Course Code:ET24CS108U	Problem Solving & Programming – I Lab
Course Objectives:	
	This course helps in gaining foundational programming skills in Python, covering algorithm design, flowcharts, debugging, conditional and looping constructs, data structures, functions, object-oriented programming, file handling, and apply them to develop a comprehensive project.
Course Outcomes:	
CO1	Able to design and implement algorithms and flowcharts for problems.
CO2	Acquire skills to write and debug Python programs, addressing common issues such as indentation errors.
CO3	Effectively use conditional statements and loops in Python to solve problems
CO4	Manage and manipulate various data structures like strings, sets, lists, tuples, and dictionaries.
CO5	Apply advanced programming concepts and demonstrate their understanding through a comprehensive mini project.
List of Experiments:	
01	Write an algorithm and flowchart for following problems Write algorithm and make flowchart to find whether the given natural number n is a prime number or not. Determine if a given number is a Palindrome or not, write algorithm and make flowchart too. Fibonacci sequence is generated by adding the previous two terms by starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, .. write algorithm and make flowchart too.
02	Write program for following problems using basic python purposefully raise Indentation Error and Correct it Write a program to compute distance between two points taking input from the user (Pythagorean Theorem)
03	Write a program for following problem using Conditional statements Write a program to find largest of three numbers using ‘if statement’. Write a program to find given number ‘n’ is even or odd using ‘if else statement’. Write a program to test whether the number input by user is positive or negative, and if number is positive, whether it is even or odd using nested if statements. Write a program to print name of day of week when we are given day of week as number (0 for Sunday, 1 for Monday,….., 6 for Saturday).

04	Write program for following problems using Looping statements Write a program to find factorial of a whole number ‘n’ using ‘for loop’. Write a program to find sum of digits of a natural number using ‘while loop’. Write a program to print following pattern using ‘nested for loops’. 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 Write a program to illustrate the difference between between ‘break’ and ‘continue’ statements program uses for loop with range (1,11), i.e., list [1,2,3,4,5,6,7,8,9,10]
05	Write program for following problems using strings Write a program that reads a word in lowercase and capitalize its alternate letters; For example, it should print passion as PaSsIoN and radar as RaDaR. Write a program that reads a line of text and a word, and prints the number of time given word occurs (appears) in the line of text.
06	Write program for following problems on sets and lists
	Write a program to illustrate operation on sets (Union, Intersection, difference and comparing) Write a program to find the mean(average) of given list. Program should give output the mean rounded to two decimal digits. Write a program to count the frequency of each element in a given list.
07	Write program for following problems on tuples and dictionaries Write a program that takes a number as input and prints it in word. For example, if the input number 2370, it should print “Two Three Seven Zero”. Write a program that builds a dictionary of months of a year and prints the days in a month for the month name given as input.
08	Write program for following problems on Functions Write a function to find Highest Common Factor (HCF), also known as Greatest Common Divisor (GCD) of two positive integers m and n. Write a function, say int IsPrime(int n) that returns the value 1 if integer argument n represents a prime number else returns value 0.Use this function in a program to print all three digit prime numbers.
	Write program for following problems on classes and objects Write a program to simulate banking operation with class. 9.2 An employer plans to pay a bonus to all employees as per the following policy: Earning — Bonus; Upto Rs. 1,00,000/- — Nil; From Rs. 1,00,001/- to Rs.2,00,00/- — Rs.1000 + 10% of the excess over Rs. 1,00,000/-; From Rs. 2,00,001/- to Rs.3,00,000/- — Rs.2000 + 20% of the excess over Rs. 2,00,000/-; Above Rs.3,00,000/- — Rs.4000 + 30% of the excess over Rs. 2,00,000/-

	The input contains the name and earnings of an employee and the desired output is the name and bonus to be paid to the employee. Create a class to represent an employee. It should include the following: Data members:Name, Earning, Bonus Member Functions:To input data, To compute bonus, To output the desired information Using this class, write a program to accomplish the intended task
10	Write a python program for File handling methods 1 To open a file and print its attributes. To create a file and then read its contents and display them on the computer screen.
11	Mini Project based on the concepts covered in the syllabus
	Text Book:
	Y.Daniel Liang, “Introduction to Python Programming and Data Structures”, Third Edition, Pearson, 2024.
	Reference Books:
	Dr.R Nageswara Rao, “Core Python Programming”, Third Edition, Dreamtech Press, 2024.
	Ashok Namdev Kamthane, Amit Ashok Kamthane, “Programming and Problem Solving with Python”, Second Edition, TMH, 2020.
	Paul J. Deitel, Harvey Deitel, “Python for Programmers”, 4th Impression, Pearson India Education, 2022.
	Cay S. Horstmann, Rance D. Necaise, “Python For Everyone”, 3rd Edition, Wiley India, 2024
	Kenneth A.Labert, “Fundamentals of Python: First Programs with MindTap, Cengage, 2024.
	Web References:
01	https://www.udemy.com/course/python-programming-projects/?couponCode=IND21PM
	Alternative NPTEL/SWAYAM Course:
01	Programming in Python https://onlinecourses.swayam2.ac.in/cec24_cs11/preview

