Task Codes Till Intermediate

Student Name: Parbinder Singh

University: Central University of Jammu

Major: BTech (Computer Science and Engineering)

Internship Duration: 1st July 2025 - 31st July 2025

Company: ShadowFox

Domain: Python Programming

Mentor: Mr. Hariharan

Coordinator: Mr. Aakash

Task 1:

Variables: -

#Variable Task 1

pi = 22 / 7 print("Value of pi:", pi) print("Data type of pi:", type(pi))

#Variable Task 2

1.2 Create a variable named 'for' and assign it 4

for = 4 # This will show a SyntaxError

Reason:

'for' is a reserved keyword in Python used for loops, hence cannot be used as a variable name.

#Variable Task 3

1P = 1000 # Principal amount R = 5 # Rate of interest T = 3 # Time in years

SI = (P * R * T) / 100 print("Simple Interest for 3 years is:", SI)

Numbers: -

#Numbers Task 1

num = 145 char = 'o' formatted = "{} is formatted as {:o}".format(num, num) print(formatted)

#Numbers Task 2

radius = 84 pi = 3.14 area = pi * (radius ** 2) water_per_m2 = 1.4 total_water = int(area * water_per_m2)

print("Pond area:", area) print("Total water in liters (no decimal):", total_water)

#Numbers Task 3

distance = 490 # meters time_minutes = 7 time_seconds = time_minutes * 60

speed = int(distance / time_seconds) print("Speed in meters per second (no decimal):", speed)

Llsts: -

justice_league = ["Superman", "Batman", "Wonder Woman", "Flash", "Aquaman", "Green Lantern"] print("1. Original:", justice_league)

Add Batgirl and Nightwing

justice_league.extend(["Batgirl", "Nightwing"]) print("2. After recruitment:", justice_league)

Move Wonder Woman to front

justice_league.remove("Wonder Woman") justice_league.insert(0, "Wonder Woman") print("3. Wonder Woman is leader:", justice_league)

Separate Aquaman and Flash with Superman

justice_league.remove("Superman") index_flash = justice_league.index("Flash") justice_league.insert(index_flash, "Superman") print("4. Superman between Aquaman and Flash:", justice_league)

Replace the list

justice_league = ["Cyborg", "Shazam", "Hawkgirl", "Martian Manhunter", "Green Arrow"] print("5. New team:", justice_league)

Sort and show new leader

justice_league.sort() print("6. Sorted team:", justice_league) print("New Leader:", justice_league[0])

If Condition: -

4.1 BMI Category

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")

4.2 City to country

city = input("Enter a city name: ") Australia = ["Sydney", "Melbourne", "Brisbane", "Perth"] UAE = ["Dubai", "Abu Dhabi", "Sharjah", "Ajman"] India = ["Mumbai", "Bangalore", "Chennai", "Delhi"] 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("City not found")

4.3 Check if two cities are in same country

city1 = input("Enter the first city: ") city2 = input("Enter the second city: ") def get_country(city): if city in Australia: return "Australia" elif city in UAE: return "UAE" elif city in India: return "India" return None country1 = get_country(city1) country2 = get_country(city2) if country1 and country1 == country2: print(f'Both cities are in {country1}") else: print("They don't belong to the same country")

For Loop: -

5.1 Dice roll simulation

import random rolls = [random.randint(1, 6) for _ in range(20)] count_6 = rolls.count(6) count_1 = rolls.count(1)Count two 6s in a row

count_double_6 = 0 for i in range(len(rolls) - 1): if rolls[i] == 6 and rolls[i+1] == 6: count_double_6 += 1 print("Rolls:", rolls) print("Number of 6s:", count_6) print("Number of 1s:", count_1) print("Two 6s in a row:", count_double_6)

5.2 Jumping jacks tracker

completed = 0 while completed < 100: completed += 10 print(f"You have completed {completed} jumping jacks.") tired = input("Are you tired? (yes/no): ").lower() if tired in ["yes", "y"]: skip = input("Do you want to skip the remaining sets? (yes/no): ").lower() if skip in ["yes", "y"]: print(f"You completed a total of {completed} jumping jacks.") break else: print("Congratulations! You completed the workout")

Dictionary: -

6.1 Friends name and length

friends = ["Amit", "Rohit", "Priya", "Sneha", "Karan"] friend_lengths = [(name, len(name)) for name in friends] print("Friend name and length:", friend_lengths)

