DR. BR Ambedkar National Institute of Technology, Jalandhar

Center of Excellence AI (Session 2025-27)

Programming with Python Laboratory : AI-523

Submitted to:
Dr. Diksha Kumari
Dr. Ruchika Arora

Submitted by:
Rohit Kumar
Roll No: 25901334
M.Tech AI (2025-27)

CONTENTS:

github:
https://github.com/Ohi-Me/NITJ_AI/tree/main/PPL_LAB

🔷 EXPERIMENT 1

Installation and Basic Programs Date: 28/08/2025 Objective: To understand variables, data types, and basic operations in Python

a) Display data of different types

In [1]:
```
a = 10
b = 3.14
c = "Python"
d = True

print(a, type(a))
print(b, type(b))
print(c, type(c))
print(d, type(d))
```

```
10 <class 'int'>
3.14 <class 'float'>
Python <class 'str'>
True <class 'bool'>
```

Code Explanation: Variables are created to store integer, float, string, and boolean values. The type() function is used to identify the data type of each variable. The print() statement displays the value along with its data type.

b) Write a program to show the application of the += operator on strings.

In [2]:
```
s = "Hello"
s += " World"
print(s)
```

```
Hello World
```

Explanation: The += operator appends one string to another and updates the original string.

c) Write a program that performs addition and multiplication on string variables.

In [3]:
```
s = "Hi"
print(s + " Python")
print(s * 3)
```

```
Hi Python
HiHiHi
```

Explanation:

- joins two strings.
- repeats the string specified number of times.

🔷 EXPERIMENT 2

Operators Based Programs Date: 04/09/2025 Objective: To apply arithmetic operators in real-life problems

a) Write a program to calculate area of a triangle using Heron's Formula.

In [4]:
```
import math  # Importing math module
a, b, c = 3, 4, 5
s = (a + b + c) / 2
# Area calculation
area = math.sqrt(s*(s-a)*(s-b)*(s-c))
print("Area of triangle:", area)
```

```
Area of triangle: 6.0
```

In [ ]:
```
Explanation:
Semi-perimeter is calculated first.
Heron's formula is applied using math.sqrt().
```

b) Write a program to calculate the total amount of money in the piggy bank, given the coins of Rs10,5,2,1.

In [6]:
```
# Taking number of coins
rs10 = int(input("enter no of 10 rs coin"))
rs5 = int(input("enter no of 5 rs coin"))
rs2 = int(input("enter no of 2 rs coin"))
rs1 = int(input("enter no of 1 rs coin"))
# Calculating total amount
total = rs10*10 + rs5*5 + rs2*2 + rs1
print("Total Amount:", total)
```

```
Total Amount: 105
```

Explanation: Each coin count is multiplied by its value. All amounts are added to get total money.

🔷 EXPERIMENT 3 Objective:To understand string manipulation and number system conversions in Python using built-in functions and type conversion methods Date: 11/09/2025

a) Write a Python program that asks the user to enter an integer in decimal (base 10) and then prints its equivalent in binary, octal, and hexadecimal using built-in functions.

In [7]:
```
# Taking decimal input
n = int(input("Enter number: "))
# Conversions
print("Binary:", bin(n))
print("Octal:", oct(n))
print("Hexadecimal:", hex(n))
```

```
Binary: 0b11001
Octal: 0o31
Hexadecimal: 0x19
```

Explanation: Built-in functions convert decimal to other number systems.

b) Write a Python program that reads a string representing a floating-point number (for example, "25.67") from the user, converts it to a float, adds 10.5 to it, and prints the result.

In [8]:
```
# Taking float as string
s = input("Enter float value: ")
# Converting and adding
num = float(s)
print(num + 10.5)
```

```
67.37
```

Explanation: String is converted to float using float() and arithmetic is performed.

🔷 EXPERIMENT 4: Operator Precedence, Conditional Statements and Loops

Objective: To study operator precedence in Python and implement conditional statements and looping constructs to solve various problems. Date: 18/09/2025

a) Write a program to solve the expression: x = 10 + 2 * 3 ** 2 - 8 / 4 and show each expression calculation and output.

