Ex: No: 1

Calculation of Test Average

Date: 5/08/2025

AIM:

To write a python program to find the best of two test average marks out of three test's

marks accepted from the user.

ALGORITHM:

Step 1: Start

Step 2: Input three test marks: test1, test2, test3

Step 3: Store the three test marks in a list or array called marks

Step 4: Sort the marks list in descending order

Step 5: Select the first two values from the sorted list as best_two

Step 6: Calculate the average:

average = (best_two[0] + best_two[1]) / 2

Step 7: Display the best two marks

Step 8: Display the calculated average

Step 9: Stop

PROGRAM:

```
 # Accept three test marks from the user
test1 = float(input("Enter marks for Test 1: "))
test2 = float(input("Enter marks for Test 2: "))
test3 = float(input("Enter marks for Test 3: "))
# Store the marks in a list
marks = [test1, test2, test3]
# Sort the list in descending order
marks.sort(reverse=True)
# Calculate the average of the best two marks
best_two_avg = (marks[0] + marks[1]) / 2
# Display the result
print("The best two test marks are:", marks[0], "and", marks[1])
print("The average of the best two test marks is:", best_two_avg)
```

OUTPUT:

```
Enter marks for Test 1: 72
Enter marks for Test 2: 85
Enter marks for Test 3: 68
Best two test marks are: [85.0, 72.0]
Average of best two test marks is: 78.5
```

RESULT:

Thus, the Python program that accepts three test marks and finds the best of two test average marks was successfully written, executed, and verified.

Ex: No:2

Palindrome Check & Digit Occurrence Count

Date: 12/08/2025

AIM:

To develop a Python program to check whether a given number is palindrome or not and

also count the number of occurrences of each digit in the input number.

ALGORITHM

1. Start

2. Input the number as a string (num).

3. Check for Palindrome:

a. Reverse the string using slicing (num[::-1]).

b. Compare the reversed string with the original:

i. If equal → Print that the number is a palindrome.

ii. Else → Print that the number is not a palindrome.

4. Initialize an empty dictionary digit_count.

5. For each character digit in the input string num:

a. If the character is a digit:

i. If the digit exists in the dictionary, increment its count.

ii. Else, add it to the dictionary with count 1.

6. Display the digit occurrence count by iterating over the dictionary in sorted order of digits.

7. End

PROGRAM:

```
# Get input from the user

num = input("Enter a number: ")

# Check for palindrome

if num == num[::-1]:

print(f"{num} is a palindrome.")

else:

print(f"{num} is not a palindrome.")

# Count digit occurrences

digit_count = {}

for digit in num:

if digit.isdigit(): # Ensure only digits are considered

digit_count[digit] = digit_count.get(digit, 0) + 1

print("\nDigit Occurrence Count:")

for digit in sorted(digit_count):

print(f"Digit {digit} occurs {digit_count[digit]} time(s)")
```

OUTPUT:

```
Enter a number: 12321

12321 is a palindrome.

Digit Occurrence Count:

Digit 1 occurs 2 time(s)
```

Digit 2 occurs 2 time(s)

Digit 3 occurs 1 time(s)

RESULT:

Thus, the Python program to check whether a number is palindrome or not and count the number of occurrences of each digit was successfully written, executed, and verified.

Ex: No:3

Fibonacci sequence

Date: 19/08/2025

AIM:

Fibonacci sequence

Fibonacci sequence: Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program

which accepts a value for N (where N >0) as input and pass this value to the function. Display

suitable error message if the condition for input value is not followed.

ALGORITHM:

Step 1: Start

Step 2: Input value N from user

Step 3: Check if N is a digit and N > 0 - If not, print "Invalid input. Please enter a positive integer." - Go to Step 8

Step 4: Convert N to integer

Step 5: Define a function fibonacci(n) - Initialize a = 0, b = 1 - Loop n times: - Print a -

Set a, b = b,

a + b Step 6: Call

fibonacci(N) Step 7: End

function

Step 8: Stop

PROGRAM:

```
# Function to print Fibonacci sequence

def fibonacci(n):

a, b = 0, 1

print("Fibonacci sequence:")

for _ in range(n):

print(a, end=' ')

a, b = b, a + b

# Get user input

N = input("Enter the number of terms (N > 0): ")

# Validate input

if N.isdigit():

N = int(N)

if N > 0:

fibonacci(N)

else:

print("Invalid input. Please enter a number greater than 0.")

else:

print("Invalid input. Please enter a positive integer.")
```

OUTPUT:

```
Enter the number of terms (N > 0): 6

Fibonacci sequence:
```

0 1 1 2 3 5

Enter the number of terms (N > 0): 0

Invalid input. Please enter a number greater than 0.

RESULT:

Thus, the Python program that accepts a value for N (where N > 0) and generates the Fibonacci sequence using a function was successfully written, executed, and verified. The program also correctly displays an appropriate error message when an invalid (non-positive) input is entered.

Ex: No: 4

Binary to Decimal & Octal to Hexadecimal Conversion

Date: 26/08/2025

AIM:

To develop a python program to convert binary to decimal, octal to hexadecimal using

functions with algorithm steps and sample output.

ALGORITHM:

1. Start

2. Define a function is_binary(s) to check if all characters in string s are '0' or '1'

3. Define a function binary_to_decimal(binary_str)

a. If is_binary(binary_str) is True:

i. Convert to decimal using int(binary_str, 2)

ii. Display the decimal value

b. Else, print "Invalid binary number"

4. Define a function is_octal(s) to check if all characters in string s are digits from '0' to '7'

5. Define a function octal_to_hexadecimal(octal_str)

a. If is_octal(octal_str) is True:

i. Convert to decimal using int(octal_str, 8)

ii. Convert to hexadecimal using hex(decimal)[2:].upper()

iii. Display the hexadecimal value

b. Else, print "Invalid octal number"

6. Input binary number as string → binary_input

7. Call binary_to_decimal(binary_input)

8. Input octal number as string → octal_input

9. Call octal_to_hexadecimal(octal_input)

10. Stop

PROGRAM:

```
def is_binary(s):
return all(char in '01' for char in s)
def binary_to_decimal(binary_str):
if is_binary(binary_str):
decimal = int(binary_str, 2)
print(f"Binary to Decimal: {binary_str} → {decimal}")
else:
print("Invalid binary number. Please enter a number containing only 0s and 1s.")
def is_octal(s):
return all(char in '01234567' for char in s)
def octal_to_hexadecimal(octal_str):
```

```
    if is_octal(octal_str):
    decimal = int(octal_str, 8)
    hexadecimal = hex(decimal)[2:].upper()
    print(f"Octal to Hexadecimal: {octal_str} → {hexadecimal}")
    else:
    print("Invalid octal number. Please enter a number containing digits 0 to 7 only.")
    # Main program
    binary_input = input("Enter a binary number: ")
    binary_to_decimal(binary_input)
    octal_input = input("Enter an octal number: ")
    octal_to_hexadecimal(octal_input)
```

OTPUT:

```
Enter a binary number: 1011
Binary to Decimal: 1011 → 11
Enter an octal number: 17
Octal to Hexadecimal: 17 → F
Enter a binary number: 1021
Invalid binary number. Please enter a number containing only 0s and 1s.
Enter an octal number: 198
Invalid octal number. Please enter a number containing digits 0 to 7 only.
```

RESULT:

Thus, the Python program to convert binary to decimal and octal to hexadecimal using functions was

Ex: No: 5

Sentence Statistics

Date: 2/09/2025

AIM:

To write a Python program that accepts a sentence and find the number of words, digits,

uppercase letters and lowercase letters.

