PROGRAM AND OUTPUT :

TASK – 1 [BEGGINER LEVEL]

# 1.1

pi = 22 / 7

print("Value stored in pi : ",pi)

print("Type of the variable : ",type(pi))

Value stored in pi : 3.142857142857143

Type of the variable : <class 'float'>

# 1.2

for = 4

for = 4

^

SyntaxError: invalid syntax

# 1.3

p\_a = 10000

r\_i = 5

t = 3

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

print("Simple Interest:", simple\_interest)

Simple Interest: 1500.0

# 2.1

def format\_number(num, fmt):

    result = format(num, fmt)

    print(result)

    return result

format\_number(145, 'o')

221

# 2.2

r = 84

pi = 3.14

water\_per\_sq\_m = 1.4

area = pi \* (r \*\* 2)

t\_w = area \* water\_per\_sq\_m

print("Pond Area:", area)

print("Total Water (liters):", int(t\_w))

Pond Area: 22155.84

Total Water (liters): 31018

# 2.3

dist = 490

t\_m = 7

t\_s = t\_m \* 60

speed = dist / t\_s

print("Speed (m/s):", int(speed))

Speed (m/s): 1

# 3

justice\_league = ["Superman", "Batman", "Wonder Woman", "Flash", "Aquaman", "Green Lantern"]

print("Number of members:", len(justice\_league))

justice\_league.extend(["Batgirl", "Nightwing"])

print("After recruiting Batgirl & Nightwing:", justice\_league)

justice\_league.remove("Wonder Woman")

justice\_league.insert(0, "Wonder Woman")

print("Wonder Woman as leader:", justice\_league)

justice\_league.remove("Superman")  # remove original Superman position

flash = justice\_league.index("Flash")

aquaman = justice\_league.index("Aquaman")

if aquaman > flash:

    aquaman, flash = flash, aquaman

justice\_league.insert(aquaman + 1, "Superman")

print("Superman between Aquaman & Flash:", justice\_league)

justice\_league = ["Cyborg", "Shazam", "Hawkgirl", "Martian Manhunter", "Green Arrow"]

print("New team assembled by Superman:", justice\_league)

justice\_league.sort()

print("Alphabetically sorted team:", justice\_league)

print("New leader:", justice\_league[0])

Number of members: 6

After recruiting Batgirl & Nightwing: ['Superman', 'Batman', 'Wonder Woman', 'Flash', 'Aquaman', 'Green Lantern', 'Batgirl', 'Nightwing']

Wonder Woman as leader: ['Wonder Woman', 'Superman', 'Batman', 'Flash', 'Aquaman', 'Green Lantern', 'Batgirl', 'Nightwing']

Superman between Aquaman & Flash: ['Wonder Woman', 'Batman', 'Flash', 'Superman', 'Aquaman', 'Green Lantern', 'Batgirl', 'Nightwing']

New team assembled by Superman: ['Cyborg', 'Shazam', 'Hawkgirl', 'Martian Manhunter', 'Green Arrow']

Alphabetically sorted team: ['Cyborg', 'Green Arrow', 'Hawkgirl', 'Martian Manhunter', 'Shazam']

New leader: Cyborg

# 4.1

height = float(input("Enter height in meters: "))

weight = float(input("Enter weight in kilograms: "))

bmi = weight / (height \*\* 2)

if bmi >= 30:

    print("Obesity")

elif 25 <= bmi < 30:

    print("Overweight")

elif 18.5 <= bmi < 25:

    print("Normal")

else:

    print("Underweight")

Enter height in meters: 158

Enter weight in kilograms: 75

Underweight

# 4.2

AUSTRALIA = ["Sydney", "Melbourne", "Brisbane", "Perth"]

UAE = ["Dubai", "Abu Dhabi", "Sharjah", "Ajman"]

INDIA = ["Mumbai", "Bangalore", "Chennai", "Delhi"]

city = input("Enter a city name: ").strip()

if city in AUSTRALIA:

    print(f"{city} is in Australia")

elif city in UAE:

    print(f"{city} is in UAE")

elif city in INDIA:

    print(f"{city} is in India")

else:

    print(f"Sorry, I don't know where {city} is.")

Enter a city name: Mumbai

Mumbai is in India

# 4.3

AUSTRALIA = ["Sydney", "Melbourne", "Brisbane", "Perth"]

UAE = ["Dubai", "Abu Dhabi", "Sharjah", "Ajman"]

INDIA = ["Mumbai", "Bangalore", "Chennai", "Delhi"]

def find\_country(city):

    if city in AUSTRALIA:

        return "Australia"

    elif city in UAE:

        return "UAE"

    elif city in INDIA:

        return "India"

    else:

        return None

c1 = input("Enter the first city: ").strip()

c2 = input("Enter the second city: ").strip()

country1 = find\_country(c1)

country2 = find\_country(c2)

if country1 and country2:

    if country1 == country2:

        print(f"Both cities are in {country1}")

    else:

        print("They don't belong to the same country")

else:

    print("One or both cities are not in the list.")

