**ISHAN SAMANTA**

PREFACE

Learning Python becomes easier when you have clear examples and ready-to-use solutions.  
 This book is created as a **practical handbook** for students, beginners, and anyone who wants quick reference code for common Python problems. Instead of long explanations, the focus here is on clean and easy-to-understand programs.

Each topic—loops, strings, functions, recursion, OOP, file handling, NumPy, pandas, and more—is covered with simple examples that can be practiced immediately. These solutions are written in a beginner-friendly style so readers can understand the logic and apply it to their own programs.

This book is designed to serve as a companion for Python learners preparing for exams, interviews, school/college assignments, or daily practice. All programs are written in a straightforward manner to encourage hands-on learning and confidence in coding.

I hope this book becomes a helpful and reliable resource in your Python learning journey.

ACKNOWLEDGEMENT

I would like to express my gratitude to everyone who supported me while creating this book. Their encouragement helped me stay motivated throughout the writing process. I am also thankful to the Python learning community, whose continuous curiosity and enthusiasm inspired me to put together this collection of solutions.

To all the readers—thank you for choosing this book as part of your learning path.

About the Author

**Ishan Samanta** is a first-year Computer Science and Engineering student with a growing interest in programming, operating systems, and practical computing. He enjoys learning by building things—from small CLI tools to larger personal projects—and believes that understanding the basics strongly is the key to mastering advanced topics later.

Ishan created this handbook to help students like him who want quick access to important exam-oriented Python questions in one place. While preparing for his own university exams, he realised that beginners often struggle to find a structured set of problems to practice. This inspired him to compile the most useful questions and present them in a simple, easy-to-understand format.

He continues to learn Python, C, data structures and plans to share more student-friendly resources in the future. This book is one of his first steps toward contributing helpful material to the programming community.

His github link:<https://github.com/ishansamanta>

**PREREQUISITES**

“This book assumes basic understanding of Python syntax such as variables, loops, and functions.”

INTRODUCTION

Python is one of the most beginner-friendly programming languages, and it is widely used in many fields—web development, data science, automation, machine learning, and more. However, beginners often struggle to find clean and simple examples they can practice immediately.

This book was created to solve that problem.

It contains **150+ Python programs**, covering topics such as:

* Basic syntax
* Loops
* Strings
* Functions
* Recursion
* Object-Oriented Programming
* File handling
* Exception handling
* NumPy
* Pandas
* Pattern printing
* Number series

Each program is short, easy to understand, and written with beginners in mind.

You can use this book as a practice companion, a quick revision guide, or a reference for assignments.If you keep practicing regularly, you will build a strong foundation that will support your journey into advanced Python topics.

CONTENTS

### *TOPIC: BASIC I/O*

1. [Basic Input/Output] Write a program to take two numbers as input and print their sum.

2. [Basic Input/Output] Take user's name and age, and display them in a formatted message.

3. [Basic Input/Output] Write a program to swap two numbers using a temporary variable.

4. [Basic Input/Output] Write a program to calculate simple interest.

5. [Basic Input/Output] Write a program to convert Celsius to Fahrenheit.

6. [Basic Input/Output] Take three inputs and print the largest.

7. [Basic Input/Output] Write a program to check if a number is positive, negative, or zero.

8. [Basic Input/Output] Write a program to calculate the area of a triangle using base and height.

9. [Basic Input/Output] Write a program to read a character and print its ASCII value.

10. [Basic Input/Output] Write a program to calculate compound interest.

### *TOPIC: IF-ELSE*

11. [If-Else] Check if a number is even or odd.

12. [If-Else] Check if a year is a leap year.

13. [If-Else] Write a program to check if a character is a vowel or consonant.

14. [If-Else] Write a program to check if a person is eligible to vote.

15. [If-Else] Check if a number is divisible by 3 and 5.

16. [If-Else] Write a program to determine the grade of a student based on marks.

17. [If-Else] Check whether a given string is a palindrome.

18. [If-Else] Find the largest of three numbers.

19. [If-Else] Check if a number is prime.

20. [If-Else] Check whether a triangle is valid based on its sides.

### *TOPIC: LOOPS*

21. [Loops] Print numbers from 1 to 100.

22. [Loops] Print the multiplication table of any number.

23. [Loops] Calculate the factorial of a number using a loop.

24. [Loops] Reverse a number using a loop.

25. [Loops] Count digits of a number.

26. [Loops] Print all even numbers between 1 and N.

27. [Loops] Print Fibonacci series up to N terms.

28. [Loops] Find the sum of digits of a number.

29. [Loops] Print numbers divisible by 7 between a given range.

30. [Loops] Generate the first N prime numbers.

### *TOPIC: STRINGS*

31. [Strings] Count vowels and consonants in a string.

32. [Strings] Reverse a string without using slicing.

33. [Strings] Check if two strings are anagrams.

34. [Strings] Count words in a sentence.

35. [Strings] Remove extra spaces from a sentence.

36. [Strings] Find frequency of each character in a string.

37. [Strings] Capitalize the first letter of each word in a string.

38. [Strings] Write a program to remove all digits from a string.

39. [Strings] Replace all vowels in a string with '\*'.

40. [Strings] Find the longest word in a sentence.

### *TOPIC: FUNCTIONS*

41. [Functions] Write a function to calculate factorial.

42. [Functions] Write a function to check palindromes.

43. [Functions] Write a function to return the greatest of three numbers.

44. [Functions] Write a function to check the prime number.

45. [Functions] Write a function that returns Fibonacci series.

46. [Functions] Write a function to count uppercase letters.

47. [Functions] Write a function to find the GCD of two numbers.

48. [Functions] Write a function to convert decimal to binary.

49. [Functions] Write a function to calculate simple interest.

50. [Functions] Write a function to check Armstrong number.

### *TOPIC: RECURSIONS*

51. [Recursion] Find factorial of a number using recursion.

52. [Recursion] Print Fibonacci series using recursion.

53. [Recursion] Find sum of digits of a number using recursion.

54. [Recursion] Reverse a string using recursion.

55. [Recursion] Find GCD of two numbers using recursion.

56. [Recursion] Find power of a number using recursion.

57. [Recursion] Calculate nCr using recursion.

58. [Recursion] Find product of digits of a number recursively.

59. [Recursion] Check palindrome using recursion.

60. [Recursion] Print numbers from 1 to N using recursion.

### *TOPIC: BASIC OBJECT ORIENTED PROGRAMMING*

61. [Basic OOP] Create a class Student and print details.

62. [Basic OOP] Create a class Rectangle with area() and perimeter().

63. [Basic OOP] Create a class Bank Account with deposit and withdrawal methods.

64. [Basic OOP] Create a class Car with attributes and display methods.

65. [Basic OOP] Create a class Employee with salary increment function.

66. [Basic OOP] Create a class Book with title, author, and display methods.

67. [Basic OOP] Create a class Calculator with add, subtract, multiply, divide.

68. [Basic OOP] Create a class Complex to add two complex numbers.

69. [Basic OOP] Create a class Queue with enqueue and dequeue methods.

70. [Basic OOP] Create a class to store and display marks of students.

### *TOPIC: FILE HANDLING*

71. [File Handling] Write a program to read a file and print its content.

72. [File Handling] Write a program to count lines in a file.

73. [File Handling] Write a program to write user input to a text file.

74. [File Handling] Write a program to copy one file into another.

75. [File Handling] Write a program to count vowels in a file.

76. [File Handling] Write a program to find the longest line in a file.

77. [File Handling] Write a program to append data to an existing file.

78. [File Handling] Write a program to search a word in a text file.

79. [File Handling] Write a program to remove blank lines from a file.

80. [File Handling] Write a program to check if a file exists.