In [9]:
```
# Step 1: Calculate power
step1 = 3 ** 2
print("Step 1: 3 ** 2 =", step1)
# Step 2: Multiply
step2 = 2 * step1
print("Step 2: 2 * 3 ** 2 =", step2)
# Step 3: Divide
step3 = 8 / 4
print("Step 3: 8 / 4 =", step3)
# Step 4: Final calculation
x = 10 + step2 - step3
print("Step 4: 10 + 2 * 3 ** 2 - 8 / 4 =", x)
```

```
Step 1: 3 ** 2 = 9
Step 2: 2 * 3 ** 2 = 18
Step 3: 8 / 4 = 2.0
Step 4: 10 + 2 * 3 ** 2 - 8 / 4 = 26.0
```

Explanation: This program evaluates the expression step by step by first calculating the power, then performing multiplication and division according to operator precedence, and finally adding and subtracting the results to obtain the value x = 26.0

b) Write a Python program that creates a dictionary to store three students' names as keys and their marks as values, then:
i) Prints the dictionary
ii) Prints the marks of a student whose name is entered by the user

In [11]:
```
# Dictionary creation
students = {"Amit":85, "Riya":90, "Neha":88}
print(students)
```

```
# Searching marks
name = input("Enter name: ")
print(students.get(name, "Student not found"))
```

```
{'Amit': 85, 'Riya': 90, 'Neha': 88}
85
```

Explanation:Dictionary stores names and marks. get() safely fetches value.

c) Write a program to print the reverse of a number.

In [12]:
```
n = int(input("enter a number"))
rev = 0
# Loop to reverse digits
while n > 0:
    rev = rev*10 + n%10
    n //= 10

print(rev)
```

```
6879876
```

Explanation: Digits are extracted using modulo and reversed using loop.

d) Write the program to print multiplication table of n, where n is entered by the user.

In [13]:
```
n = int(input("Enter a number"))

# Printing table
for i in range(1,11):
    print(n, "x", i, "=", n*i)
```

```
25 x 1 = 25
25 x 2 = 50
25 x 3 = 75
25 x 4 = 100
25 x 5 = 125
25 x 6 = 150
25 x 7 = 175
25 x 8 = 200
25 x 9 = 225
25 x 10 = 250
```

Explanation: This program takes a number as input and uses a for loop to print its multiplication table from 1 to 10.

🔷 EXPERIMENT 5: Nested Loops, String Multiplication and Keyword Arguments

Objective: To understand the use of nested loops for pattern generation, string multiplication, and keyword arguments in Python functions.

Date: 25/09/2025

a) Write a program to print the following pattern:
1 12 123 1234 12345

In [15]:
```
for i in range(1, 6):
    # print spaces
```

```
    for s in range(5 - i):
        print(" ", end="")

    # print numbers
    for j in range(1, i + 1):
        print(j, end="")

    # move to next line
    print()
```

```
    1
   12
  123
 1234
12345
```

Explanation: This program uses nested loops to first print spaces and then numbers to form a right-aligned number pattern.

b) Write the program to print multiplication table of n, where n is entered by the user

In [16]:
```
n = int(input("Enter a number"))

# Printing table
for i in range(1,11):
    print(n, "x", i, "=", n*i)
```

```
45 x 1 = 45
45 x 2 = 90
45 x 3 = 135
45 x 4 = 180
45 x 5 = 225
45 x 6 = 270
45 x 7 = 315
45 x 8 = 360
45 x 9 = 405
45 x 10 = 450
```

Explanation: This program takes a number as input and uses a for loop to print its multiplication table from 1 to 10.

c) Write a Python function student info that takes three keyword arguments: name, age, and course. Call the function using keyword arguments in any order and print the information in a readable format.

In [17]:
```
def student_info(name, age, course):
    print(name, age, course)

# Calling with keywords
student_info(course="AI", name="Ravi", age=20)
```

```
Ravi 20 AI
```

Explanation: Keyword arguments allow passing values in any order.

d) Write a program that prints absolute value, square root and cube of a number.