01. Write an algorithm and flowchart for following problems

1.1. Write algorithm and make flowchart to find whether the given natural number n is a prime number or not.

program

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

number = 20  # Change this to check other numbers
if is_prime(number):
    print(f"{number} is a prime number.")
else:
    print(f"{number} is not a prime number.")
```

Output

```
20 is not a prime number.
```

1.2. Determine if a given number is a Palindrome / not, write algorithm and make flowchart too.

program

```
num = 12
temp = num
reverse = 0
while temp > 0:
    remainder = temp % 10
    print(remainder)
    reverse = (reverse * 10) + remainder
    temp = temp // 10
if num == reverse:
  print('Palindrome')
else:
  print("Not Palindrome")
```

Output

```
Not Palindrome
```

1.3. Fibonacci sequence is generated by adding the previous two terms by starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, .. write algorithm and make flowchart too.

program

```
# Function to generate Fibonacci sequence
def fibonacci_sequence(terms):
    first = 1
    second = 2
    count = 0

    # Print the first two terms
    print(first)
    print(second)

    while count < (terms - 2):
        next_term = first + second
        print(next_term)
        first = second
        second = next_term
        count += 1

# Generate the first 10 terms
fibonacci_sequence(10)
```

Output

```
1 2 3 5 8 13 21 34 55 89
```

02. Write program for following problems using basic python

2.1 purposefully raise Indentation Error and Correct it

An IndentationError occurs in Python when there is an incorrect level of indentation. Below is an example of code that will raise an indentation error:

program

```
def greet(name):
    print("Hello,", name)
     print("Welcome to the programming world!")  # This line has incorrect indentation
# Example usage
greet("Alice")
```

Error Output

When you run this code, you will see an error similar to the following:

```
  File "script.py", line 3
    print("Welcome to the programming world!")  # This line has incorrect indentation
    ^
IndentationError: unexpected indent
```

Explanation of the Indentation Error

In the code above, the second print statement has an extra space before it, causing it to be incorrectly indented. In Python, consistent indentation is crucial because it defines the scope of loops, functions, and conditionals.

Correcting the Indentation Error
To fix the indentation error, ensure that the indentation levels are consistent. Here's the corrected code:

program

```
def greet(name):
    print("Hello,", name)
    print("Welcome to the programming world!")  # Corrected indentation
# Example usage
greet("Alice")
```

Error Output

```
Hello, Alice
Welcome to the programming world!
```

Explanation of the Correction

1. Consistent Indentation: Both print statements are now indented with the same number of spaces (typically 4 spaces is standard in Python).
2. Functionality: The corrected code will run without any errors.

Expected Output After Correction
When you run the corrected code, the output will be:

2.2 Write a program to compute distance between two points taking input from the user (Pythagorean Theorem)

program

```
import math
# Get coordinates from the user
x1 = float(input("Enter x1 coordinate of point 1: "))
y1 = float(input("Enter y1 coordinate of point 1: "))
x2 = float(input("Enter x2 coordinate of point 2: "))
y2 = float(input("Enter y2 coordinate of point 2: "))

# Calculate distance using the Pythagorean theorem
distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

# Display the result
print(f"The distance between point 1({x1}, {y1}) and point 2({x2}, {y2}) is: {distance}")
```

Output

```
Enter x1 coordinate of point 1:  1
Enter y1 coordinate of point 1:  2
Enter x2 coordinate of point 2:  3
Enter y2 coordinate of point 2:  4
The distance between point 1(1.0, 2.0) and point 2(3.0, 4.0) is: 
2.8284271247461903
```

03. Write a program for following problem using Conditional statements

3.1 Write a program to find largest of three numbers using ‘if statement’

program