ALGORITHM:

1. Start

2. Input a sentence from the user and store it in a variable sentence

3. Initialize counters:

a. word_count = 0

b. digit_count = 0

c. uppercase_count = 0

d. lowercase_count = 0

4. Count words using:

a. word_count = len(sentence.split())

(splits the sentence by whitespace)

5. Iterate through each character ch in the sentence:

a. If ch.isdigit() → increment digit_count

b. If ch.isupper() → increment uppercase_count

c. If ch.islower() → increment lowercase_count

6. Display the values of:

a. word count

b. digit count

c. uppercase letter count

d. lowercase letter count

7. Stop

PROGRAM:

```
# Input
sentence = input("Enter a sentence: ")
# Initialize counters
word_count = len(sentence.split())
digit_count = 0
uppercase_count = 0
lowercase_count = 0
# Iterate over characters
for ch in sentence:
if ch.isdigit():
```

```
    digit_count += 1
    elif ch.isupper():
    uppercase_count += 1
    elif ch.islower():
    lowercase_count += 1
    # Output
    print("Number of words:", word_count)
    print("Number of digits:", digit_count)
    print("Number of uppercase letters:", uppercase_count)
    print("Number of lowercase letters:", lowercase_count)
```

OUTPUT:

```
Enter a sentence: Hello World 123!
Number of words: 3
Number of digits: 3
Number of uppercase letters: 2
Number of lowercase letters: 8
```

RESULT:

Thus, the Python program that accepts a sentence and finds the number of words, digits, uppercase letters, and lowercase letters was successfully written, executed, and verified.

Ex No: 6

String Similarity

Date: 09/09/2025

AIM:

String Similarity

To write a Python program to find the string similarity between two given strings.

ALGORITHM:

1. Start

2. Input two strings: string1 and string2

3. Find the length of the shorter string → min_len

4. Initialize a counter match_count = 0

5. Loop from i = 0 to min_len - 1:

A. If string1[i] == string2[i] → increment match_count

6. Calculate similarity ratio:

B. similarity = (match_count / max(len(string1), len(string2))) * 100

7. Display the similarity percentage

8. Stop

PROGRAM:

```
# Input
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")
# Find length of shorter string
min_len = min(len(string1), len(string2))
max_len = max(len(string1), len(string2))
# Initialize match counter
match_count = 0
# Compare characters
for i in range(min_len):
if string1[i] == string2[i]:
match_count += 1
#Calculate similarity percentage
similarity = (match_count / max_len) * 100
# Output
print(f"String similarity: {similarity:.2f}%")
```

OUTPUT:

```
Enter the first string: apple
Enter the second string: apply
String similarity: 80.00%
Enter the first string: hello
```

Enter the second string: yellow

String similarity: 60.00%

RESULTS:

Thus, the Python program to find the string similarity between two given strings was

successfully written, executed, and verified.

Ex: No:7

Insertion Sort & Merge Sort on lists

Date: 16/09/2025

AIM:

To write a python program to implement insertion sort and merge sort using lists.

ALGORITHM FOR INSERTION SORT

1. Start

2. Input the list of elements

3. Loop from index 1 to end of the list:

o Store the current value in a variable key

o Set j = i - 1

o While j >= 0 and list[j] > key:

▪ Move list[j] to position j + 1

▪ Decrease j by 1

o Place key at position j + 1

4. Display the sorted list

5. Stop

ALGORITHM FOR MERGE SORT

1. Start

2. If the list has more than 1 element:

o Find the middle index

o Split the list into left and right halves

o Recursively call merge sort on both halves

o Merge the two sorted halves into a single sorted list

3. Use a helper function merge() to combine two sorted lists

4. Display the sorted list

5. Stop

PROGRAM FOR INSERTION SORT:

```
def insertion_sort(arr):

for i in range(1, len(arr)):


key = arr[i]

j = i - 1

while j >= 0 and arr[j] > key:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

# Sample input

arr = [34, 8, 64, 51, 32, 21]

print("Original list:", arr)

insertion_sort(arr)

print("Sorted list (Insertion Sort):", arr)
```

PROGRAM FOR MERGE SORT:

```
def merge_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left_half = arr[:mid]

right_half = arr[mid:]

merge_sort(left_half)

merge_sort(right_half)

i = j = k = 0

# Merge the sorted halves

while i < len(left_half) and j < len(right_half):

if left_half[i] < right_half[j]:

arr[k] = left_half[i]

i += 1

else:

arr[k] = right_half[j]

j += 1

k += 1

# Check for remaining elements

while i < len(left_half):

arr[k] = left_half[i]

i += 1
```

```
        k += 1
        while j < len(right_half):
        arr[k] = right_half[j]
        j += 1
        k += 1
        # Sample input
        arr = [34, 8, 64, 51, 32, 21]
        print("Original list:", arr)
        merge_sort(arr)
        print("Sorted list (Merge Sort):", arr)
```

OUTPUT:

```
Original list: [34, 8, 64, 51, 32, 21]
Sorted list (Insertion Sort): [8, 21, 32, 34, 51, 64]
Original list: [34, 8, 64, 51, 32, 21]
Sorted list (Merge Sort): [8, 21, 32, 34, 51, 64]
```

RESULT:

Thus, the Python program to implement insertion sort and merge sort using lists

successfully written, executed, and verified.

Ex: No: 8

Check Phone Number

Date: 23/09/2025

AIM:

To Write a function called isphonenumber () to recognize a pattern 415-555-4242 without using

regular expression and also write the code to recognize the same pattern using regular expression.