In [19]:
```
import math

n = int(input("enter a no"))

print(abs(n))
print(math.sqrt(n))
print(n**3)
```

```
55
7.416198487095663
166375
```

Explanation: Built-in and math functions perform calculations.

🔷 EXPERIMENT 6: Program Based on Functions

Objective: To understand different types of functions in Python including lambda functions, recursive functions, and user-defined functions. Date: 26/09/2025

a) Write a Python program that uses a lambda function to: i)Take a number as input. ii)Use a lambda function to square the number. iii.) Print the squared value.

In [22]:
```
# Lambda function
n=int(input("enter a no"))
square = lambda x: x*x
print(square(n))
```

```
16
```

Explanation: This code uses a lambda (anonymous) function to define a one-line function that takes a number as input, multiplies it by itself to find the square, and then prints the square of 5, which is 25.

b) Write a recursive function count digits(n) that returns the number of digits in a positive integer n.

In [23]:
```
def count_digits(n):
    if n == 0:
        return 0
    return 1 + count_digits(n//10)

print(count_digits(1234))
```

```
4
```

Explanation: This program uses recursion to count the number of digits in a number by repeatedly removing the last digit using integer division until the number becomes 0, giving the output 4.

c) Write a Python program to: Take n integers into a list. Print the list in original order. Print the list in reverse order (without using [::-1] or reverse()).

In [24]:
```
n = int(input("Enter number of elements: "))
lst = []

# Taking input into list
```

```
for i in range(n):
    num = int(input("Enter element: "))
    lst.append(num)
# Printing original list
print("Original list:", lst)

print("Reversed list:", end=" ")
for i in range(len(lst) - 1, -1, -1):
    print(lst[i], end=" ")
```

```
Original list: [45, 88, 23, 776, 67]
Reversed list: 67 776 23 88 45
```

Explanation: This program works by first reading how many integers the user wants to enter, storing those integers one by one in a list, printing the list exactly as it was entered, and then traversing the list from the last index to the first using a loop to display the elements in reverse order without using any built-in reversing functions.

🔷 EXPERIMENT 7: Modules and Packages

Objective: To understand the concept of modules and packages in Python and learn how to import and use built-in modules such as random and matplotlib. Date: 30/09/2025

a) Write a Python program that: Imports the random module with alias name rnd. Prints 5 random integers between 1 and 100 using the alias.

In [25]:
```
import random as rnd

for i in range(5):
    print(rnd.randint(1,100))
```

```
96
41
5
32
27
```