### *TOPIC: EXCEPTION HANDLING*

81. [Exception Handling] Write a program to handle division by zero.

82. [Exception Handling] Write a program to handle invalid value input.

83. [Exception Handling] Write a program to handle file not found errors.

84. [Exception Handling] Write a program that raises and handles a custom exception.

85. [Exception Handling] Write a program with multiple exception blocks.

86. [Exception Handling] Write a program that handles type conversion errors.

87. [Exception Handling] Write a program to demonstrate try-else blocks.

88. [Exception Handling] Write a program to demonstrate finally block.

89. [Exception Handling] Write a safe input function using exceptions.

90. [Exception Handling] Write a program to catch index errors in lists.

### *TOPIC: NUMBER & SERIES PROBLEMS*

91. [Number Problems] Check Armstrong number.

92. [Number Problems] Check Krishnamurthy number.

93. [Number Problems] Check Perfect number.

94. [Number Problems] Check Abundant numbers.

95. [Number Problems] Check Harshad number.

96. [Number Problems] Find LCM of two numbers.

97. [Number Problems] Find GCD of two numbers.

98. [Number Problems] Check Automorphic numbers.

99. [Number Problems] Check Strong numbers.

100. [Number Problems] Check Neon number.

101. [Series Problems] Print arithmetic progression.

102. [Series Problems] Print geometric progression.

103. [Series Problems] Generate series: 1, 4, 9, 16… (squares).

104. [Series Problems] Generate series: 1, 8, 27… (cubes).

105. [Series Problems] Generate series: 1, 2, 4, 7, 11…

106. [Series Problems] Generate Fibonacci series.

107. [Series Problems] Generate prime number series.

108. [Series Problems] Generate harmonic series.

109. [Series Problems] Generate odd number series.

110. [Series Problems] Generate even number series.

### *TOPIC: PATTERN PROBLEMS*

111. [Patterns] Print a square pattern of \*.

112. [Patterns] Print a right triangle of numbers.

113. [Patterns] Print Floyd's triangle.

114. [Patterns] Print Pascal's triangle.

115. [Patterns] Print inverted pyramid.

116. [Patterns] Print number pyramid 1 22 333.

117. [Patterns] Print alphabet triangle.

118. [Patterns] Print hollow squares.

119. [Patterns] Print diamond pattern.

120. [Patterns] Print butterfly pattern.

### 

### *TOPIC: NUMPY*

121. [NumPy] Create a NumPy array and print its shape.

122. [NumPy] Perform element-wise addition of two NumPy arrays.

123. [NumPy] Find mean, median, and variance of array.

124. [NumPy] Reshape a NumPy array.

125. [NumPy] Slice a NumPy array.

126. [NumPy] Stack two arrays horizontally.

127. [NumPy] Stack two arrays vertically.

128. [NumPy] Generate an array of zeros.

129. [NumPy] Generate an array of random integers.

130. [NumPy] Find unique elements in an array.

### *TOPIC: MATPLOTLIB*

131. [Matplotlib] Plot a simple line graph.

132. [Matplotlib] Plot a bar chart of student marks.

133. [Matplotlib] Plot a pie chart of expenses.

134. [Matplotlib] Plot sine wave using NumPy + Matplotlib.

135. [Matplotlib] Plot scatter plot between two variables.

136. [Matplotlib] Plot histogram of heights.

137. [Matplotlib] Add labels and title to the graph.

138. [Matplotlib] Customize line style and width.

139. [Matplotlib] Create a subplot with 2 graphs.

140. [Matplotlib] Save a plot as an image.

### 

### *TOPIC: PANDAS*

141. [Pandas] Create a DataFrame from a dictionary.

142. [Pandas] Read a CSV file using pandas.

143. [Pandas] Display first and last 5 rows of a DataFrame.

144. [Pandas] Sort DataFrame by a column.

145. [Pandas] Select specific rows and columns.

146. [Pandas] Filter DataFrame using conditions.

147. [Pandas] Add a new column to DataFrame.

148. [Pandas] Drop a column from DataFrame.

149. [Pandas] Handle missing values.

150. [Pandas] Group data and calculate mean.

***##AFTER THE 150 QUESTIONS, THERE ARE ANOTHER 20 BONUS QUESTIONS.##***

### *TOPIC: BASIC I/O*

SOLUTIONS:

1. [Basic Input/Output] Write a program to take two numbers as input and print their sum.

n1= int(input("Enter your first number: ")

n2= int(input("Enter your second number: ")

sums= n1+n2

print("The summation of the two numbers are: ", sums)

2. [Basic Input/Output] Take user's name and age, and display them in a formatted message.

username = input("Enter the user's name": )

age = int(input("Enter the age of the user: ")

print(f"The name of the user is {username} and the he/she is {age} years old.")

3. [Basic Input/Output] Write a program to swap two numbers using a temporary variable.

n1= int(input("Enter the first number: "))

n2= int(input("Enter the second number: "))

print("The numbers before swapping are:", n1 , n2)

temp=n2

n2=n1

n1= temp

print("The numbers after swapping are: ", n1 , n2)

4. [Basic Input/Output] Write a program to calculate simple interest.

p= float(input("Enter the principal amount: "))

r= float(input("Enter the rate %: "))

t= float(input("Enter the time span: "))

simple\_interest = (p\*r\*t)/100

print("The simple interest is: ", simple\_interest)

5. [Basic Input/Output] Write a program to convert Celsius to Fahrenheit.

temp\_c = float(input("Enter the temperature in Celsius (°C): "))

temp\_f = (9/5 \* temp\_c)+32

print(f"Temperature in Celsius :{temp\_c}°C to fahrenheit is: {temp\_f}°F")

6. [Basic Input/Output] Take three inputs and print the largest.

n1= int(input("Enter the first number: "))

n2= int(input("Enter the second number: "))

n3= int(input("Enter the third number: " ))

maximum= max(n1, n2, n3)

print("The largest number is : ", maximum)

7. [Basic Input/Output] Write a program to check if a number is positive, negative, or zero.

num = float(input("Enter your number: "))

if num>0:

print("The number is positive,")

elif num<0:

print("The number is negative.")

else:

print("The number is zero.")

8. [Basic Input/Output] Write a program to calculate the area of a triangle using base and height.

base = float(input("Enter the base length of the triangle (only units): "))

height = float(input("Enter the height length of the triangle (only units): "))

area = 1/2 \*base\*height

print("The area of the triangle is: ", area))

9. [Basic Input/Output] Write a program to read a character and print its ASCII value.

char\_input = input("Enter a character: ")

if len(char\_input) == 1:

ascii\_value = ord(char\_input)

print(f"The ASCII value of '{char\_input}' is {ascii\_value}")

else:

print("Please enter only a single character.")

10. [Basic Input/Output] Write a program to calculate compound interest.

principal = float(input("Enter the principal amount: "))

rate = float(input("Enter the rate %: "))

time = float(input("Enter the number of years: "))

amount = principal \* ((1 + rate/100) \*\* time)

compound\_interest = amount - principal

print("Final amount after compound interest:", amount)

print("Total compound interest earned:", compound\_interest)

### *TOPIC: IF-ELSE*

SOLUTIONS:

11. [If-Else] Check if a number is even or odd.

num= int(input("Enter your number: "))

if num%2==0:

print("The given number is even")

else:

print("The given number is odd.")

12. [If-Else] Check if a year is a leap year.

year= int(input("Enter the year: "))

if (year%4==0 and year%100!=0) or (year%400==0):

print("The given year is leap year.")

else:

print("The given year is not a leap year.")