Enter the first city: Mumbai

Enter the second city: Dubai

They don't belong to the same country

# 5.1

import random

rolls = 20

count\_6 = 0

count\_1 = 0

count\_two\_6s\_in\_row = 0

previous\_roll = None

for i in range(rolls):

    roll = random.randint(1, 6)

    print(f"Roll {i+1}: {roll}")

    if roll == 6:

        count\_6 += 1

    if roll == 1:

        count\_1 += 1

    if previous\_roll == 6 and roll == 6:

        count\_two\_6s\_in\_row += 1

    previous\_roll = roll

print("\n--- Statistics ---")

print("Times rolled 6:", count\_6)

print("Times rolled 1:", count\_1)

print("Times rolled two 6s in a row:", count\_two\_6s\_in\_row)

Roll 1: 2

Roll 2: 6

Roll 3: 2

Roll 4: 5

Roll 5: 5

Roll 6: 6

Roll 7: 5

Roll 8: 4

Roll 9: 1

Roll 10: 2

Roll 11: 1

Roll 12: 1

Roll 13: 1

Roll 14: 4

Roll 15: 5

Roll 16: 4

Roll 17: 4

Roll 18: 2

Roll 19: 2

Roll 20: 2

--- Statistics ---

Times rolled 6: 2

Times rolled 1: 4

Times rolled two 6s in a row: 0

# 5.2

total\_jumping\_jacks = 100

set\_size = 10

completed = 0

for i in range(0, total\_jumping\_jacks, set\_size):

    completed += set\_size

    print(f"You completed {completed} jumping jacks.")

    if completed >= total\_jumping\_jacks:

        print("Congratulations! You completed the workout.")

        break

    tired = input("Are you tired? (yes/y or no/n): ").strip().lower()

    if tired in ["yes", "y"]:

        skip = input("Do you want to skip the remaining sets? (yes/y or no/n): ").strip().lower()

        if skip in ["yes", "y"]:

            print(f"You completed a total of {completed} jumping jacks.")

            break

        else:

            remaining = total\_jumping\_jacks - completed

            print(f"{remaining} jumping jacks remaining.")

    else:

        remaining = total\_jumping\_jacks - completed

        print(f"{remaining} jumping jacks remaining.")

You completed 10 jumping jacks.

You completed a total of 10 jumping jacks.

# 6.1

# List of friends

friends = ["Aditya", "Rahul", "Sneha", "Priya", "Vikram"]

# Create list of tuples (name, length of name)

name\_lengths = [(name, len(name)) for name in friends]

print("Friends with name lengths:")

for item in name\_lengths:

    print(item)

Friends with name lengths:

('Aditya', 6)

('Rahul', 5)

('Sneha', 5)

('Priya', 5)

('Vikram', 6)

# 6.2

# Your expenses

your\_expenses = {

    "Hotel": 1200,

    "Food": 800,

    "Transportation": 500,

    "Attractions": 300,

    "Miscellaneous": 200

}

# Partner's expenses

partner\_expenses = {

    "Hotel": 1000,

    "Food": 900,

    "Transportation": 600,

    "Attractions": 400,

    "Miscellaneous": 150

}

# Calculate totals

your\_total = sum(your\_expenses.values())

partner\_total = sum(partner\_expenses.values())

print(f"Your total expenses: {your\_total}")

print(f"Partner's total expenses: {partner\_total}")

# Who spent more

if your\_total > partner\_total:

    print("You spent more overall.")

elif partner\_total > your\_total:

    print("Your partner spent more overall.")

else:

    print("Both spent the same amount.")

# Find category with significant difference

max\_difference = 0

diff\_category = ""

for category in your\_expenses:

    difference = abs(your\_expenses[category] - partner\_expenses[category])

    if difference > max\_difference:

        max\_difference = difference

        diff\_category = category

print(f"Biggest spending difference is in '{diff\_category}' category: {max\_difference}")

Your total expenses: 3000

Partner's total expenses: 3050

Your partner spent more overall.