ALGORITHM:

Without using Regular Expressions

1. Define a function isphonenumber(s)

2. Check if the length of s is exactly 12

3. Check:

    1.Characters at index 3 and 7 are -

    2.All other characters are digits

4. If all conditions are true, return True; else return False

Using Regular Expressions

1. Import the re module

2. Define the pattern: ^\d{3}-\d{3}-\d{4}$

    1. \d{3}: 3 digits -: hyphen

2. \d{4}: 4 digits

3. Use re.fullmatch() to match the string against the pattern

4. Return True if matched, else False

PROGRAM WITHOUT USING REGULAR EXPRESSION

```
def isphonenumber(s):
if len(s) != 12:
return False
if s[3] != '-' or s[7] != '-':
return False
if not (s[0:3].isdigit() and s[4:7].isdigit() and s[8:12].isdigit()):
return False
return True
# Sample test
print("Without regex:")
print(isphonenumber("415-555-4242")) # True
print(isphonenumber("415-55A-4242")) # False
print(isphonenumber("415-5555-242")) # False
```

PROGRAM USING REGULAR EXPRESSION

```
import re
def isphonenumber_regex(s):
pattern = r"^\d{3}-\d{3}-\d{4}$"
return bool(re.fullmatch(pattern, s))
```

```
# Sample test
print("\nWith regex:")
print(isphonenumber_regex("415-555-4242")) # True
print(isphonenumber_regex("415-55A-4242")) # False
print(isphonenumber_regex("4155554242"))
```

OUTPUT:

```
Without regex:
True
False
False
With regex:
True
False
False
# False
```

RESULT:

Thus, the Python program to recognize a phone number pattern using both normal functions

Ex: No:9 Search Phone Number & Email

Date: 30/09/2025

AIM:

To develop a python program that could search the text in a file for phone numbers

(+919502497844) and email addresses (ramesh123@gmail.com)

ALGORITHM:

1. Start

2. Open the text file in read mode

3. Read the content of the file into a string

4. Import the re module for regular expressions

5. Define:

a) Regex pattern for phone numbers: r"\+91\d{10}"

b) Regex pattern for email addresses: r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"

6. Use re.findall() to find all matches of phone numbers

7. Use re.findall() to find all matches of email addresses

8. Display the extracted phone numbers and email addresses

9. Close the file

10. End

PROGRAM:

```
import re

def extract_contacts(filename):

# Open and read the file

with open(filename, 'r') as file:

content = file.read()

# Regex patterns phone_pattern

= r"\+91\d{10}"

email_pattern = r"[a-zA-ZO-9._%+-]+@[a-zA-ZO-9.-]+\.[a-zA-Z]{2,}"

# Finding all phone numbers and emails

phones = re.findall(phone_pattern, content)

emails = re.findall(email_pattern, content)

# Display results

print("Phone Numbers Found:")

for phone in phones:

print(phone)

print("\nEmail Addresses Found:")

for email in emails:

print(email)

# Example usage

extract_contacts("sample.txt") # Make sure this file exists with test data
```

SAMPLE CONTENT OF SAMPLE.TXT

Here are some contacts:

Call me at +919502497844 or +919845628656.

You can also email me at ramesh123@gmail.com or contact_hr@example.co.in

OUTPUT:

Phone Numbers Found:

+919502497844

+919845628656

Email Addresses Found:

Ramesh123@gmail.com

contact_hr@example.co.in

RESULT:

Thus, the Python program that searches the text in a file for phone numbers and email

addresses using regular expressions was successfully written, executed, and verified.

Ex: No:10 File Operations

DATE: 7/10/2025

AIM:

To write a python program to accept a file name from the user and perform the following

operations 1. Display the first N line of the file 2. Find the frequency of occurrence of the word

accepted from the user in the file.

ALGORITHM:

1. Start

2. Accept the filename from the user.

3. Open the file in read mode.

4. Ask the user to input number of lines (N) to display.

5. Read and display the first N lines.

6. Ask the user to enter a word to search.

7. Read the entire file again.

8. Convert content and word to lowercase.

9. Use .count() to find the number of occurrences of the word.

10. Display the frequency.

11. Close the file.

12. End

PROGRAM:

```
def main():
# Step 1: Input file name
filename = input("Enter the filename: ")
# Step 2: Open file and display first N lines
file = open(filename, 'r')
N = int(input("Enter number of lines to display: "))
print(f"\nFirst {N} lines of the file:")
for i in range(N):
line = file.readline()
if line == ':
print("End of file reached.")
break
print(line.strip())
file.close()
# Step 3: Count word frequency
word = input("\nEnter the word to find its frequency: ").lower()
file = open(filename, 'r')
content = file.read().lower()
frequency = content.count(word)
file.close()
print(f"\nThe word '{word}' occurred {frequency} times in the file.")
```

```
# Run the function
main()
```

OUTPUT:

```
Enter the filename: demo.txt
Enter number of lines to display: 2
Enter the word to find its frequency: data
First 2 lines of the file:
Data is essential in today's world.
Big data is changing the industry.
The word 'data' occurred 2 times in the file.
```

RESULT:

Thus, the Python program that displays the first N lines of a file and finds the frequency of a given word

was successfully written, executed, and verified.

Ex: No:11 **Zip operation on a folder**

Date: 14/10/2025

AIM:

To develop a program to backing up a given Folder (Folder in a current working directory)

into a ZIP File by using relevant modules and suitable methods.

ALGORITHM:

1. Start

2. Import the required modules: os, zipfile

3. Get the folder name from the user (it should be a folder in the current working directory)

4. Create a valid ZIP file name using the folder name (e.g., foldername_backup.zip)

5. Create a new ZIP file using zipfile.ZipFile() in write mode

6. Use os.walk() to iterate over all files and subfolders inside the folder

7. For each file:

o Create a relative path for the file inside the ZIP

o Write the file to the ZIP using .write()

8. Close the ZIP file

9. Display a message that the folder has been successfully backed up

10. End

PROGRAM:

```
import os
import zipfile
def backup_folder_to_zip(folder_name):
# Step 1: Construct ZIP file name
zip_filename = folder_name + '_backup.zip'
# Step 2: Create the ZIP file
with zipfile.ZipFile(zip_filename, 'w') as backup_zip:
# Step 3: Walk through the folder
for folderpath, subfolders, filenames in os.walk(folder_name):
for filename in filenames:
file_path = os.path.join(folderpath, filename)
# Store the file with relative path
relative_path = os.path.relpath(file_path, folder_name)
backup_zip.write(file_path, arcname=os.path.join(folder_name, relative_path))
print(f"Folder '{folder_name}' has been successfully backed up to
'{zip_filename}'.")
# Main Program
folder = input("Enter the folder name to backup (must be in current directory):
")
if os.path.isdir(folder):
backup_folder_to_zip(folder)
```

```
else:
print("The specified folder does not exist in the current directory.")
```

OUTPUT:

```
Enter the folder name to backup (must be in current directory): myproject
Folder 'myproject' has been successfully backed up to 'myproject_backup.zip'.
```

RESULT:

Thus, the Python program that creates a ZIP backup of a specified folder (from the current working directory) using shutil.make_archive() was successfully written, executed, and verified.

The ZIP file is created in the same directory and the program reports the full path of the created archive.

Ex: No:12 Inheritance

Date: 28/10/2025

AIM:

To write a python program to find the area of triangle, circle and rectangle by using the

concept of inheritance.