13. [If-Else] Write a program to check if a character is a vowel or consonant.

user= input("Enter your character")

if len(user)==1 and user.isalpha():

user = user.upper()

#this code is without using list

if user in "AEIOU":

print("The character is a vowel.")

else:

print("The character is a consonant.")

14. [If-Else] Write a program to check if a person is eligible to vote.

user= int(input("Enter your age: "))

if user>=18:

print("You are eligible to vote.")

else:

print("You are now eligible to vote.")

15. [If-Else] Check if a number is divisible by 3 and 5.

user= int(input("Enter the number: "))

if user%3==0 and user%5==0:

print("The number is divisible by both 3 and 5")

else:

print("The number is not divisible by both 3 and 5")

16. [If-Else] Write a program to determine the grade of a student based on marks.

### Sample Grading Criteria:

**You can adjust these ranges if needed:**

* **90–100 → Grade A**
* **80–89 → Grade B**
* **70–79 → Grade C**
* **60–69 → Grade D**
* **Below 60 → Grade F**

user\_marks= int(input("Enter your marks: "))

if 90<=user\_marks<=100:

print("Grade A")

elif 80<=user\_marks<=89:

print("Grade B")

elif 70<=user\_marks<=79:

print("Grade C")

elif 60<=user\_marks<=69:

print("Grade D")

elif user\_marks<60:

print("Grade F")

17. [If-Else] Check whether a given string is a palindrome.

string= input("Enter your string: ")

string=string.lower()

string2= string[::-1]

if string2==string:

print("The given string is palindrome.")

else:

print("The given string is not palindrome,.")

18. [If-Else] Find the largest of three numbers.

num1= int(input("Enter the first number: "))

num2= int(input("Enter the seond number: "))

num3= int(input("Enter the third number: "))

max= num1

if num2>max :

max=num2

if num3>max :

max=num3

print("The largest number is: ", max)

19. [If-Else] Check if a number is prime.

num = int(input("Enter the number: "))

if num <= 1:

print("The number is not a prime number.")

else:

for i in range(2, num):

if num % i == 0:

print("The number is not a prime number.")

break

else:

print("The number is a prime number.")

20. [If-Else] Check whether a triangle is valid based on its sides.

### Triangle Validity Rule:

### A triangle is valid if each side is less than the sum of the other two sides:

* a + b > c
* a + c > b
* b + c > a

a= int(input("Enter the first side of the triangle: "))

b= int(input("Enter the second side of the triangle:"))

c= int(input("Enteer the thirds side of the triangle: "))

if (a+b>c) or (a+c>b) or (b+c>b):

print("The triangle is valid.")

else :

print("The triangle is not valid.")

### *TOPIC: LOOPS*

SOLUTIONS:

21. [Loops] Print numbers from 1 to 100.

for i in range(1,101):

print(i)

22. [Loops] Print the multiplication table of any number.

i = int(input("Enter the number whose multiplication table you want: "))

for j in range(1, 11):

print(f"{i} x {j} = {i\*j}")

23. [Loops] Calculate the factorial of a number using a loop.

n = int(input("Enter a number: "))

fact = 1

for i in range(1, n + 1):

fact \*= i

print("Factorial =", fact)

24. [Loops] Reverse a number using a loop.

num = int(input("Enter a number: "))

rev = 0

while num > 0:

digit = num % 10

rev = rev \* 10 + digit

num //= 10

print("Reversed number:", rev)

25. [Loops] Count digits of a number.

num = int(input("Enter a number: "))

count = 0

temp = num

while temp > 0:

count += 1

temp //= 10

print("Total digits:", count)

26. [Loops] Print all even numbers between 1 and N.

n = int(input("Enter N: "))

for i in range(1, n + 1):

if i % 2 == 0:

print(i)

27. [Loops] Print Fibonacci series up to N terms.

n = int(input("Enter number of terms: "))

a, b = 0, 1

for i in range(n):

print(a, end=" ")

a, b = b, a + b

28. [Loops] Find the sum of digits of a number.

num = int(input("Enter a number: "))

total = 0

temp = num

while temp > 0:

total += temp % 10

temp //= 10

print("Sum of digits:", total)

29. [Loops] Print numbers divisible by 7 between a given range.

start = int(input("Enter start of range: "))

end = int(input("Enter end of range: "))

for i in range(start, end + 1):

if i % 7 == 0:

print(i)

30. [Loops] Generate the first N prime numbers.

n = int(input("Enter how many prime numbers you want: "))

count = 0

num = 2

while count < n:

is\_prime = True

for i in range(2, num):

if num % i == 0:

is\_prime = False

break

if is\_prime:

print(num)

count += 1

num += 1

### *TOPIC: STRINGS*

SOLUTIONS:

31. [Strings] Count vowels and consonants in a string.

text = input("Enter a string: ")

vowels = "aeiouAEIOU"

v\_count = 0

c\_count = 0

for ch in text:

if ch.isalpha():

if ch in vowels:

v\_count += 1

else:

c\_count += 1

print("Vowels:", v\_count)

print("Consonants:", c\_count)

32. [Strings] Reverse a string without using slicing.

text = "abc"

rev = ""

for ch in text:

rev = ch + rev

print(rev)

33. [Strings] Check if two strings are anagrams.

s1 = input("Enter first string: ").replace(" ", "").lower()

s2 = input("Enter second string: ").replace(" ", "").lower()

if sorted(s1) == sorted(s2):

print("They are anagrams")

else:

print("They are not anagrams")

34. [Strings] Count words in a sentence.

sentence = input("Enter a sentence: ")

words = sentence.split()

print("Number of words:", len(words))

35. [Strings] Remove extra spaces from a sentence.

sentence = input("Enter a sentence: ")

cleaned = " ".join(sentence.split())

print("Cleaned sentence:", cleaned)

36. [Strings] Find frequency of each character in a string.

text = input("Enter a string: ")

freq = {}

for ch in text:

if ch in freq:

freq[ch] += 1

else:

freq[ch] = 1

print("Character frequencies:")

for ch, count in freq.items():

print(ch, ":", count)

37. [Strings] Capitalize the first letter of each word in a string.

text = input("Enter a sentence: ")

words = text.split()

result = ""

for w in words:

result += w[0].upper() + w[1:] + " "

print(result.strip())

38. [Strings] Write a program to remove all digits from a string.

text = input("Enter a string: ")

result = ""

for ch in text:

if not ch.isdigit():

result += ch

print("String without digits:", result)

39. [Strings] Replace all vowels in a string with '\*'.

text = input("Enter a string: ")

vowels = "aeiouAEIOU"

result = ""

for ch in text:

if ch in vowels:

result += "\*"

else:

result += ch

print("Modified string:", result)

40. [Strings] Find the longest word in a sentence.

sentence = input("Enter a sentence: ")

words = sentence.split()

longest = ""

for w in words:

if len(w) > len(longest):

longest = w

print("Longest word:", longest)

### *TOPIC: FUNCTIONS*

SOLUTIONS:

41. [Functions] Write a function to calculate factorial.

def factorial(n):

fact = 1

for i in range(1, n + 1):

fact \*= i

return fact

42. [Functions] Write a function to check palindromes.

def is\_palindrome(text):

text = text.lower().replace(" ", "")

return text == text[::-1]

43. [Functions] Write a function to return the greatest of three numbers.

def greatest(a, b, c):

if a >= b and a >= c:

return a

elif b >= a and b >= c:

return b

else:

return c

44. [Functions] Write a function to check the prime number.

def is\_prime(num):

if num <= 1:

return False

for i in range(2, num):

if num % i == 0:

return False

return True

45. [Functions] Write a function that returns Fibonacci series.

def fibonacci(n):

series = []

a, b = 0, 1

for i in range(n):

series.append(a)

a, b = b, a + b

return series

46. [Functions] Write a function to count uppercase letters.

def count\_uppercase(text):

count = 0

for ch in text:

if ch.isupper():

count += 1

return count

47. [Functions] Write a function to find the GCD of two numbers.