Biggest spending difference is in 'Hotel' category: 200

# 7

import pandas as pd

data = {

    "Name": ["Alice", "Bob", "Charlie", "David", "Eve"],

    "Math": [85, 70, 95, 60, 88],

    "Science": [90, 88, 92, 75, 84],

    "English": [78, 82, 89, 70, 91]

}

df = pd.DataFrame(data)

df.to\_csv("student\_marks.csv", index=False)

df = pd.read\_csv("student\_marks.csv")

marks\_columns = [col for col in df.columns if col != "Name"]

df["total\_marks"] = df[marks\_columns].sum(axis=1)

df["average"] = df[marks\_columns].mean(axis=1)

df.to\_csv("student\_marks\_with\_totals.csv", index=False)

print("✅ CSV files created successfully!")

print(df)

✅ CSV files created successfully!

Name Math Science English total\_marks average

0 Alice 85 90 78 253 84.333333

1 Bob 70 88 82 240 80.000000

2 Charlie 95 92 89 276 92.000000

3 David 60 75 70 205 68.333333

4 Eve 88 84 91 263 87.666667

# 8

class Avenger:

    def \_\_init\_\_(self, name, age, gender, super\_power, weapon, leader=False):

        self.name = name

        self.age = age

        self.gender = gender

        self.super\_power = super\_power

        self.weapon = weapon

        self.leader = leader

    def get\_info(self):

        """Display information about the Avenger."""

        print(f"Name: {self.name}")

        print(f"Age: {self.age}")

        print(f"Gender: {self.gender}")

        print(f"Super Power: {self.super\_power}")

        print(f"Weapon: {self.weapon}")

        print("-" \* 40)

    def is\_leader(self):

        """Check if Avenger is a leader."""

        if self.leader:

            print(f"{self.name} is the leader of the Avengers.")

        else:

            print(f"{self.name} is not the leader of the Avengers.")

# Superheroes data

super\_heroes = [

    ("Captain America", 100, "Male", "Super strength", "Shield", True),

    ("Iron Man", 48, "Male", "Technology", "Armor", False),

    ("Black Widow", 35, "Female", "Superhuman", "Batons", False),

    ("Hulk", 49, "Male", "Unlimited Strength", "No Weapon", False),

    ("Thor", 1500, "Male", "Super Energy", "Mjölnir", False),

    ("Hawkeye", 41, "Male", "Fighting Skills", "Bow and Arrows", False)

]

# Create Avenger objects

avengers\_team = [Avenger(\*hero) for hero in super\_heroes]

# Display info & check leadership

for avenger in avengers\_team:

    avenger.get\_info()

    avenger.is\_leader()

Name: Captain America

Age: 100

Gender: Male

Super Power: Super strength

Weapon: Shield

----------------------------------------

Captain America is the leader of the Avengers.

Name: Iron Man

Age: 48

Gender: Male

Super Power: Technology

Weapon: Armor

----------------------------------------

Iron Man is not the leader of the Avengers.

Name: Black Widow

Age: 35

Gender: Female

Super Power: Superhuman

Weapon: Batons

----------------------------------------

Black Widow is not the leader of the Avengers.

Name: Hulk

Age: 49

Gender: Male

Super Power: Unlimited Strength

...

Super Power: Fighting Skills

Weapon: Bow and Arrows

----------------------------------------

Hawkeye is not the leader of the Avengers.

# 9

# Base Class

class MobilePhone:

    def \_\_init\_\_(self, screen\_type, network\_type, dual\_sim, front\_camera, rear\_camera, ram, storage):

        self.screen\_type = screen\_type

        self.network\_type = network\_type

        self.dual\_sim = dual\_sim

        self.front\_camera = front\_camera

        self.rear\_camera = rear\_camera

        self.ram = ram

        self.storage = storage

    def make\_call(self, number):

        print(f"Calling {number}...")

    def receive\_call(self):

        print("Incoming call...")

    def take\_a\_picture(self, camera\_type="rear"):

        if camera\_type.lower() == "rear":

            print(f"Taking picture with {self.rear\_camera} rear camera.")

        else:

            print(f"Taking selfie with {self.front\_camera} front camera.")

# Child Class - Apple

class Apple(MobilePhone):

    def \_\_init\_\_(self, model, screen\_type, network\_type, dual\_sim, front\_camera, rear\_camera, ram, storage):

        super().\_\_init\_\_(screen\_type, network\_type, dual\_sim, front\_camera, rear\_camera, ram, storage)

        self.model = model

    def show\_details(self):

        print(f"Apple {self.model}: {self.ram} RAM, {self.storage} storage, {self.network\_type} network")

# Child Class - Samsung

class Samsung(MobilePhone):

    def \_\_init\_\_(self, model, screen\_type, network\_type, dual\_sim, front\_camera, rear\_camera, ram, storage):

        super().\_\_init\_\_(screen\_type, network\_type, dual\_sim, front\_camera, rear\_camera, ram, storage)

        self.model = model

    def show\_details(self):

        print(f"Samsung {self.model}: {self.ram} RAM, {self.storage} storage, {self.network\_type} network")