ALGORITHM:

1. Start

2. Define a base class Shape with a method area() (can be an empty or general method).

3. Define a derived class Triangle that inherits from Shape:

a) Take base and height as input

b) Implement the area() method as: (0.5 * base * height)

4. Define a derived class Rectangle that inherits from Shape:

a) Take length and width as input

b) Implement the area() method as: length * width

5. Define a derived class Circle that inherits from Shape:

a) Take radius as input

b) Implement the area() method as: π * radius^2

6. Create instances of each derived class and call the area() method

7. Display the results

8. End

PROGRAM:

```
import math

# Base Class

class Shape:

def area(self):

return 0

# Derived Class for Triangle

class Triangle(Shape):

def  init  (self, base, height):

self.base = base

self.height = height

def area(self):

return 0.5 * self.base * self.height

# Derived Class for Rectangle

class Rectangle(Shape):

def  init (self, length, width):

self.length = length

self.width = width

def area(self):

return self.length * self.width

# Derived Class for Circle
```

```
class Circle(Shape):

def  init (self, radius):

self.radius = radius

def area(self):

return math.pi * self.radius ** 2

# Main Program

print("Area Calculations:")

# Triangle

b = float(input("Enter base of triangle: ")) h

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

triangle = Triangle(b, h)

print(f"Area of Triangle: {triangle.area():.2f}")

# Rectangle

l = float(input("\nEnter length of rectangle: "))

w = float(input("Enter width of rectangle: "))

rectangle = Rectangle(l, w)

print(f"Area of Rectangle: {rectangle.area():.2f}")

# Circle

r = float(input("\nEnter radius of circle: "))

circle = Circle(r)

print(f"Area of Circle: {circle.area():.2f}")
```

OUTPUT:

Area Calculations:

Enter base of triangle: 10

Enter height of triangle: 5

Area of Triangle: 25.00

Enter length of rectangle: 8

Enter width of rectangle: 4

Area of Rectangle: 32.00

Enter radius of circle: 7

Area of Circle: 153.94

RESULT:

Thus, the Python program to find the area of triangle, circle, and rectangle using inheritance

was successfully written, executed, and verified.

Ex: No:13 Employee Details using Class

Date: 4/11/2025

AIM:

To write a python program by creating a class called Employee to store the details of Name,

Employee_ID, Department and Salary, and implement a method to update salary of employees

belonging to a given department.

ALGORITHM:

1. Start

2. Define a class Employee with attributes:

a) name

b) emp_id

c) department

d) salary

3. Define the __init__() method to initialize these attributes.

4. Define a method update_salary_by_dept(self, percentage) inside the class to update the

salary based on the given percentage (optional: can be done externally too).

5. Create a list of Employee objects.

6. Accept department name and increment percentage from the user.

7. Traverse the list and for each employee:

a) If employee's department matches the input department:

i)

Update salary by applying the percentage increase.

8. Display updated employee details.

9. End

PROGRAM:

```
# Define the Employee class

class Employee:

def  init (self, name, emp_id, department, salary): self.name

= name

self.emp_id = emp_id

self.department = department

self.salary = salary

def display(self):

print(f"Name: {self.name}, ID: {self.emp_id}, Dept: {self.department}, Salary:

{self.salary}")

def update_salary(self, percentage):

self.salary += self.salary * (percentage / 100)

# Create list of Employee objects
```

```
employees = [
Employee("Alice", 101, "HR", 50000),
Employee("Bob", 102, "IT", 60000),
Employee("Charlie", 103, "HR", 52000),
Employee("David", 104, "Finance", 55000),
Employee("Eva", 105, "IT", 58000)
]
# Accept department and increment percentage
dept = input("Enter department to update salary: ")
percent = float(input("Enter salary increment percentage: "))
print("\n--- Updated Employee Details ---")
for emp in employees:
if emp.department.lower() == dept.lower():
emp.update_salary(percent)
emp.display()
```

OUTPUT:

```
Enter department to update salary: HR
Enter salary increment percentage: 10 --- Updated Employee Details ---
Name: Alice, ID: 101, Dept: HR, Salary: 55000.0
Name: Bob, ID: 102, Dept: IT, Salary: 60000
Name: Charlie, ID: 103, Dept: HR, Salary: 57200.0
Name: David, ID: 104, Dept: Finance, Salary: 55000
```

Name: Eva, ID: 105, Dept: IT, Salary: 58000

RESULT:

Thus, the Python program to store employee details and update salary of employees belonging

to a given department was successfully written, executed, and verified.

Ex: No:14 Polymorphism and Inheritance

Date: 11/11/2025

AIM:

To Write a python program to find the whether the given input is palindrome or not (for

both string and integer) using the concept of polymorphism and inheritance.

ALGORITHM:

1. Start

2. Define a base class called PalindromeChecker with a method is_palindrome(self, value).

3. Define two derived classes:

a) StringPalindromeChecker that overrides is_palindrome() to handle strings.

b) IntegerPalindromeChecker that overrides is_palindrome() to handle integers.

4. In the is_palindrome() method of each class:

a) Reverse the input (string or number) and compare it with the original.

5. Accept input from the user and determine whether it is an integer or string.

6. Create an object of the appropriate class.

7. Call the is_palindrome() method and display the result.

8. End

PROGRAM:

```
# Base class

class PalindromeChecker:

def is_palindrome(self, value):

raise NotImplementedError("Subclasses should implement this!")

# Derived class for string palindrome

class StringPalindromeChecker(PalindromeChecker):

def is_palindrome(self, value):

value = value.lower().replace(" ", "") # Normalize string

return value == value[::-1]

# Derived class for integer palindrome

class IntegerPalindromeChecker(PalindromeChecker):

def is_palindrome(self, value):

str_value = str(value)

return str_value == str_value[::-1]

# Main logic

input_value = input("Enter a value (string or integer): ")

if input_value.isdigit():

checker = IntegerPalindromeChecker()

is_pal = checker.is_palindrome(int(input_value))
```

```
else:
checker = StringPalindromeChecker()
is_pal = checker.is_palindrome(input_value)
# Display the result
if is_pal:
print("The input is a palindrome.")
else:
print("The input is not a palindrome.")
```

OUTPUT:

```
Enter a value (string or integer): 12321
The input is a palindrome.
Enter a value (string or integer): Racecar
The input is a palindrome.
Enter a value (string or integer): Hello
The input is not a palindrome.
```

RESULT:

Thus, the Python program using polymorphism and inheritance to check whether a given string or integer is a palindrome was successfully written, executed, and verified.

Ex: No:15 Spreadsheet Operations

Date: 11/11/2025

AIM:

To demonstrate python program to read the data from the spreadsheet and write the data

in to the spreadsheet

ALGORITHM:

1. Start

2. Import the openpyxl module.

3. Load the existing Excel file using openpyxl.load_workbook().

4. Select the worksheet to read from using workbook[sheet_name].

5. Read data from specific rows and columns using loops or direct cell references.

6. Create a new workbook (or load another workbook) for writing using Workbook() or

load_workbook().