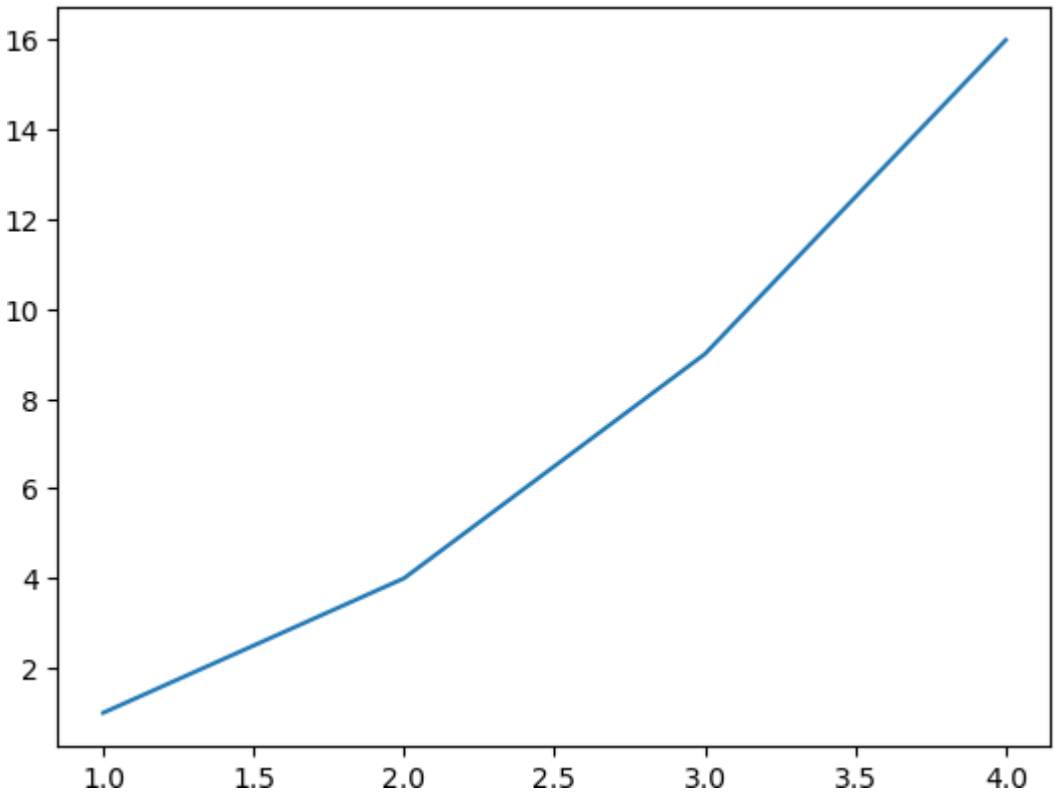
Explanation : This code imports Python's random module with the alias rnd and then runs a loop five times to generate and print a different random integer between 1 and 100 (inclusive) on each iteration.

b) Write a Python program that: i.) Imports pyplot from the matplotlib package as plt. ii.)Plots a simple line graph for x = [1, 2, 3, 4] and y = [1, 4, 9, 16]

In [26]:
```
import matplotlib.pyplot as plt

x = [1,2,3,4]
y = [1,4,9,16]

plt.plot(x,y)
plt.show()
```



Explanation: This program uses matplotlib to plot a line graph showing the relationship between x values and their corresponding y values, and then displays the graph on the screen.

🔷 EXPERIMENT 8: String and Its Operations

Objective: To perform various string operations in Python such as vowel removal, palindrome checking, and reversing the order of words in a sentence. Date: 16/10/2025

a) Write a program that takes a string as input and prints a new string where all vowels are removed, but the order of the remaining characters is preserved.

In [27]:
```
s = input("Enter a string: ")
result = ""

for ch in s:
    if ch not in "aeiouAEIOU":
        result = result + ch

print("String without vowels:", result)
```

```
String without vowels: pythn s fn
```

Explanation: This program checks each character of the input string and builds a new string by adding only those characters that are not vowels.

b) Given a string, write a program to check whether it is a palindrome or not, i gnoring case and all non-alphanumeric characters.

In [28]:
```
s = input("Enter a string: ")

clean = ""
for ch in s:
    if ch.isalnum():        # keep only letters and numbers
        clean += ch.lower()

if clean == clean[::-1]:
    print("Palindrome")
else:
    print("Not a palindrome")
```

```
Not a palindrome
```

Explanation: This program removes all non-alphanumeric characters, converts the string to lowercase, and then checks if it reads the same forwards and backwards.

c) Write a program that takes a sentence as input and prints the words in reverse order, while keeping the characters inside each word in the same order.

In [29]:
```
sentence = input("Enter a sentence: ")

words = sentence.split()    # split sentence into words
rev_words = []

for i in range(len(words) - 1, -1, -1):
    rev_words.append(words[i])

result = " ".join(rev_words)
print("Reversed sentence:", result)
```

```
Reversed sentence: fun is python
```

Explanation: This program splits the sentence into words, reverses the order of the words using a loop, and joins them back to form the final sentence

🔷 EXPERIMENT 9: File Handling

Objective: To understand file handling operations in Python including reading text files, counting characters, and copying binary data from one file to another. Date: 30/10/2025

a)Write a program that counts the number of tabs, spaces and newline characters in a file.