```
#to find largest of 3 numbers using "if statement"
a = int(input("Enter a: "))
b = int(input("Enter b: "))
c = int(input("Enter c: "))
if a > b and a > c:
    print(a, "is the largest")
elif b > a and b > c:
    print(b, "is the largest")
else:
    print(c, "is the largest")
```

Output

```
Enter a:  3
Enter b:  1
Enter c:  2
3 is the largest
```

3.2 Write a program to find given number ‘n’ is even or odd using ‘if else statement’.

program

```
n=int(input("enter n:"))
if n%2==0:
    print(n,"is even number")
else:
    print(n,"is odd number")
```

Output

```
enter n: 5
5 is odd number
```

3.3 Write a program to test whether the number input by user is positive or negative, and if number is positive, whether it is even or odd using nested if statements.

program

```
number = float(input("Enter a number: "))
# Check if the number is positive or negative
if number >= 0:
    print(f"{number} is a positive number.")
    # Nested if to check if the positive number is even or odd
    if number % 2 == 0:
        print(f"{number} is an even number.")
    else:
        print(f"{number} is an odd number.")
else:
    print(f"{number} is a negative number.")
```

Output

```
Enter a number:  10
10.0 is a positive number.
10.0 is an even number.
```

3.4 Write a program to print name of day of week when we are given day of week as number (0 for Sunday, 1 for Monday,….., 6 for Saturday).

program

```
day_number = int(input("Enter day number (0-6): "))
# Loop to match the number with the day name
if day_number == 0:
    print("Sunday")
elif day_number == 1:
    print("Monday")
elif day_number == 2:
    print("Tuesday")
elif day_number == 3:
    print("Wednesday")
elif day_number == 4:
    print("Thursday")
elif day_number == 5:
    print("Friday")
elif day_number == 6:
    print("Saturday")
else:
    print("Invalid input! Please enter a number between 0 and 6.")
```

Output

```
Enter day number (0-6):  4
Thursday
```

04. Write program for following problems using Looping statements

4.1 Write a program to find factorial of a whole number ‘n’ using ‘for loop’.

program

```
# Input a whole number
n = int(input("Enter a whole number: "))
# Initialize the factorial value to 1
factorial = 1
# Calculate factorial using for loop
if n >= 0:
    for i in range(1, n + 1):
        factorial *= i
    print("Factorial of", n, "is", factorial)
else:
    print("Factorial is not defined for negative numbers.")
```

Output

```
Enter a whole number:  5
Factorial of 5 is 120
```

4.2 Write a program to find sum of digits of a natural number using ‘while loop’

program

```
# Input from the user
number = int(input("Enter a natural number: "))
# Initialize sum to 0
sum_digits = 0
# Process each digit using while loop
while number > 0:
    digit = number % 10      # Get the last digit
    sum_digits += digit      # Add the digit to the sum
    number = number // 10     # Remove the last digit
# Output the result
print("The sum of the digits is:", sum_digits)
```

Output

```
Enter a natural number:  121
The sum of the digits is: 4
```

4.3 Write a program to print following pattern using ‘nested for loops’.

program

```
n = 4           # Number of rows
for i in range(1, n + 1):
    for j in range(i):
        print(i, end=" ")
    print()
```

Output

```
1
2 2
3 3 3
4 4 4 4
```

4.4 Write a program to illustrate the difference between between ‘break’ and ‘continue’ statements program uses for loop with range (1,11), i.e., list [1,2,3,4,5,6,7,8,9,10]

program

```
print("Using break:")
for i in range(1, 11):
    if i == 6:
        break  # Terminates the loop when i is 6
    print(i, end=" ")
print("\n")  # Move to the next line

print("Using continue:")
for i in range(1, 11):
    if i == 6:
        continue  # Skips the iteration when i is 6, but continues the loop
    print(i, end=" ")
print()  # To end the line
```

Output

```
Using break: 1 2 3 4 5
Using continue: 1 2 3 4 5 7 8 9 10
```

05 Write program for following problems using strings
5.1 Write a program that reads a word in lowercase and capitalize its alternate letters; For example, it should print passion as PaSsIoN and radar as RaDaR.

program

```
# Read a word from the user
word = input("Enter a word in lowercase: ")
 # Initialize an empty string to store the result
result = ""
# Loop through the word and capitalize alternate letters
for index, letter in enumerate(word):
    if index % 2 == 0:
        result += letter.upper()  # Capitalize the letter if the index is even
    else:
        result += letter.lower()  # Keep the letter lowercase if the index is odd

print("Capitalized alternate letters:", result)
```

Output

```
Enter a word in lowercase:  passion
Capitalized alternate letters: PaSsIoN
Enter the word to count its occurrences:  this
The word 'this' appears 1 time(s) in the given text.
```

5.2 Write a program that reads a line of text and a word, and prints the number of time given word occurs (appears) in the line of text.

program