7. Select or create a sheet for writing.

8. Write the read data into cells using sheet.cell(row, column).value = data.

9. Save the workbook using workbook.save(filename).

10. End

Required Library

You must install openpyxl if not already installed:

PROGRAM

```
import openpyxl
from openpyxl import Workbook
# Step 1: Load the workbook and select the sheet to read
read_wb = openpyxl.load_workbook("input_data.xlsx")
read_sheet = read_wb.active
# Step 2: Create a new workbook for writing
write_wb = Workbook()
write_sheet = write_wb.active
# Step 3: Read data from read_sheet and write to write_sheet
for i in range(1, read_sheet.max_row + 1):
for j in range(1, read_sheet.max_column + 1):
# Read from input sheet
cell_value = read_sheet.cell(row=i, column=j).value
# Write to output sheet
write_sheet.cell(row=i, column=j).value = cell_value
# Step 4: Save the new workbook
write_wb.save("output_data.xlsx")
print("Data has been read from 'input_data.xlsx' and written to
'output_data.xlsx'")
```

OUTPUT:

Assume your input_data.xlsx file contains:

Name

John

Age

25

Alice

30

After execution, a new file output_data.xlsx is created with the same data.

RESULT:

Thus, the Python program to read and write data from a spreadsheet using the openpyxl

module was successfully written, executed, and verified.

Ex: No:16 Merge selected pages from Multiple PDFs to a new PDF

Date: 18/11/2025

AIM:

To write a python program to combine select pages from many PDFs.

ALGORITHM:

1. Start

2. Import the required module: PyPDF2

3. Create a PdfWriter object to write the final output PDF

4. For each input PDF:

a) Open the file in binary read mode

b) Create a PdfReader object

c) For each required page number:

i) Check if the page exists

ii) Add the selected page to the PdfWriter object

5. Write the combined pages into a new PDF file

6. Close all file streams

7. End

REQUIRED LIBRARY:

1. Page numbers are zero-indexed (0 = page 1).

2. Make sure all input PDF files exist in the same directory or give full paths.

3. Install PyPDF2 if not already installed:

Install using the statement – pip install PyPDF2

PROGRAM:

```
import PyPDF2

def combine_selected_pages(pdf_info_list, output_filename):
"""
Combines selected pages from multiple PDF files.
:param pdf_info_list: List of tuples (pdf_filename, list_of_page_numbers)
:param output_filename: Name of the output PDF file
"""
pdf_writer = PyPDF2.PdfWriter()
for pdf_file, page_numbers in pdf_info_list:
with open(pdf_file, 'rb') as file:
pdf_reader = PyPDF2.PdfReader(file)
total_pages = len(pdf_reader.pages)
for page_num in page_numbers:
if 0 <= page_num < total_pages:
pdf_writer.add_page(pdf_reader.pages[page_num])
else:
print(f"Page number {page_num} is out of range in {pdf_file}")
with open(output_filename, 'wb') as output_file:
pdf_writer.write(output_file)
```

```
print(f"Combined PDF saved as '{output_filename}'")

# Example usage:

pdf_inputs = [

("sample1.pdf", [0, 2]), # 1st and 3rd page from sample1.pdf

("sample2.pdf", [1]),

# 2nd page from sample2.pdf

("sample3.pdf", [0, 1]) # 1st and 2nd pages from sample3.pdf

]

combine_selected_pages(pdf_inputs, "combined_output.pdf")
```

OUTPUT:

```
Combined PDF saved as 'combined_output.pdf'
```

RESULT:

Ex: No:17 Fetch weather data from the JSON

Date: 18/11/2025

AIM:

To fetch weather data from the JSON: Write a python program to fetch current weather

data from the JSON file

ALGORITHM:

1. Start

2. Import the json module

3. Define a function fetch_weather_from_json(file_path)

4. Open the JSON file in read mode

5. Use json.load() to parse the JSON content into a Python dictionary

6. Access the following data from the dictionary:

a) location["name"] – Name of the city

b) location["country"] – Name of the country

c) current["temperature"] – Current temperature

d) current["humidity"] – Current humidity

e) current["wind_speed"] – Current wind speed

f) current["weather_descriptions"][0] – Current weather description

7. Print the weather report in a readable format

8. End the function

9. Call the function with the path to the JSON file

Sample JSON File (weather.json)

```
{
"location": { "name":
"Chennai",
"country": "India"
},
"current": {
"temperature": 31,
"humidity": 76,
"wind_speed": 12,
"weather_descriptions": ["Partly cloudy"]
}
}
```

PROGRAM:

```
import json
def fetch_weather_from_json(file_path):
with open(file_path, 'r') as f:
data = json.load(f)
location = data['location']['name']
country = data['location']['country']
temperature = data['current']['temperature']
```

```
humidity = data['current']['humidity']

wind_speed = data['current']['wind_speed']

description = data['current']['weather_descriptions'][0]

print(f"Weather Report for {location}, {country}")

print(f"Condition : {description}")

print(f"Temperature : {temperature}°C")

print(f"Humidity : {humidity}%")

print(f"Wind Speed : {wind_speed} km/h")

# Example usage

fetch_weather_from_json("weather.json")
```

OUTPUT

```
Weather Report for Chennai, India

Condition : Partly cloudy

Temperature : 31°C

Humidity : 76%

Wind Speed : 12 km/h
```

RESULT:

Thus, the Python program to fetch current weather data from a JSON file was successfully written, executed, and verified.

Ex: No:18

BASIC PYTHON PROGRAMS

Date:

1) Ask the users first name and display the output messages as "Hello" first name

PROGRAM:

```
Firstname=input()
print("Hello",Firstname)
```

OUTPUT:

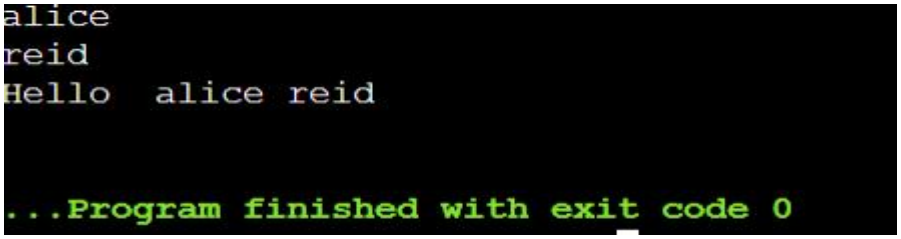
```
alexa
Hello alexa


...Program finished with exit code 0
```

2) Ask the users first name and then ask for their surname and display the output message "Hello"First name surname

PROGRAM:

```
Firstname=input()
Surname=input()
Print("Hello",Firstname,Surname)
```

OUTPUT:

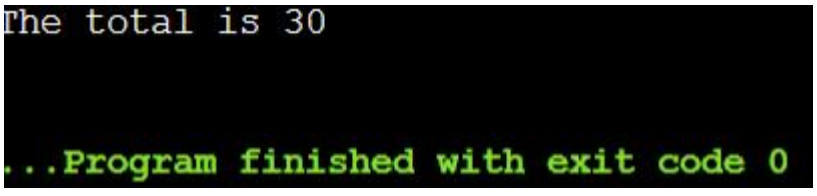


3) Ask the user to enter two numbers.Add then together and display the answer as the total is answer.

PROGRAM:

```
a=10
b=20
print("The total is",a+b)
```

OUTPUT:



4) Ask for the total price of the bill,then ask how many diners there are.Divide the total bill by the number of diners and show how much each person must pay.