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

48. [Functions] Write a function to convert decimal to binary.

def dec\_to\_bin(num):

binary = ""

if num == 0:

return "0"

while num > 0:

binary = str(num % 2) + binary

num //= 2

return binary

49. [Functions] Write a function to calculate simple interest.

def simple\_interest(p, r, t):

return (p \* r \* t) / 100

50. [Functions] Write a function to check Armstrong number.

def is\_armstrong(num):

temp = num

total = 0

while temp > 0:

digit = temp % 10

total += digit \*\* 3

temp //= 10

return total == num

### *TOPIC: RECURSIONS*

SOLUTIONS:

51. [Recursion] Find factorial of a number using recursion.

def factorial(n):

if n == 0 or n == 1:

return 1

return n \* factorial(n - 1)

52. [Recursion] Print Fibonacci series using recursion.

def fib(n):

if n <= 1:

return n

return fib(n - 1) + fib(n - 2)

def print\_fib\_series(n):

for i in range(n):

print(fib(i), end=" ")

53. [Recursion] Find sum of digits of a number using recursion.

def sum\_digits(num):

if num == 0:

return 0

return (num % 10) + sum\_digits(num // 10)

54. [Recursion] Reverse a string using recursion.

def reverse\_string(s):

if s == "":

return s

return reverse\_string(s[1:]) + s[0]

55. [Recursion] Find GCD of two numbers using recursion.

def gcd(a, b):

if b == 0:

return a

return gcd(b, a % b)

56. [Recursion] Find power of a number using recursion.

def power(base, exp):

if exp == 0:

return 1

return base \* power(base, exp - 1)

57. [Recursion] Calculate nCr using recursion.

**Formula:**

**nCr = n! / (r! × (n − r)!)**

**Recursive form:**

**C(n, r) = C(n - 1, r - 1) + C(n - 1, r)**

def nCr(n, r):

if r == 0 or r == n:

return 1

return nCr(n - 1, r - 1) + nCr(n - 1, r)

58. [Recursion] Find product of digits of a number recursively.

def product\_digits(num):

if num < 10:

return num

return (num % 10) \* product\_digits(num // 10)

59. [Recursion] Check palindrome using recursion.

def is\_palindrome(s):

if len(s) <= 1:

return True

if s[0] != s[-1]:

return False

return is\_palindrome(s[1:-1])

60. [Recursion] Print numbers from 1 to N using recursion.

def print\_numbers(n):

if n == 0:

return

print\_numbers(n - 1)

print(n)

### *TOPIC: BASIC OBJECT ORIENTED PROGRAMMING*

SO;UTIONS:

61. [Basic OOP] Create a class Student and print details.

class Student:

def \_\_init\_\_(self, name, roll, age):

self.name = name

self.roll = roll

self.age = age

def show\_details(self):

print("Name:", self.name)

print("Roll Number:", self.roll)

print("Age:", self.age)

62. [Basic OOP] Create a class Rectangle with area() and perimeter().

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

def perimeter(self):

return 2 \* (self.length + self.width)

63. [Basic OOP] Create a class BankAccount with deposit and withdrawal methods.

class BankAccount:

def \_\_init\_\_(self, owner, balance=0):

self.owner = owner

self.balance = balance

def deposit(self, amount):

self.balance += amount

print("Amount Deposited:", amount)

def withdraw(self, amount):

if amount <= self.balance:

self.balance -= amount

print("Amount Withdrawn:", amount)

else:

print("Insufficient Balance")

def show\_balance(self):

print("Current Balance:", self.balance)

64. [Basic OOP] Create a class Car with attributes and display methods.

class Car:

def \_\_init\_\_(self, brand, model, year):

self.brand = brand

self.model = model

self.year = year

def show\_info(self):

print("Brand:", self.brand)

print("Model:", self.model)

print("Year:", self.year)

65. [Basic OOP] Create a class Employee with salary increment function.

class Employee:

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

def increment(self, amount):

self.salary += amount

def show\_salary(self):

print("Updated Salary:", self.salary)

66. [Basic OOP] Create a class Book with title, author, and display methods.

class Book:

def \_\_init\_\_(self, title, author, pages):

self.title = title

self.author = author

self.pages = pages

def show\_book(self):

print("Title:", self.title)

print("Author:", self.author)

print("Pages:", self.pages)

67. [Basic OOP] Create a class Calculator with add, subtract, multiply, divide.

class Calculator:

def add(self, a, b):

return a + b

def subtract(self, a, b):

return a - b

def multiply(self, a, b):

return a \* b

def divide(self, a, b):

if b == 0:

return "Cannot divide by zero"

return a / b

68. [Basic OOP] Create a class Complex to add two complex numbers.

class Complex:

def \_\_init\_\_(self, real, imag):

self.real = real

self.imag = imag

def add(self, other):

r = self.real + other.real

i = self.imag + other.imag

return Complex(r, i)

def show(self):

print(f"{self.real} + {self.imag}i")

69. [Basic OOP] Create a class Queue with enqueue and dequeue methods.

class Queue:

def \_\_init\_\_(self):

self.items = []

def enqueue(self, value):

self.items.append(value)

def dequeue(self):

if len(self.items) == 0:

return "Queue is empty"

return self.items.pop(0)

def show(self):

print("Queue:", self.items)

70. [Basic OOP] Create a class to store and display marks of students.

class Marks:

def \_\_init\_\_(self, name, marks\_list):

self.name = name

self.marks = marks\_list

def show\_details(self):

print("Name:", self.name)

print("Marks:", self.marks)

print("Total:", sum(self.marks))

print("Average:", sum(self.marks) / len(self.marks))

### *TOPIC: FILE HANDLING*

SOLUTIONS:

71. [File Handling] Write a program to read a file and print its content.

filename = input("Enter file name: ")

with open(filename, "r") as f:

content = f.read()

print(content)

72. [File Handling] Write a program to count lines in a file.

filename = input("Enter file name: ")

with open(filename, "r") as f:

lines = f.readlines()

print("Total lines:", len(lines))

73. [File Handling] Write a program to write user input to a text file.

filename = input("Enter file name: ")

data = input("Enter text to write: ")

with open(filename, "w") as f:

f.write(data)

print("Data written successfully.")

74. [File Handling] Write a program to copy one file into another.

source = input("Enter source file: ")

target = input("Enter destination file: ")

with open(source, "r") as f1, open(target, "w") as f2:

for line in f1:

f2.write(line)

print("File copied successfully.")

75. [File Handling] Write a program to count vowels in a file.

filename = input("Enter file name: ")

vowels = "aeiouAEIOU"

count = 0

with open(filename, "r") as f:

for ch in f.read():

if ch in vowels:

count += 1

print("Total vowels:", count)

76. [File Handling] Write a program to find the longest line in a file.

filename = input("Enter file name: ")

with open(filename, "r") as f:

lines = f.readlines()

longest = ""

for line in lines:

if len(line) > len(longest):

longest = line

print("Longest line:")

print(longest)

77. [File Handling] Write a program to append data to an existing file.

filename = input("Enter file name: ")

data = input("Enter text to append: ")

with open(filename, "a") as f:

f.write("\n" + data)

print("Data appended successfully.")

78. [File Handling] Write a program to search a word in a text file.

filename = input("Enter file name: ")

word = input("Enter word to search: ")

found = False

with open(filename, "r") as f:

for line in f:

if word in line:

found = True

break

if found:

print("Word found in file.")

else:

print("Word not found.")