```
# Read a line of text from the user
text = input("Enter a line of text: ")
word = input("Enter the word to count its occurrences: ")
    # Count occurrences of the word in the text
word_count = text.lower().split().count(word.lower())  # Case insensitive count
print(f"The word '{word}' appears {word_count} time(s) in the given text.")
```

Output

```
Enter a line of text:  this is a new line
Enter the word to count its occurrences:  this
The word 'this' appears 1 time(s) in the given text.
```

06. Write program for following problems on sets and lists
6.1 Write a program to illustrate operation on sets (Union, Intersection, difference and comparing)

program

```
# Define two sets
set_a = {1, 2, 3, 4, 5}
set_b = {4, 5, 6, 7, 8}

# 1. Union
union_set = set_a.union(set_b)  # or set_a | set_b
print("Union:", union_set)  # Output: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
# 2. Intersection
intersection_set = set_a.intersection(set_b)  # or set_a & set_b
print("Intersection:", intersection_set)  # Output: {4, 5}

# 3. Difference
difference_set = set_a.difference(set_b)  # or set_a - set_b
print("Difference (A - B):", difference_set)  # Output: {1, 2, 3}

# 4. Symmetric Difference
symmetric_difference_set = set_a.symmetric_difference(set_b)  # or set_a ^ set_b
print("Symmetric Difference:", symmetric_difference_set)  # Output: {1, 2, 3, 6, 7, 8}

# 5. Subset
is_subset = set_a.issubset(set_b)  # Check if A is a subset of B
print("Is A a subset of B?", is_subset)  # Output: False

# 6. Superset
is_superset = set_a.issuperset({4, 5})  # Check if A is a superset of {4, 5}
print("Is A a superset of {4, 5}?", is_superset)  # Output: True

# 7. Adding Elements
set_a.add(9)
print("Set A after adding 9:", set_a)  # Output: {1, 2, 3, 4, 5, 9}

# 8. Removing Elements
set_b.remove(8)  # Raises KeyError if 8 is not present
print("Set B after removing 8:", set_b)  # Output: {4, 5, 6, 7}

# 9. Clearing a Set
set_a.clear()  # Removes all elements from set A
print("Set A after clearing:", set_a)  # Output: set()
```

Output

```
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference (A - B): {1, 2, 3}
Symmetric Difference: {1, 2, 3, 6, 7, 8}
Is A a subset of B? False
Is A a superset of {4, 5}? True
Set A after adding 9: {1, 2, 3, 4, 5, 9}
Set B after removing 8: {4, 5, 6, 7}
Set A after clearing: set()
```

6.2 Write a program to find the mean(average) of given list. Program should give output the mean rounded to two decimal digits.

program	from statistics import mean lst = [1.1111, 2.2222, 3.3333, 4.4444, 5.5555, 6.6666] list_avg = mean(lst) print("Average value of the list:\n", list_avg) print("Average value of the list with precision upto 3 decimal value:\n", round(list_avg,2))
Output	Average value of the list: 3.88885 Average value of the list with precision upto 3 decimal value: 3.89

6.3 Write a program to count the frequency of each element in a given list.

program	# initializing the list
	random_list = ['A', 'A', 'B', 'C', 'B', 'D', 'D', 'A', 'B']
	frequency = {}
	
	# iterating over the list
	for item in random_list:
	    # checking the element in dictionary
	    if item in frequency:
	        # incrementing the counter
	        frequency[item] += 1
	    else:
	        # initializing the count
	        frequency[item] = 1
	
	# printing the frequency
	print(frequency)

Output	{'A': 3, 'B': 3, 'C': 1, 'D': 2}

07. Write program for following problems on tuples and dictionaries
7.1 Write a program that takes a number as input and prints it in word. For example, if the input number 2370, it should print “Two Three Seven Zero”.

program	def printValue(digit):
	    if digit == '0':
	        print("Zero ", end = " ")
	    elif digit == '1':
	        print("One ", end = " ")
	    elif digit == '2':
	        print("Two ", end = " ")
	    elif digit=='3':
	        print("Three",end=" ")
	    elif digit == '4':
	        print("Four ", end = " ")
	    elif digit == '5':
	        print("Five ", end = " ")