PROGRAM:

```
a=int(input())
b=int(input())
c=a/b
print("total price",a)
print("diners are",b)
print("each person must pay",c)
```

OUTPUT:

```
total price 2000
diners are 2
each person must pay 1000.0

...Program finished with exit code 0
```

5)Ask the user to enter three numbers add together the first two number and then multiply the total by the third.Display the answer as the answer is

PROGRAM:

```
a=10
b=20
c=30
print("The answer is",a+b+c)
```

OUTPUT:

```
The answer is 60


...Program finished with exit code 0
```

6)Ask the user for a number of days and then will show many hours,minutes,seconds are there in that number of days.

PROGRAM:

```
a=int(input())
print("number of days",a)
print("number of hours",a*24)
print("number of minutes",a*24*60)
print("number of seconds",a*24*60*3600)
```

OUTPUT:

```
3
number of days 3
number of hours 72
number of minutes 4320
number of seconds 15552000
```

7)Task the user to enter a number over 100 and then enter a number under 10 and tell how many times the smaller number goes into the larger number in a user friendly format.

PROGRAM:

```
a=110
b=5
print("number of times the smaller number goes into the larger number",a/b)
```

OUTPUT:

```
number of times the smaller number goes into the larger number 22.
0
```

8) There are 2204 pounds in akilogram.Ask the user and enter weight in kilograms and convert it into pounds

PROGRAM:

```
a=int(input())
print("weight in kilograms to pounds",a*2204)
```

OUTPUT:

```
5
weight in kilograms to pounds 11020
```

9)Ask how many slices of pizza the user started with and ask how many slices they have eaten.work out how many slices they have left and display the answer in a user friendly format.

PROGRAM:

```
a=10
b=5
print("total slices",a)
print("slices eaten",b)
print("slices they have left",a-b)
```

OUTPUT:

```
total slices 10
slices eaten 5
slices they have left 5
```

10) Write a program to check the given number is +ve/-ve find whether it is even or odd

PROGRAM:

```
a=int(input())
if a>0:
    print(a,''is positive'')
if (a%2==0):
    print(a,''is even'')
else:
    print(a,''is odd'')
else:
    print(a,''is negative'')
```

EX NO: 19

CONTROL STATEMENTS

DATE:

1)Program using control statements

PROGRAM:

```
a=int(input())
b= int(input())
c= int(input())
if a>b and a>c:
    print(a,''is bigger from given numbers'')
elif b>c and b>a:
    print(b,''is bigger from given numbers'')
else:
    print(c,''is bigger from given numbers'')
```

OUTPUT:

```
6
9
9 is bigger from given numbers
```

2) Calculation of Simple interest

PROGRAM:

```
p=int(input())
n=3
if (p>=1,00,000):
   r=5/100
   print("simple interest",((p*n*r)//100))
else:
      r=3.5/100
      print("simple interest",(p*n*r/100))
```

OUTPUT:

```
200000
simple interest 300.0
```

EX NO:20

LISTS

DATE:

PROGRAM:

```
a=[1,3,4,5,6,7]
for i in range(0,len(a)):
    e=a.pop(0)
    a.append(e)
    print(a)
```

OUTPUT:

```
[6, 7, 1, 3, 4, 5]
[7, 1, 3, 4, 5, 6]
[1, 3, 4, 5, 6, 7]
```

PROGRAM:

```
my_list=['p','r','o','g','r','a','m','m','e']
print(my_list[2:5])
print(my_list[5:])
print(my_list[:])
list1=[10,15,11,65,30]
list1.insert(1,80)
```

```
print("inserted elements in the list:",list1)

list1.sort()

print("sorted elements in the list is:",list1)

list1.reverse()

print("reversed element in the list is :",list1)

list1.pop(3)

print("removed element in the list is :",list1)

print("largest element in the list is :",max(list1))

print("smallest element in the list is :",min(list1))
```

OUTPUT:

```
removed element in the list is : [80, 65, 30, 11, 10]
largest element in the list is : 80
smallest element in the list is : 10
```

PROGRAM:

```
a=[2,4,7,9,1,4]

for i in a:

    if i%2!=0:

        print(i)
```

OUTPUT:

```
7
9
1
```

PROGRAM:

```
fruits=["apple","banana","cherry","kiwi","mango"]

newlist=[]

for x in fruits:

    if "a" in x:

        newlist.append(x)

print(newlist)
```

OUTPUT:

```
['apple', 'banana', 'mango']
```

PROGRAM:

```
a=[2,5,10,15,20,18]

for i in a:

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

        print(i)
```

OUTPUT:

```
15
```

PROGRAM:

```
a=[1,2,4,4,8,5,5]
b=[]
for i in a:
    b=append(a.count(i))
    print(b)
```

OUTPUT:

```
1
1
2
2
1
2
2
Counts list: [1, 1, 2, 2, 1, 2, 2]
```

EX NO:21

TUPLE

DATE:

PROGRAM:

```
tuple1 = (1, 'a', 2, 'b')

tuple2 = (2, 'c', 4, 'd')

print(type(tuple1))

print(len(tuple1))

print(tuple1 + tuple2)

print(tuple2 * 3)
```

OUTPUT:

```
<class 'tuple'>
4
(1, 'a', 2, 'b', 2, 'c', 4, 'd')
(2, 'c', 4, 'd', 2, 'c', 4, 'd', 2, 'c', 4, 'd')
```

PROGRAM:

```
t=tuple()

n=int(input("how many values you want to enter"))

for i in range(n):

    a=input("enter number")

    t=t+(a,)

print("enter numbers are")

print(t)
```

OUTPUT:

```
how many values you want to enter5
enter number4
enter number6
enter number8
9enter number
enter number2
enter numbers are
('4', '6', '8', '9', '2')
```

EX NO:22

DICTIONARY

DATE:

PROGRAM:

```
country_capital={"united staes":"washington
dc","england":"london","germany":"berlin","canada":"ottawa"}
print(len(country_capital))
print(country_capital)
country_capital["italy"]="rome"
print(country_capital)
del country_capital["germany"]
print(country_capital)
country_capital.clear()
```

```
print(country_capital)
```

OUTPUT:

```
'berlin', 'canada': 'ottawa', 'italy': 'rome'}
{'united staes': 'washington dc', 'england': 'london', 'canada': '
ottawa', 'italy': 'rome'}
{}
```

EX NO:23

STRING

DATE:

PROGRAM:

```
l,u,p,d=0,0,0,0

s="R@m@_fOrtu9e$"
```

```
if(len(s)>=8):
    for i in s:
        if(i.islower()):
            l+=1
        if(i.isupper()):
            u+=1
        if(i.isdigit()):
            d+=1
        if(i=='@' or i=='$' or i=='_'):
            p+=1
if(l>=1 and u>=1 and d>=1 and p>=1 and l+u+p+d ==len(s)):
    print("Valid Password")
else:
    if (u == 0):
        print("upper case is missing")
    elif (l == 0):
        print("lower case is missing")
    elif (p == 0):
        print("character is missing")
```

```
    elif (d == 0):
        print("digit is missing")
    print("Invalid password")
```