In [32]:
```
f = open("file.txt","r")
text = f.read()

print(text.count(" "),"whitespaces")
print(text.count("\n"),"newline characters")
print(text.count("\t"),"tabs")
```

```
5 whitespaces
3 newline characters
3 tabs
```

Explanation: This program reads the file character by character and counts spaces, tabs, and newline characters using simple conditional checks.

b) Write a program that copies first 10 bytes of a binary file into another.

In [34]:
```
with open("source.bin", "wb") as f:
    f.write(b"ABCDEFGHIJKLMNOPQRSTUVWXYZ")
# Copy first 10 bytes
with open("source.bin", "rb") as src:
    data = src.read(10)

with open("dest.bin", "wb") as dest:
    dest.write(data)

# Read and print content of new file
with open("dest.bin", "rb") as dest:
    content = dest.read()
    print("Content of new file:", content)
```

```
Content of new file: b'ABCDEFGHIJ'
```

Explanation: This program reads the first 10 bytes from a binary file, writes them into another file, and then reads and prints the content of the new file.

🔷 EXPERIMENT 10: Data Structures – Lists, Tuples and Sets

Objective:To study and implement Python data structures such as lists, tuples, and sets and perform operations like removing duplicates, searching elements, and set operations. Date: 06/11/2025

a) Write a Python program to: i)Take n integers from the user and store them in a list. ii)Create a new list that contains the same elements but without duplicates. iii)Print both the original list and the new list.

In [36]:
```
n = int(input("Enter number of elements: "))
lst = []

# i) Take n integers from user
for i in range(n):
    num = int(input("Enter element: "))
    lst.append(num)

# ii) Create new list without duplicates
new_list = []
for x in lst:
    if x not in new_list:
        new_list.append(x)

# iii) Print both lists
print("Original list:", lst)
print("List without duplicates:", new_list)
```

```
Original list: [55, 77, 55, 77, 34]
List without duplicates: [55, 77, 34]
```

Explanation: This program stores user-entered integers in a list, creates a second list by adding only unique elements, and then prints both lists.

b) Write a Python program to: Create a tuple, e.g. nums = (10, 20, 30, 40, 50). Ask the user to enter a number. Print the index of that number in the tuple if it exists, otherwise print a message saying it is not present.

In [37]:
```
nums = (10, 20, 30, 40, 50)

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

if n in nums:
    print("Number found at index:", nums.index(n))
else:
    print("Number is not present in the tuple")
```

```
Number found at index: 3
```

Explanation: This program checks whether the user-entered number exists in the tuple and prints its index if found, otherwise displays a not-present message.

c) Write a program that reads two sequences of integers (allow duplicates) and prints: the set of elements common to both sequences, the set of elements that appear in exactly one of the two sequences (symmetric difference). Elements in each output set should be printed in sorted order without duplicates.

In [38]:
```
# Read first sequence
n1 = int(input("Enter number of elements in first sequence: "))
seq1 = []
for i in range(n1):
    seq1.append(int(input("Enter element: ")))

# Read second sequence
n2 = int(input("Enter number of elements in second sequence: "))
seq2 = []
for i in range(n2):
    seq2.append(int(input("Enter element: ")))

# Convert to sets to remove duplicates
set1 = set(seq1)
set2 = set(seq2)

# Common elements
common = sorted(set1 & set2)

# Elements in exactly one sequence (symmetric difference)
sym_diff = sorted(set1 ^ set2)

# Print results
print("Common elements:", common)
print("Symmetric difference:", sym_diff)
```

```
Common elements: []
Symmetric difference: [22, 23, 34, 55, 67, 75, 79, 88]
```

In [ ]:
```
Explanation: This program reads two integer sequences, converts them into
and then prints the sorted common elements and the sorted symmetric
difference
```

🔷EXPERIMENT 11: Programming Exercises with Classes and Objects
Objective: To understand object-oriented programming concepts in Python such as classes, objects, class variables, and relationships between different classes. Date: 13/11/2025

a) Write a program with class Employee that keeps a track of the number of employees in an organization and also stores their name, designation and salary details.