79. [File Handling] Write a program to remove blank lines from a file.

source = input("Enter source file: ")

target = input("Enter destination file: ")

with open(source, "r") as f1, open(target, "w") as f2:

for line in f1:

if line.strip() != "":

f2.write(line)

print("Blank lines removed.")

80. [File Handling] Write a program to check if a file exists.

import os

filename = input("Enter file name: ")

if os.path.isfile(filename):

print("File exists.")

else:

print("File does not exist.")

### *TOPIC: EXCEPTION HANDLING*

SOLUTIONS:

81. [Exception Handling] Write a program to handle division by zero.

try:

a = int(input("Enter numerator: "))

b = int(input("Enter denominator: "))

print("Result:", a / b)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

82. [Exception Handling] Write a program to handle invalid value input.

try:

n = int(input("Enter a number: "))

print("You entered:", n)

except ValueError:

print("Error: Please enter a valid integer.")

83. [Exception Handling] Write a program to handle file not found errors.

filename = input("Enter file name: ")

try:

with open(filename, "r") as f:

print(f.read())

except FileNotFoundError:

print("Error: File does not exist.")

84. [Exception Handling] Write a program that raises and handles a custom exception.

class AgeError(Exception):

pass

try:

age = int(input("Enter your age: "))

if age < 0:

raise AgeError("Age cannot be negative.")

print("Age accepted:", age)

except AgeError as e:

print("Custom Error:", e)

85. [Exception Handling] Write a program with multiple exception blocks.

try:

a = int(input("Enter number: "))

b = int(input("Enter divisor: "))

print(a / b)

except ValueError:

print("Error: Enter numbers only.")

except ZeroDivisionError:

print("Error: Divisor cannot be zero.")

86. [Exception Handling] Write a program that handles type conversion errors.

try:

s = input("Enter a number: ")

n = int(s)

print("Converted value:", n)

except ValueError:

print("Error: Cannot convert to integer.")

87. [Exception Handling] Write a program to demonstrate try-else blocks.

try:

n = int(input("Enter a number: "))

except ValueError:

print("Invalid number.")

else:

print("You entered:", n)

88. [Exception Handling] Write a program to demonstrate finally block.

try:

n = int(input("Enter a number: "))

print("Square:", n \* n)

except ValueError:

print("Invalid input.")

finally:

print("Execution completed.")

89. [Exception Handling] Write a safe input function using exceptions.

def safe\_input():

while True:

try:

n = int(input("Enter an integer: "))

return n

except ValueError:

print("Invalid input. Try again.")

num = safe\_input()

print("You entered:", num)

90. [Exception Handling] Write a program to catch index errors in lists.

items = [10, 20, 30, 40]

try:

ind = int(input("Enter index: "))

print("Value:", items[ind])

except IndexError:

print("Error: Index out of range.")

### *TOPIC: NUMBER & SERIES PROBLEMS*

SOLUTIONS:

91. [Number Problems] Check Armstrong number.

num = int(input("Enter number: "))

s = str(num)

total = 0

for ch in s:

total += int(ch) \*\* len(s)

if total == num:

print("Armstrong Number")

else:

print("Not Armstrong")

92. [Number Problems] Check Krishnamurthy number.

num = int(input("Enter number: "))

def fact(n):

f = 1

for i in range(1, n+1):

f \*= i

return f

temp = num

total = 0

while temp > 0:

total += fact(temp % 10)

temp //= 10

if total == num:

print("Krishnamurthy Number")

else:

print("Not Krishnamurthy")

93. [Number Problems] Check Perfect number.

num = int(input("Enter number: "))

total = 0

for i in range(1, num):

if num % i == 0:

total += i

if total == num:

print("Perfect Number")

else:

print("Not Perfect")

94. [Number Problems] Check Abundant numbers.

num = int(input("Enter number: "))

total = 0

for i in range(1, num):

if num % i == 0:

total += i

if total > num:

print("Abundant Number")

else:

print("Not Abundant")

95. [Number Problems] Check Harshad number.

num = int(input("Enter number: "))

s = sum(int(ch) for ch in str(num))

if num % s == 0:

print("Harshad Number")

else:

print("Not Harshad")

96. [Number Problems] Find LCM of two numbers.

a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

m = max(a, b)

while True:

if m % a == 0 and m % b == 0:

print("LCM:", m)

break

m += 1

97. [Number Problems] Find GCD of two numbers.

a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

while b != 0:

a, b = b, a % b

print("GCD:", a)

98. [Number Problems] Check Automorphic numbers.

num = int(input("Enter number: "))

sq = num \* num

if str(sq).endswith(str(num)):

print("Automorphic Number")

else:

print("Not Automorphic")

99. [Number Problems] Check Strong numbers.

num = int(input("Enter number: "))

def fact(n):

f = 1

for i in range(1, n+1):

f \*= i

return f

total = sum(fact(int(ch)) for ch in str(num))

if total == num:

print("Strong Number")

else:

print("Not Strong")

100. [Number Problems] Check Neon number.

num = int(input("Enter number: "))

sq = num \* num

total = sum(int(ch) for ch in str(sq))

if total == num:

print("Neon Number")

else:

print("Not Neon")

101. [Series Problems] Print arithmetic progression.

a = int(input("Enter first term: "))

d = int(input("Enter common difference: "))

n = int(input("How many terms? "))

for i in range(n):

print(a + i\*d, end=" ")

102. [Series Problems] Print geometric progression.

a = int(input("Enter first term: "))

r = int(input("Enter ratio: "))

n = int(input("How many terms? "))

for i in range(n):

print(a \* (r \*\* i), end=" ")

103. [Series Problems] Generate series: 1, 4, 9, 16… (squares).

n = int(input("How many terms? "))

for i in range(1, n+1):

print(i\*i, end=" ")

104. [Series Problems] Generate series: 1, 8, 27… (cubes).

n = int(input("How many terms? "))

for i in range(1, n+1):

print(i\*i\*i, end=" ")

105. [Series Problems] Generate series: 1, 2, 4, 7, 11…

n = int(input("How many terms? "))

a = 1

inc = 1

for \_ in range(n):

print(a, end=" ")

a += inc

inc += 1

106. [Series Problems] Generate Fibonacci series.

n = int(input("How many terms? "))

a, b = 0, 1

for \_ in range(n):

print(a, end=" ")

a, b = b, a + b

107. [Series Problems] Generate prime number series.

n = int(input("How many primes? "))

count = 0

num = 2

while count < n:

flag = True

for i in range(2, num):

if num % i == 0:

flag = False

break

if flag:

print(num, end=" ")

count += 1

num += 1

108. [Series Problems] Generate harmonic series.

n = int(input("How many terms? "))

for i in range(1, n+1):

print(f"1/{i}", end=" ")

109. [Series Problems] Generate odd number series.

n = int(input("How many terms? "))

for i in range(n):

print(2\*i + 1, end=" ")

110. [Series Problems] Generate even number series.

n = int(input("How many terms? "))

for i in range(n):

print(2\*i, end=" ")

### *TOPIC: PATTERN PROBLEMS*

SOLUTIONS:

111. [Patterns] Print a square pattern of \*.

\* \* \* \*

\* \* \* \*

\* \* \* \*

\* \* \* \*

n = int(input("Enter size: "))

for i in range(n):

for j in range(n):

print("\*", end=" ")

print()

112. [Patterns] Print a right triangle of numbers.

1

12

123

1234

n = int(input("Enter rows: "))

for i in range(1, n+1):

for j in range(1, i+1):

print(j, end="")

print()

113. [Patterns] Print Floyd's triangle.