```
	elif digit == '6':
		print("Six ", end = " ")
	elif digit == '7':
		print("Seven", end = " ")
	elif digit == '8':
		print("Eight", end = " ")
	elif digit == '9':
		print("Nine ", end = " ")

# Function to iterate through every
# digit in the given number
def printWord(N):
	i = 0
	length = len(N)
	# Finding each digit of the number
	while i < length:
		# Print the digit in words
		printValue(N[i])
		i += 1

# Driver code
N = "2370"
printWord(N)
```

Output

```
Two  Three Seven Zero
```

7.2 Write a program that builds a dictionary of months of a year and prints the days in a month for the month name given as input.

program

```
  # Dictionary of months and their corresponding number of days
 # Note: ignoring leap years for simplicity
  months = {
     "January": 31,       "February": 28,            "March": 31,
     "April": 30,         "May": 31,                 "June": 30,
     "July": 31,          "August": 31,              "September": 30,
     "October": 31,      "November": 30,            "December": 31
  }

  # Get month name input from the user
  month_name = input("Enter the name of a month: ").strip()
  # Check if the month exists in the dictionary and print the days
  if month_name in months:
     print(f"The month of {month_name} has {months[month_name]} days.")
  else:
     print("Invalid month name. Please enter a valid month (e.g., January, February,
etc.).")
```

Output

```
Enter the name of a month:  August
```

```
The month of August has 31 days.
```

08. Write program for following problems on Functions

8.1 Write a function to find Highest Common Factor (HCF), also known as Greatest Common Divisor (GCD) of two positive integers m and n.

program

```
x = 54
y = 24

if x > y:
        smaller = y
else:
        smaller = x
for i in range(1, smaller+1):
        if((x % i == 0) and (y % i == 0)):
            hcf = i
print("The H.C.F. is", hcf)
```

Output

```
The H.C.F. is 6
```

8.2 Write a function, say int IsPrime(int n) that returns the value 1 if integer argument n represents a prime number else returns value 0.Use this function in a program to print all three digit prime numbers.

program

```
def is_prime(n):
    if n <= 1:  # 0 and 1 are not prime numbers
        return 0
    for i in range(2, int(n**0.5) + 1):  # Check for factors from 2 to √n
        if n % i == 0:  # If n is divisible by i, it's not prime
            return 0
    return 1  # n is prime

def print_three_digit_primes():
    """
    Print all three-digit prime numbers.
    """
    print("Three-digit prime numbers:")
    for num in range(100, 1000):  # Iterate through all three-digit numbers
        if is_prime(num):  # Check if the number is prime
            print(num, end=' ')  # Print the prime number

# Run the function to print three-digit primes
print_three_digit_primes()
```

Output

```
101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193
197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307
```

	311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 997

09. Write program for following problems on classes and objects

program

```
class Car:
    def __init__(self, brand, model, year):
        """Initialize the car's attributes."""
        self.brand = brand
        self.model = model
        self.year = year

    def abcd(self):
        """Make the car honk."""
        return "Car Details"

    def display_info(self):
        """Display the car's information."""
        print(f"Car Brand: {self.brand}")
        print(f"Car Model: {self.model}")
        print(f"Car Year: {self.year}")


# Example usage
if __name__ == "__main__":
    # Create a Car object
    my_car = Car("Honda City", "Mahindra", 2020)

    # Call methods on the Car object
    print(my_car.abcd())  # Output: Honk! Honk!
    my_car.display_info()  # Display car details
```

Output

```
Car Details
Car Brand: Honda City
Car Model: Mahindra
Car Year: 2020
```

9. Write program for following problems on classes and objects

9.1Write a program to simulate banking operation with class.

program

```
class BankAccount:
    def __init__(self, account_holder):
        self.account_holder = account_holder
        self.balance = 0.0

    def deposit(self, amount):
        """Deposit money into the account."""
        self.balance += amount
        print(f"Deposited: ${amount:.2f}. New balance: ${self.balance:.2f}")

    def withdraw(self, amount):
        """Withdraw money from the account if sufficient balance."""
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: ${amount:.2f}. New balance: ${self.balance:.2f}")
        else:
            print("Insufficient funds!")

    def get_balance(self):
        """Return the current balance."""
        return self.balance

# Example usage
if __name__ == "__main__":
    # Create a bank account object
    account = BankAccount("Prasad")

    # Perform operations
    account.deposit(1000)
    account.withdraw(300)
    print(f"Account balance: ${account.get_balance():.2f}")
```

Output