# Creating Apple objects

iphone13 = Apple("iPhone 13", "Touch Screen", "5G", False, "12MP", "48MP", "4GB", "128GB")

iphoneSE = Apple("iPhone SE", "Touch Screen", "4G", False, "8MP", "12MP", "3GB", "64GB")

# Creating Samsung objects

samsungS21 = Samsung("Galaxy S21", "Touch Screen", "5G", True, "16MP", "48MP", "4GB", "128GB")

samsungA50 = Samsung("Galaxy A50", "Touch Screen", "4G", True, "12MP", "32MP", "3GB", "64GB")

# Example Usage

iphone13.show\_details()

iphone13.make\_call("9876543210")

iphone13.take\_a\_picture("front")

samsungS21.show\_details()

samsungS21.receive\_call()

samsungS21.take\_a\_picture()

Apple iPhone 13: 4GB RAM, 128GB storage, 5G network

Calling 9876543210...

Taking selfie with 12MP front camera.

Samsung Galaxy S21: 4GB RAM, 128GB storage, 5G network

Incoming call...

Taking picture with 48MP rear camera.

TASK – 2 [INTERMEDIATE LEVEL]

# 1

import requests

from bs4 import BeautifulSoup

def scrape\_python\_blogs(limit=5):

    url = "https://www.python.org/blogs/"

    response = requests.get(url, timeout=5)

    if response.status\_code != 200:

        return f"❌ Failed to retrieve webpage. Status: {response.status\_code}"

    soup = BeautifulSoup(response.text, 'html.parser')

    blogs = soup.select("ul.list-recent-posts li a")

    print("📌 Latest Python Blogs:\n")

    for blog in blogs[:limit]:

        title = blog.get\_text(strip=True)

        link = blog['href']

        if not link.startswith("http"):

            link = "https://www.python.org" + link

        print(f"{title} -> {link}")

scrape\_python\_blogs()

Unmasking Phantom Dependencies with Software Bill-of-Materials as Ecosystem Neutral Metadata -> <https://pyfound.blogspot.com/2025/08/unmasking-phantom-dependencies-with.html>

Python 3.13.6 is now available -> <https://pythoninsider.blogspot.com/2025/08/python-3136-is-now-available.html>

The PSF has paused our Grants Program -> <https://pyfound.blogspot.com/2025/08/the-psf-has-paused-our-grants-program.html>

PSF Board Election Nominations Opening July 29th -> <https://pyfound.blogspot.com/2025/07/psf-board-nominations-opening-july-29th.html>

Python 3.14 release candidate 1 is go! -> <https://pythoninsider.blogspot.com/2025/07/python-314-release-candidate-1-is-go.html>

# 2

import random

import sys

def hangman():

    words\_with\_hints = {

        "python": "A popular programming language 🐍",

        "java": "A widely used programming language ☕",

        "computer": "An electronic device for processing data 💻",

        "internet": "A global system of interconnected computers 🌐",

        "hangman": "A classic word guessing game 🎯"

    }

    word, hint = random.choice(list(words\_with\_hints.items()))

    guessed = set()

    attempts = 6

    print("🎯 Welcome to Hangman!")

    print(f"💡 Hint: {hint}")

    while attempts > 0:

        display = [letter if letter in guessed else "\_" for letter in word]

        print(" ".join(display))

        sys.stdout.flush()  # Force output before waiting for input

        if "\_" not in display:

            print("🏆 You guessed it! The word was:", word)

            break

        guess = input("Guess a letter: ").lower()

        if not guess.isalpha() or len(guess) != 1:

            print("❌ Please enter a single letter.")

            continue

        if guess in guessed:

            print("⚠️ You already guessed that letter.")

            continue

        guessed.add(guess)

        if guess not in word:

            attempts -= 1

            print(f"❌ Wrong! {attempts} attempts left.")

    else:

        print(f"💀 Game Over! The word was: {word}")

hangman()

🎯 Welcome to Hangman!

💡 Hint: An electronic device for processing data 💻

\_ \_ \_ \_ \_ \_ \_ \_

❌ Wrong! 5 attempts left.

\_ \_ \_ \_ \_ \_ \_ \_

c \_ \_ \_ \_ \_ \_ \_

c o \_ \_ \_ \_ \_ \_

c o m \_ \_ \_ \_ \_

c o m p \_ \_ \_ \_

c o m p u \_ \_ \_

c o m p u t \_ \_

c o m p u t e \_

c o m p u t e r

🏆 You guessed it! The word was: computer