1

2 3

4 5 6

7 8 9 10

n = int(input("Enter rows: "))

num = 1

for i in range(1, n+1):

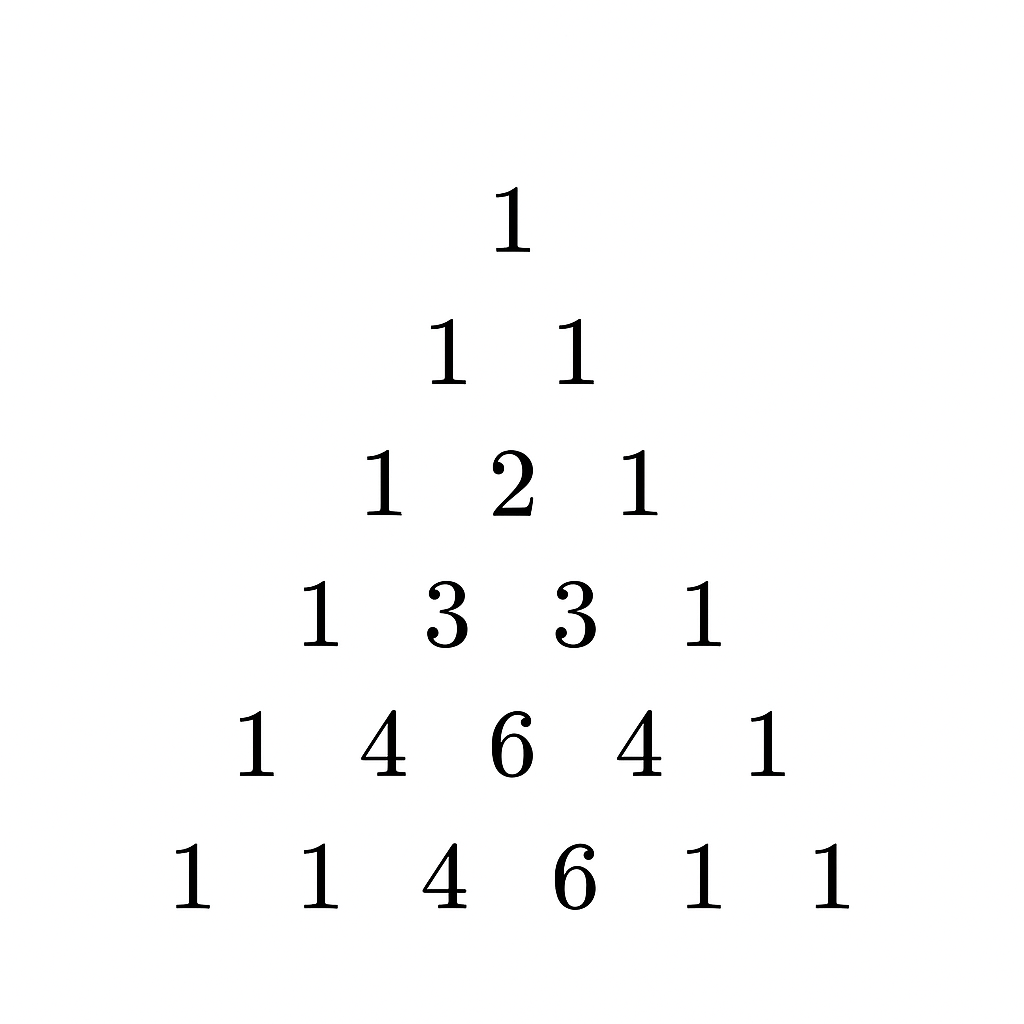
for j in range(i):

print(num, end=" ")

num += 1

print()

114. [Patterns] Print Pascal's triangle.



n = int(input("Enter rows: "))

for i in range(n):

print(" " \* (n - i), end="")

val = 1

for j in range(i + 1):

print(val, end=" ")

val = val \* (i - j) // (j + 1)

print()

115. [Patterns] Print inverted pyramid.

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

n = int(input("Enter rows: "))

for i in range(n):

print(" " \* i + "\*" \* (n - i))

116. [Patterns] Print number pyramid 1 22 333.

1

22

333

4444

n = int(input("Enter rows: "))

for i in range(1, n+1):

print(str(i) \* i)

117. [Patterns] Print alphabet triangle.

A

AB

ABC

ABCD

n = int(input("Enter rows: "))

for i in range(1, n+1):

for j in range(i):

print(chr(65 + j), end="")

print()

118. [Patterns] Print hollow squares.

\*\*\*\*\*

\* \*

\* \*

\*\*\*\*\*

n = int(input("Enter size: "))

for i in range(n):

for j in range(n):

if i == 0 or i == n-1 or j == 0 or j == n-1:

print("\*", end="")

else:

print(" ", end="")

print()

119. [Patterns] Print diamond pattern.

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\***

**\***

n = int(input("Enter rows: "))

for i in range(1, n+1):

print(" " \* (n - i) + "\*" \* (2\*i - 1))

for i in range(n-1, 0, -1):

print(" " \* (n - i) + "\*" \* (2\*i - 1))

120. [Patterns] Print butterfly pattern.

n = int(input("Enter size: "))

for i in range(1, n+1):

print("\*" \* i + " " \* (2\*(n-i)) + "\*" \* i)

for i in range(n, 0, -1):

print("\*" \* i + " " \* (2\*(n-i)) + "\*" \* i)

### *TOPIC: NUMPY*

SOLUTIONS:

121. [NumPy] Create a NumPy array and print its shape.

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print("Array:", arr)

print("Shape:", arr.shape)

122. [NumPy] Perform element-wise addition of two NumPy arrays.

import numpy as np

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

result = a + b

print("Result:", result)

123. [NumPy] Find mean, median, and variance of array.

import numpy as np

arr = np.array([10, 20, 30, 40])

print("Mean:", np.mean(arr))

print("Median:", np.median(arr))

print("Variance:", np.var(arr))

124. [NumPy] Reshape a NumPy array.

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6])

reshaped = arr.reshape(2, 3)

print(reshaped)

Output:

[[1 2 3]

[4 5 6]]

125. [NumPy] Slice a NumPy array. (You can slice as per your choice)

import numpy as np

arr = np.array([10, 20, 30, 40, 50, 60])

print(arr[1:4])

print(arr[:3])

print(arr[-3:])

126. [NumPy] Stack two arrays horizontally.

import numpy as np

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

h = np.hstack((a, b))

print(h)

Output:

[1 2 3 4 5 6]

127. [NumPy] Stack two arrays vertically.

import numpy as np

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

v = np.vstack((a, b))

print(v)

Output:

[[1 2 3]

[4 5 6]]

128. [NumPy] Generate an array of zeros.

import numpy as np

arr = np.zeros((3, 4))

print(arr)

Output:

[[0. 0. 0. 0.]

[0. 0. 0. 0.]

[0. 0. 0. 0.]]

129. [NumPy] Generate an array of random integers.

import numpy as np

arr = np.random.randint(1, 50, size=5)

print(arr)

Output:

[29 3 18 8 17]

130. [NumPy] Find unique elements in an array.

import numpy as np

arr = np.array([1, 2, 2, 3, 4, 4, 5])

unique\_vals = np.unique(arr)

print(unique\_vals)

Output:

[1 2 3 4 5]

### *TOPIC: MATPLOTLIB*

SOLUTIONS:

131. [Matplotlib] Plot a simple line graph.