OUTPUT:

```
Valid Password
```

PROGRAM:

```
t="gud mrng5"
for i in t:
    if (i.isdigit() and i.isalpha()):
        print(i)
```

PROGRAM:

```
t="Hello world!"
u=t.replace("world","everyone")
print(u)
```

OUTPUT:

```
Hello everyone!
```

PROGRAM:

```
t= " Hello World! "
```

```
cleaned=t.strip()

print(cleaned)
```

OUTPUT:

```
Hello World!
```

PROGRAM:

```
str='apple mango apple orange orange apple guava mango mango'

str=str.split()

print(str)

str2=[]

for i in str:

    if i not in str2:

        str2.append(i)

print(str2)

for i in range(0, len(str2)):

    print("Frequency of",str2[i],'is:',str.count(str2[i]))
```

OUTPUT:

```
Frequency of apple is: 3
Frequency of mango is: 3
Frequency of orange is: 2
Frequency of guava is: 1
```

PROGRAM:

```
a="Hello, World!"

print(a[2:5])

print(a[:5])

print(a[2:])

print(a[-5:-1])
```

OUTPUT:

```
llo
Hello
llo, World!
orld
```

Ex: No:24

FILE HANDLING

Date:

PROGRAM:

1.

```
import os.path
import sys
fname = input("Enter the filename : ")
if not os.path.isfile(fname):
print("File", fname, "doesn't exists")
sys.exit(0)
infile = open(fname, 'r')
lineList = infile.readlines()
lineCount = int(input("Enter the number of lines you want to display :
"))
for i in range(lineCount):
print(i+1, ":", lineList[i], end="")
word = input("Enter a word : ")
cnt = 0
for line in lineList:
cnt += line.count(word)
print("The word", word, "appears", cnt, "times in the file")
```

OUTPUT:

```
Enter the filename : wel.txt
Enter the number of lines you want to display : 1
1 : welcome to cse..
Enter a word : cse
The word cse appears 1 times in the file.
```

[Program finished]

PROGRAM:

2 # create_wel_file.py

```
file_path = "wel.txt"
content = "welcome to cse.."
  with open(file_path, "w", encoding="utf-8") as f:
f.write(content)
  print(f"Created {file_path} with content: {content!r}")
```

OUTPUT:

Created wel.txt with content: 'welcome to cse..'

[Program finished]

PROGRAM:

3. # create_example_file.py

```
  # File name and content
   file_name = "example.txt"
   content = " cse python programming."
 # Create and write to the file
```

```
    with open(file_name, "w", encoding="utf-8") as f:
        f.write(content)
        print(f"File '{file_name}' created successfully!")
    print("Content written to file:")
        print(content)
```

OUTPUT:

```
File 'example.txt' created successfully!
Content written to file:
cse python programming.
[Program finished]
```

PROGRAM:

4.
```
from random import randint
    def main():
    # Write 50 random numbers into a file
    fp1 = open("WriteNumRandom.txt", "w")
    for i in range(50):
    x = randint(500, 1000)
    fp1.write(str(x) + " ")
```

```
        fp1.close()
        print("50 random numbers written to 'WriteNumRandom.txt' successfully.\n")
        # Read the entire content using read()
        fp1 = open("WriteNumRandom.txt", "r")
        data = fp1.read()
        print("📖 Data read using read():")
        print(data)
        fp1.close()
        # Read the content using readlines()
        fp1 = open("WriteNumRandom.txt", "r")
        lines = fp1.readlines()
        print("\n📖 Data read using readlines():")
        print(lines)
        fp1.close()
        main()
```

OUTPUT:

```
50 random numbers written to 'WriteNumRandom.txt' successfully.
📖 Data read using read():
```

613 802 685 834 686 702 907 785 870 779 912 867 509 639 923 789 520 704 723 765 787 844 875 898 500 963 909 855 838 714 524 762 596 684 726 910 869 604 508 626 826 841 845 833 540 946 860 623 737 681

▨ Data read using readlines():

['613 802 685 834 686 702 907 785 870 779 912 867 509 639 923 789 520 704 723 765 787 844 875 898 500 963 909 855 838 714 524 762 596 684 726 910 869 604 508 626 826 841 845 833 540 946 860 623 737 681 ']

[Program finished]

PROGRAM:

5.
```
# Open readdemo1.txt in write mode
with open("readdemo1.txt", "w") as file:
file.write("Python programming\n")
file.write("Welcome to\n")
file.write("Language\n")
file.write("Programming language\n")
print("readdemo1.txt created successfully with content.")
```

OUTPUT:

readdemo1.txt created successfully with content.

[Program finished]

PROGRAM:

6.
```
def main():
    obj1 = open("Demo1.txt", "w")  # Opens file in Write mode
    obj1.write("Hello, How are You?\n")
    obj1.write("Welcome to The chapter File Handling.\n")
    obj1.write("Enjoy the session.\n")
    obj1.close()
    main()
    print("Demo1.txt created successfully with content.")
```

OUTPUT:

Demo1.txt created successfully with content.

[Program finished]

PROGRAM:

7.
```
file_object = open("my_data.txt", "a")
 print("Enter lines to append to the file. Type 'exit' to stop.")
while True:
```

```
    line = input("Enter line to append: ")
    if line.lower() == 'exit':
    break
    file_object.write(line + "\n")
    # Close the file
  file_object.close()
  print("Data appended to 'my_data.txt' successfully.")
```

OUTPUT:

```
Enter lines to append to the file. Type 'exit' to stop.
Enter line to append: 1
Enter line to append: 5
Enter line to append: 8
Enter line to append: exit
Data appended to 'my_data.txt' successfully.
[Program finished
```

PROGRAM:

8.
```
with open("my_data.txt", "r") as file1:
    read_content = file1.read()
    print(read_content)
```

OUTPUT: 3

6

8

[Program finished]

PROGRAM:

9. with open("example.txt", "r") as file:

    content = file.read() # Reads the entire file

    print(content)

    f = open("Wel.txt", "r")

    print(f.read(3))

    f = open("Wel.txt", "r")

    print(f.read(-1))

    f = open("Wel.txt", "r")

    print(f.readable())

    f = open("Wel.txt", "r")

    f.seek(2)

    print(f.readline())

    f = open("WeL.txt", "r")

    print(f.seekable())

```
f = open("WeL.txt", "r")
print(f.tell())
f = open("wel.txt", "r")
print(f.readline())
print(f.tell())
```

OUTPUT:

```
 cse python programming.
wel
welcome to cse..
True
lcome to cse..
True
0
Welcome to cse..
16
[Program finished]
```

EX NO:26

EXCEPTION HANDLING:

DATE:

PROGRAM:

1. try:

    n = 0

    res = 100 / n

except ZeroDivisionError:

print("You can't divide by zero!")