```
Deposited: $1000.00. New balance: $1000.00
Withdrew: $300.00. New balance: $700.00
Account balance: $700.00
```

9.2 An employer plans to pay a bonus to all employees as per the following policy:

Earning	Bonus
Upto Rs. 1,00,000/-	Nil
From Rs. 1,00,001/- to Rs.2,00,00/-	Rs.1000 + 10% of the excess over Rs. 1,00,000/-
From Rs. 2,00,001/- to Rs.3,00,000/-	Rs.2000 + 20% of the excess over Rs. 2,00,000/-
Above Rs.3,00,000/-	Rs.4000 + 30% of the excess over Rs. 2,00,000/-

CODE

program

```
class Employee:
    def __init__(self):
        self.name = ""
        self.earning = 0.0
        self.bonus = 0.0

    def input_data(self):
        """Method to input employee data."""
        self.name = input("Enter employee name: ")
        self.earning = float(input("Enter employee earning: "))

    def compute_bonus(self):
        """Method to compute the bonus based on earnings."""
        # Assuming the bonus is 10% of the earnings
        if self.earning  < 100000:
            self.bonus = self.earning
        if 100000 <= self.earning <= 200000:
            self.bonus = self.earning + self.earning * 0.10
        if 200000 <= self.earning <= 300000:
            self.bonus = self.earning + self.earning * 0.20
        if self.earning > 300000:
            self.bonus = self.earning + self.earning * 0.30

    def output_info(self):
        """Method to output the employee's name and bonus."""
        print(f"Employee Name: {self.name}")
        print(f"Earnings: {self.earning}")
        print(f"Bonus to be Paid: {self.bonus}")

# Example usage of the Employee class
def main():
    employee = Employee()  # Create an Employee object
    employee.input_data()  # Input employee data
    employee.compute_bonus()  # Compute the bonus
    employee.output_info()  # Output the employee's information

# Run the program
if __name__ == "__main__":
```

	main()
Output	Enter employee name: a Enter employee earning: 300000 Employee Name: a Earnings: 300000.0 Bonus to be Paid: 360000.0

10 . Write a python program for File handling methods

10.1 To open a file and print its attributes.

File Name	a.py

program	import os
	import time
	
	def print_file_attributes(filename):
	    """Print the attributes of the specified file."""
	    if os.path.exists(filename):
	        # Get file size
	        file_size = os.path.getsize(filename)
	
	        # Get file modification time
	        modification_time = os.path.getmtime(filename)
	
	        # Get file creation time
	        creation_time = os.path.getctime(filename)
	
	        # Get file permissions
	        file_permissions = os.stat(filename).st_mode
	
	        print(f"File: {filename}")
	        print(f"Size: {file_size} bytes")
	        print(f"Created: {time.ctime(creation_time)}")
	        print(f"Last Modified: {time.ctime(modification_time)}")
	        print(f"Permissions: {oct(file_permissions)}")
	# Convert to octal for better readability
	    else:
	        print(f"The file '{filename}' does not exist.")
	
	# Example usage
	if __name__ == "__main__":
	    # Specify the filename
	    filename = 'example.txt' # Change this to the file you want to check
	
	    # Call the function to print file attributes

```
    print_file_attributes(filename)
```

Run

```
python a.py
```

Output

```
File: example.txt
Size: 68 bytes
Created: Tue Oct 29 10:59:56 2024
Last Modified: Tue Oct 29 11:01:38 2024
Permissions: 0o100666
```

10.2 To create a file and then read its contents and display them on the computer screen.

File Name: a.py

program

```python
# Function to create a file and write content to it
def create_file(filename, content):
    with open(filename, 'w') as file:  # Open file in write mode
        file.write(content)  # Write content to the file
    print(f"File '{filename}' created and content written.")

# Function to read the contents of a file
def read_file(filename):
    with open(filename, 'r') as file:  # Open file in read mode
        contents = file.read()  # Read the entire file
    return contents

# Example usage
if __name__ == "__main__":
    # Specify the filename and content
    filename = 'example.txt'
    content = "Hello, this is a sample text file.\nWelcome to Python file handling."

    # Create a file and write content
    create_file(filename, content)

    # Read the file and display its contents
    file_contents = read_file(filename)
    print("\nContents of the file:")
    print(file_contents)
```

Run

```
python a.py
```

Output

```
File 'example.txt' created and content written.
Contents of the file:
Hello, this is a sample text file.
Welcome to Python file handling.
```