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

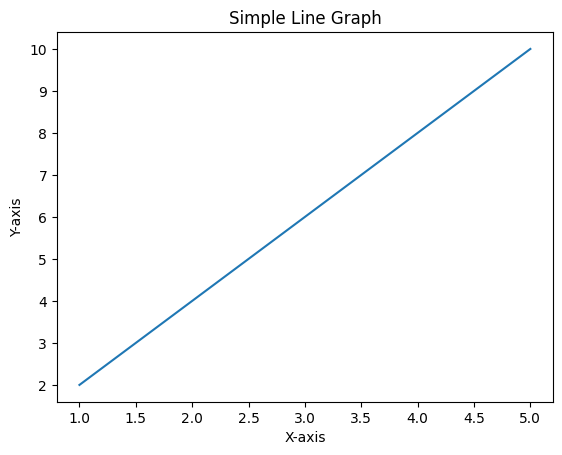
plt.plot(x, y)

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("Simple Line Graph")

plt.show()



132. [Matplotlib] Plot a bar chart of student marks.

import matplotlib.pyplot as plt

students = ["A", "B", "C", "D"]

marks = [75, 88, 92, 60]

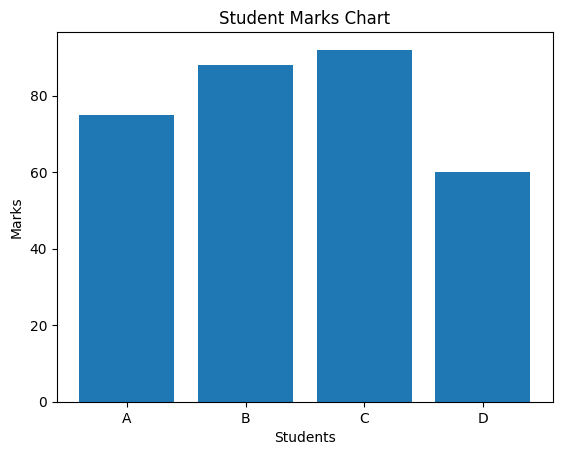
plt.bar(students, marks)

plt.xlabel("Students")

plt.ylabel("Marks")

plt.title("Student Marks Chart")

plt.show()



133. [Matplotlib] Plot a pie chart of expenses.

import matplotlib.pyplot as plt

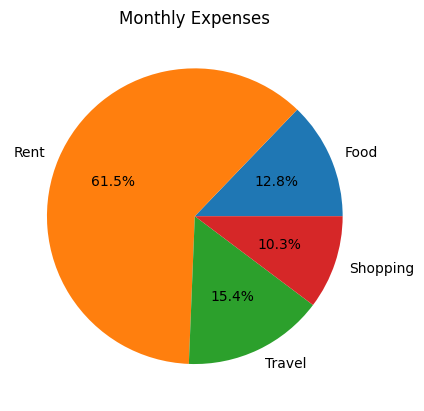
labels = ["Food", "Rent", "Travel", "Shopping"]

values = [2500, 12000, 3000, 2000]

plt.pie(values, labels=labels, autopct="%0.1f%%")

plt.title("Monthly Expenses")

plt.show()



134. [Matplotlib] Plot sine wave using NumPy + Matplotlib.

import numpy as np

import matplotlib.pyplot as plt

x = np.linspace(0, 20, 200)

y = np.sin(x)

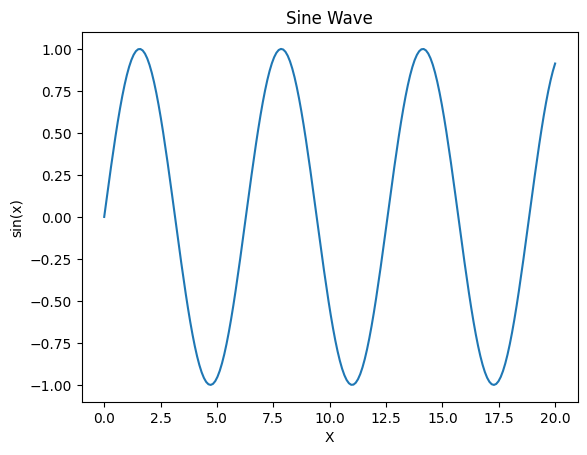
plt.plot(x, y)

plt.title("Sine Wave")

plt.xlabel("X")

plt.ylabel("sin(x)")

plt.show()



135. [Matplotlib] Plot scatter plot between two variables.

import matplotlib.pyplot as plt

x = [5, 7, 8, 9, 11]

y = [12, 14, 16, 18, 20]

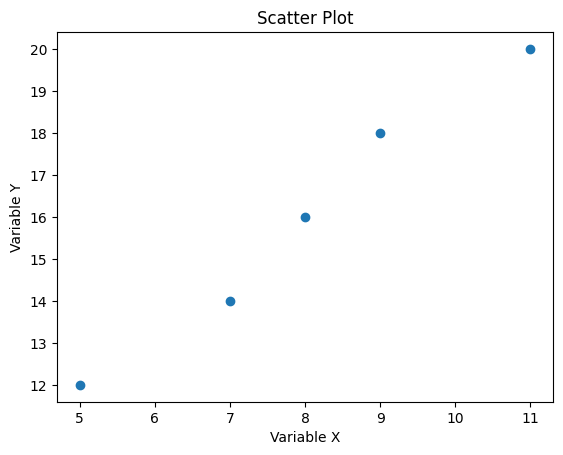
plt.scatter(x, y)

plt.xlabel("Variable X")

plt.ylabel("Variable Y")

plt.title("Scatter Plot")

plt.show()



136. [Matplotlib] Plot histogram of heights.

import matplotlib.pyplot as plt

heights = [150, 155, 160, 162, 168, 170, 172, 175, 178, 180]

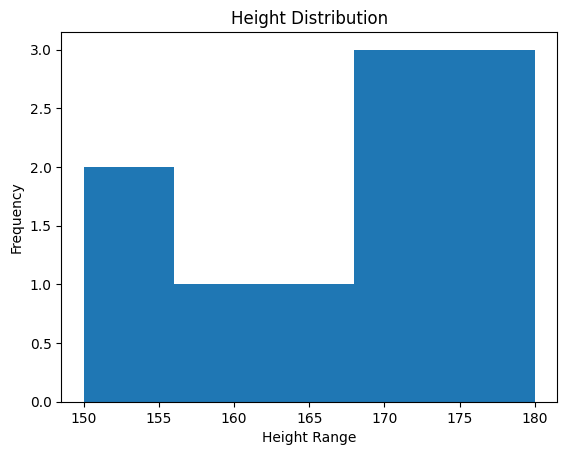
plt.hist(heights, bins=5)

plt.xlabel("Height Range")

plt.ylabel("Frequency")

plt.title("Height Distribution")

plt.show()



137. [Matplotlib] Add labels and title to the graph.

import matplotlib.pyplot as plt

x = [1, 2, 3]

y = [2, 5, 7]

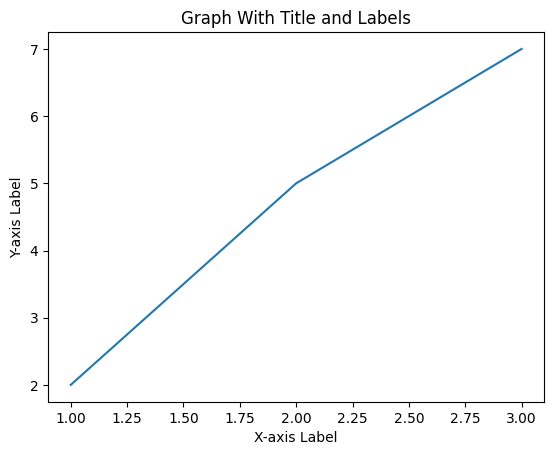
plt.plot(x, y)

plt.xlabel("X-axis Label")

plt.ylabel("Y-axis Label")

plt.title("Graph With Title and Labels")

plt.show()



138. [Matplotlib] Customize line style and width.

