicine))
print(doc.prescribe(syrup_medicine))
print(doc.prescribe(capsule_medicine))
```

OUTPUT

```
Swallow with water.
Take with a spoon.
Take with warm water.
```

#7. Design a Python program to simulate different types of Drug Formulations using OOP principles. The program should:

1. Use Inheritance to create different drug formulations (e.g., Tablet, Capsule, Injection).

2. Implement Encapsulation to protect sensitive drug data.

3. Apply Abstraction to define a blueprint for drug formulations.

4. Demonstrate Polymorphism by overriding methods in different drug types

```
from abc import ABC, abstractmethod

class DrugFormulation(ABC):
    def __init__(self, name, dosage, manufacturer):
        self._name = name  # Encapsulation: Protecting sensitive drug data
        self._dosage = dosage
        self._manufacturer = manufacturer

    @abstractmethod
    def administer(self):
        pass

    def get_info(self):
        return f"{self._name} ({self._dosage}) by {self._manufacturer}"

class Tablet(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} tablet orally with water."

class Capsule(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} capsule with warm water."

class Injection(DrugFormulation):
    def administer(self):
        return f"Administer {self._name} injection intravenously."

def prescribe_drug(drug: DrugFormulation):
    return drug.administer()

tablet = Tablet("Paracetamol", "500mg", "PharmaCorp")
```

```
capsule = Capsule("Amoxicillin", "250mg", "MediHealth")
injection = Injection("Insulin", "10ml", "BioCare")

print(prescribe_drug(tablet))
print(prescribe_drug(capsule))
print(prescribe_drug(injection))
```

OUTPUT

```
    Administer Paracetamol tablet orally with water.
    Administer Amoxicillin capsule with warm water.
    Administer Insulin injection intravenously.
```