In [39]:
```
class Employee:
    count = 0    # class variable to track number of employees

    def __init__(self, name, designation, salary):
        self.name = name
        self.designation = designation
        self.salary = salary
        Employee.count += 1

    def display(self):
        print("Name:", self.name)
        print("Designation:", self.designation)
        print("Salary:", self.salary)
        print()

# Creating employee objects
e1 = Employee("Amit", "Manager", 50000)
e2 = Employee("Riya", "Developer", 40000)
e3 = Employee("Neha", "Tester", 35000)

# Display details
e1.display()
e2.display()
e3.display()

# Display total employees
print("Total number of employees:", Employee.count)
```

```
Name: Amit
Designation: Manager
Salary: 50000

Name: Riya
Designation: Developer
Salary: 40000

Name: Neha
Designation: Tester
Salary: 35000

Total number of employees: 3
```

Explanation: This program defines an Employee class that stores employee details and uses a class variable to count and display the total number of employees created.

b) Write a program that has classes such as Student, Course, and Department. Enroll a student in a course of a particular department.
Student Class (name, rollno, course, year)

Course Class (name, year)

Department Class (name)

In [40]:
```python
# Department class
class Department:
    def __init__(self, name):
        self.name = name

# Course class
class Course:
    def __init__(self, name, year, department):
        self.name = name
        self.year = year
        self.department = department

# Student class
class Student:
    def __init__(self, name, rollno, year):
        self.name = name
        self.rollno = rollno
        self.year = year
        self.course = None

    def enroll(self, course):
        self.course = course

    def display(self):
        print("Student Name:", self.name)
        print("Roll No:", self.rollno)
        print("Year:", self.year)
        print("Course:", self.course.name)
        print("Department:", self.course.department.name)

# Creating objects
dept = Department("Computer Science")
course = Course("Python Programming", 2025, dept)
student = Student("Riya", 101, 2025)

# Enroll student
student.enroll(course)

# Display details
student.display()
```

```
Student Name: Riya
Roll No: 101
Year: 2025
Course: Python Programming
Department: Computer Science
```

Explanation: This program uses Student, Course, and Department classes where a student is enrolled in a course that belongs to a specific department and their details are displayed.

🔷 EXPERIMENT 12: Operator Overloading and Error Handling

Objective: To understand operator overloading and exception handling in Python and implement programs for input validation and error handling. Date: 20/11/2025

a) Write a program that overloads the + operator on a class Student that has attributes name and marks.

In [41]:
```
class Student:
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks

    def __add__(self, other):
        return self.marks + other.marks

# Creating student objects
s1 = Student("Amit", 85)
s2 = Student("Riya", 90)

# Using overloaded + operator
total_marks = s1 + s2

print("Total Marks:", total_marks)
```

```
Total Marks: 175
```

Explanation: This program overloads the + operator using the **add**() method so that adding two Student objects returns the sum of their marks.

b) Write a program that validates name and age as entered by the user to determine whether the person can cast vote or not.

In [43]:
```
name = input("Enter your name: ")
age = int(input("Enter your age: "))

# Validate name
if not name.isalpha():
    print("Invalid name entered")
else:
    # Validate age for voting
    if age >= 18:
        print(name, "is eligible to vote")
    else:
        print(name, "is not eligible to vote")
```

```
riya is eligible to vote
```

Explanation: This program checks whether the entered name contains only letters and then verifies if the user's age is 18 or above to decide voting eligibility.

c) Write a program that asks the user to enter an integer. If the user enters something that is not an integer (like "abc" or 3.5), the program should catch the error, print "Invalid input, please enter an integer.", and ask again until a valid integer is entered.

In [44]:
```
while True:
    try:
        n = int(input("Enter an integer: "))
        print("You entered:", n)
        break
    except ValueError:
        print("Invalid input, please enter an integer.")
```

```
Invalid input, please enter an integer.
Invalid input, please enter an integer.
You entered: 45
```

Explanation: This program repeatedly asks for input and uses try–except to catch invalid entries until the user enters a valid integer.