import matplotlib.pyplot as plt

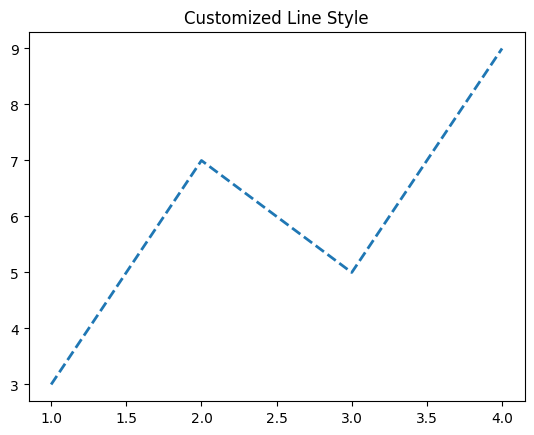
x = [1, 2, 3, 4]

y = [3, 7, 5, 9]

plt.plot(x, y, linestyle="--", linewidth=2)

plt.title("Customized Line Style")

plt.show()



139. [Matplotlib] Create a subplot with 2 graphs.

import matplotlib.pyplot as plt

x = [1, 2, 3, 4]

plt.subplot(1, 2, 1)

plt.plot(x, [i \* 2 for i in x])

plt.title("Graph 1")

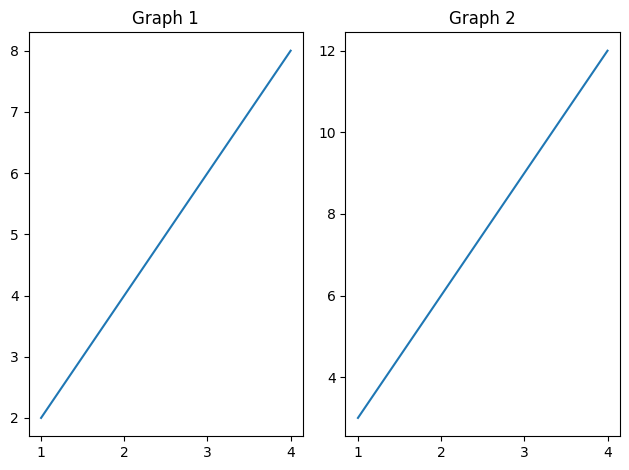
plt.subplot(1, 2, 2)

plt.plot(x, [i \* 3 for i in x])

plt.title("Graph 2")

plt.tight\_layout()

plt.show()



140. [Matplotlib] Save a plot as an image.

import matplotlib.pyplot as plt

x = [1, 2, 3, 4]

y = [2, 4, 6, 8]

plt.plot(x, y)

plt.title("Saved Plot")

plt.savefig("my\_plot.png")

plt.show()



### *TOPIC: PANDAS*

SOLUTIONS:

141. [Pandas] Create a DataFrame from a dictionary.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

print(df)

Output:

Name Marks

0 Aman 85

1 Riya 92

2 Karan 78

142. [Pandas] Read a CSV file using pandas.

import pandas as pd

df = pd.read\_csv("data.csv")

print(df)

143. [Pandas] Display first and last 5 rows of a DataFrame.

import pandas as pd

df = pd.read\_csv("data.csv")

print(df.head())

print(df.tail())

144. [Pandas] Sort DataFrame by a column.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

sorted\_df = df.sort\_values("Marks")

print(sorted\_df)

Output:

Name Marks

2 Karan 78

0 Aman 85

1 Riya 92

145. [Pandas] Select specific rows and columns.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

result = df.loc[0:1, ["Name", "Marks"]]

print(result)

Output:

Name Marks

0 Aman 85

1 Riya 92

146. [Pandas] Filter DataFrame using conditions.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

high\_marks = df[df["Marks"] > 80]

print(high\_marks)

Output:

Name Marks

0 Aman 85

1 Riya 92

147. [Pandas] Add a new column to DataFrame.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

df["Grade"] = ["A", "A", "B", "C"][:len(df)] # simple example

print(df)

Output:

Name Marks Grade

0 Aman 85 A

1 Riya 92 A

2 Karan 78 B

148. [Pandas] Drop a column from DataFrame.

import pandas as pd

data = {

"Name": ["Aman", "Riya", "Karan"],

"Marks": [85, 92, 78]

}

df = pd.DataFrame(data)

df = df.drop("Marks", axis=1)

print(df)

Output:

Name

0 Aman

1 Riya

2 Karan

149. [Pandas] Handle missing values.(***A part of data preprocessing for a machine learning model***)

import pandas as pd

df = pd.read\_csv("data.csv")

clean\_df = df.fillna(0)

print(clean\_df)

150. [Pandas] Group data and calculate mean.

import pandas as pd

df = pd.read\_csv("data.csv")

grouped = df.groupby("Class")["Marks"].mean()

print(grouped)

BONUS QUESTIONS:

TOPICS:

***LISTS:***

1. Find the largest and smallest number in a list

nums = [12, 4, 7, 19, 3]

print("Largest:", max(nums))

print("Smallest:", min(nums))

2.Remove duplicates from a list

nums = [1, 2, 2, 3, 4, 4, 5]

unique\_list = list(set(nums))

print(unique\_list)

3. Count occurrences of an element

nums = [1, 2, 3, 2, 2, 5]

print(nums.count(2))

4. Reverse a list without using reverse()

nums = [10, 20, 30, 40]

rev = nums[::-1]

print(rev)

5. Find the sum of all elements

nums = [5, 10, 15]

print(sum(nums))

***TUPLES:***

6. Check if an element exists in a tuple

t = (4, 8, 12, 16)

print(8 in t)

### 7. Convert tuple to list and back to tuple

t = (1, 2, 3)

lst = list(t)

lst.append(4)

t = tuple(lst)

print(t)

8. Find index of an element in a tuple

t = ("apple", "banana", "orange")

print(t.index("banana"))

9. Count occurrences in a tuple

t = (1, 2, 2, 3, 2)

print(t.count(2))

### 10. Concatenate two tuples

a = (1, 2)

b = (3, 4)

c = a + b

print(c)

***SETS:***

11. Find union and intersection of two sets

a = {1, 2, 3}

b = {3, 4, 5}

print("Union:", a | b)

print("Intersection:", a & b)

12. Check if one set is subset of another

a = {1, 2}

b = {1, 2, 3, 4}

print(a.issubset(b))

13. Add and remove elements from a set

s = {10, 20, 30}

s.add(40)

s.remove(20)

print(s)

14. Find difference between two sets

a = {1, 2, 3}

b = {2, 3, 4}

print(a - b)

15. Remove duplicates from a list using set

nums = [1, 2, 3, 3, 4]

unique\_nums = list(set(nums))

print(unique\_nums)

***DICTIONARIES:***

### 16. Count frequency of each element in a list

nums = [1, 2, 2, 3, 3, 3]

freq = {}

for n in nums:

freq[n] = freq.get(n, 0) + 1

print(freq)

17. Print keys and values separately

d = {"name": "Amit", "age": 22, "city": "Delhi"}

print(d.keys())

print(d.values())

18. Add and update dictionary items

d = {"name": "Amit"}

d["age"] = 22

d["name"] = "Sumit"

print(d)

19. Merge two dictionaries

a = {"x": 1, "y": 2}

b = {"z": 3}

c = {\*\*a, \*\*b}

print(c)

20. Create dictionary from two lists

keys = ["a", "b", "c"]

values = [1, 2, 3]

d = dict(zip(keys, values))

print(d)

Thank you for reading this book and being part of this learning journey. I hope the programs inside helped you practice Python with confidence. If this handbook supported your studies or coding progress, a short review would mean a lot and help others discover the book.