try :

```
        n = int(input('enter value '))
      except ValueError:
      print("Enter  only  numbers except 0 !")
       n = int(input('enter value '))
       res = 100 / n
       print(res)
else:
   print("Result is", res)
finally:
   print("Execution complete.")
```

OUTPUT:

```
You can't divide by zero!
enter value 4422
Execution complete.
[Program finished]
```

PROGRAM:

```
2. def divide_num(num):
  try:
```

```
    f = open("testfile", "a")
    r = 10/num
    f.write("Result 10/%d is %d" %(num,r))
  except ZeroDivisionError as error:
    print(error)
    print('Zero is not a valid argument here')
  except FileNotFoundError as error:
    print(error)
    print('Passed file does not exist')
  finally:
    print('closing file')
    f.close()
divide_num(0)
```

OUTPUT:

```
division by zero
Zero is not a valid argument here
closing file
[Program finished]
```

EX NO:26

COLLECTIONS,MODULES AND PACKAGES

DATE:

PROGRAM:

```
from collections import namedtuple

point=namedtuple('point',['x','y'])

p=point(10,20)

print(p.x)

print(p.y)

print(p)
```

OUTPUT:

```
10
20
point(x=10, y=20)
```

PROGRAM:

```
from collections  import Counter

counts=Counter(['a','b','c','b'])

print(counts)
```

OUTPUT:

```
Counter({'b': 2, 'a': 1, 'c': 1})
```

PROGRAM:

```
from collections import deque

d=deque([1,2,3])

print(d)

d.appendleft(0)

print(d)

d.pop()

print(d)
```

OUTPUT:

```
deque([1, 2, 3])
deque([0, 1, 2, 3])
deque([0, 1, 2])
```

PROGRAM:

```
from collections import defaultdict

s=[('yellow',1),('blue',2),('yellow',3),('blue',4),('red',1)]
```

```
d=defaultdict(list)

for k,v in s:

    d[k].append(v)

print(d)
```

OUTPUT:

```
defaultdict(<class 'list'>, {'yellow': [1, 3], 'blue': [2, 4], 're
d': [1]})
```

PROGRAM:

```
import random

num=random.random()

num=num*100

print(num)
```

OUTPUT:

```
6.10081865008929
```

PROGRAM:

```
import re

t="apple,orange;banana,grape"

fruits=re.split('[;,]',t)

print(fruits)
```

```
f=["apple","orange","banana"]

j=",".join(f)

print(j)
```

OUTPUT:

```
['apple', 'orange', 'banana', 'grape']
apple,orange,banana
```

PROGRAM:

```
import random

num=random.random()

num=num*100

print(num)
```

OUTPUT:

```
27.715216855852766
```

PROGRAM:

```
import random

num=random.randint(0,9)

print(num)
```

OUTPUT:

```
6
```

PROGRAM:

```
import random

num1=random.randint(0,1000)

num2=random.randint(0,1000)

newrand=num1/num2

print(newrand)
```

OUTPUT:

```
2.5901639344262297
```

PROGRAM:

```
import shelve

shelf_file=shelve.open('mydata.db',flag='c')

shelf_file['my_string']="hello,shelve!"

shelf_file['my_list']=[10,20,30,40]

shelf_file['my_number']=123

shelf_file.close()

shelf_file = shelve.open('mydata.db',flag='r')

retrieved_string =shelf_file['my_string']

retrieved_list=shelf_file['my_list']

retrieved_number=shelf_file['my_number']
```

```
print(f"REtrieved string:{retrieved_string}")

print(f"Retrieved list:{retrieved_list}")

print(f"Retrieved number:{retrieved_number}")

shelf_file.close()
```

OUTPUT:

```
REtrieved string:hello,shelve!
Retrieved list:[10, 20, 30, 40]
Retrieved number:123
```

PROGRAM:

```
user_tags={

    "U001":{"admin","developer","tester"},

    "U002":{"developer","frontened"},

    "U003":{"tester"}

}

def add_tag_to_user(user_id,tag):

 """Adds a tag to users set of tags"""

 if user_id in user_tags:

     user_tags[user_id].add(tag)

     print(f"Tag'{tag}'added to user'{user_id}'.")

 else:
```

```
    print("User not found")
def check_user_has_tag(user_id,tag):
 """Checks if the user has specific tag"""
 if user_id in user_tags:
   if tag in user_tags[user_id]:
     print(f"User'{user_id}'HAS tag'{tag}'.")
     return True
   else:
      print(f"User'{user_id}'DOES NOT have tag'{tag}'.")
 else:
     print("User not found")
     return False
add_tag_to_user("U001","backend")
check_user_has_tag("U002","developer")
check_user_has_tag("U003","admin")
```

OUTPUT:

```
Tag'backend'added to user'U001'.
User'U002'HAS tag'developer'.
User'U003'DOES NOT have tag'admin'.
```

PROGRAM:

```
students={
    "S001":{"name":"Alice","age":20,"courses":["MAth","Physics"]},
    "S002":{"name":"Bob","age":21,"courses":["Chemistry","Biology"]},
    "S003":{"name":"Charlie","age":19,"courses":["Math","Computer
Science"]}
}
def add_student(student_id,name,age,courses):
 """Adds a new student to the system."""
 students[student_id]={"name":name,"age":age,"courses":courses}
 print(f"Student{name}added.")
def enroll_course(student_id,course_name):
 """Enrolls a student in new course."""
 if student_id in students:
    students[student_id]["courses"].append(course_name)
    print(f"Student{students[student_id]['name']}enrolled
in{course_name}.")
 else:
    print("Student not found")
def get_student_info(student_id):
 """Retrives and prints a students information"""
```

```
 if student_id in students:
     info =students[student_id]

print(f"ID:{student_id},Name:{info['name']},Age:{info['age']},Courses:{','.joi
n(info['courses'])}")
 else:
     print("Student not found.")
add_student("S004","David",22,["History"])
enroll_course("S001","Chemistry")
get_student_info("S001")
```

OUTPUT:

```
StudentDavidadded.
Student not found
Student not found.
```

PROGRAM:

```
inventory={
    "Laptop":(1200,10),
    "Mouse":(25,50),
    "Keyboard":(75,20)
}
```

```
def add_product(product_name,price,quantity):
    """adds new product to inventory"""
    inventory[product_name]=(price,quantity)
    print(f"Product'{product_name}'added.")
def update_quantity(product_name,new_quantity):
    """Updates the quantity of the existing code"""
    if product_name in inventory:
        price,_=inventory[product_name]
        inventory[product_name]=(price,new_quantity)
        print(f"Quantity of'{product_name}'updated to{new_quantity}.")
    else:
        print("Product not found")
def list_products():
    """List all products and their details"""
print("Current Inventory")
for product,details in inventory.items():
     price,quantity=details
print(f"-{product}:Price=${price},Quantity={quantity}")
add_product("Monitor",300,15)
```

update_quantity("Laptop",8)

list_products()

OUTPUT:

```
-Keyboard:Price=$75,Quantity=20
Product'Monitor'added.
Quantity of'Laptop'updated to8.
```