6.2 Trip expenses comparison

your_expenses = { "Hotel": 1200, "Food": 800, "Transportation": 500, "Attractions": 300, "Miscellaneous": 200 } partner_expenses = { "Hotel": 1000, "Food": 900, "Transportation": 600, "Attractions": 400, "Miscellaneous": 150 } total_your = sum(your_expenses.values()) total_partner = sum(partner_expenses.values()) print("Your total expenses:", total_your) print("Partner's total expenses:", total_partner)

if total_your > total_partner: print("You spent more.") else: print("Your partner spent more.")

Find significant expense difference

for category in your_expenses: diff = abs(your_expenses[category] - partner_expenses[category]) if diff > 100: print(f"Significant difference in {category}: ₹{diff}")

File handling: -

```
import csv

# 7.1 Read existing CSV file and create dictionary (demo sample)
# Let's assume we have a list of student records:
students_data = [
    {"Name": "Amit", "Math": 80, "Science": 70, "English": 90},
    {"Name": "Priya", "Math": 85, "Science": 75, "English": 95},
    {"Name": "Rohit", "Math": 78, "Science": 80, "English": 82},
]

# Add total and average fields
for student in students_data:
    total = student["Math"] + student["Science"] + student["English"]
    average = total / 3
    student["Total"] = total
    student["Average"] = round(average, 2)

# 7.2 Write new data to file
with open("student_results.csv", "w", newline="") as file:
    fieldnames = ["Name", "Math", "Science", "English", "Total", "Average"]
    writer = csv.DictWriter(file, fieldnames=fieldnames)
```

```
    writer.writeheader()
    for student in students_data:
        writer.writerow(student)

print("Student results written to student_results.csv")
```

Classes and Objects: -

class Avenger: def init(self, name, age, gender, power, weapon): self.name = name self.age = age self.gender = gender self.power = power self.weapon = weapon

```
def get_info(self):
    return f"{self.name} ({self.gender}, {self.age}) - Power: {self.power}, Weapon:
{self.weapon}"

def is_leader(self):
    return self.name == "Captain America"
```

Creating Avengers

super_heroes = [Avenger("Captain America", 100, "Male", "Super strength", "Shield"), Avenger("Iron Man", 45, "Male", "Technology", "Armor"), Avenger("Black Widow", 35, "Female", "Superhuman", "Batons"), Avenger("Hulk", 40, "Male", "Unlimited Strength", "None"), Avenger("Thor", 1500, "Male", "Super Energy", "Mjölnir"), Avenger("Hawkeye", 38, "Male", "Fighting Skills", "Bow and Arrows"),]

for hero in super_heroes: print(hero.get_info()) if hero.is_leader(): print("-> This Avenger is the leader")

Inheritance: -

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):
    print("Making a call...")

def receive_call(self):
    print("Receiving a call...")

def take_picture(self):
    print(f"Taking a picture with {self.rear_camera} rear camera")
```

class Apple(MobilePhone): def init(self, model, *args): super().init(*args) self.model = model

class Samsung(MobilePhone): def init(self, model, *args): super().init(*args) self.model = model

Create Apple and Samsung objects

iphone = Apple("iPhone 14", "Touch Screen", "5G", False, "12MP", "48MP", "4GB", "64GB") s22 = Samsung("Galaxy S22", "Touch Screen", "5G", True, "10MP", "32MP", "4GB", "128GB")

iphone.make_call() s22.take_picture()

Task 2:

Web Scraper: -

import requests from bs4 import BeautifulSoup import csv

Target URL (Example ShadowFox-like website)

url = "https://shadowfox.in/"

try: response = requests.get(url, timeout=10) response.raise_for_status() # Raise HTTPError for bad status soup = BeautifulSoup(response.text, 'html.parser')

```
# Example: Extract all links and their texts
links = soup.find_all('a')
extracted_data = []

for link in links:
    text = link.get_text(strip=True)
    href = link.get('href')
    if text and href:
        extracted_data.append({"text": text, "link": href})

# Save data to CSV
with open("shadowfox_links.csv", "w", newline="", encoding="utf-8") as file:
    writer = csv.DictWriter(file, fieldnames=["text", "link"])
    writer.writeheader()
    writer.writerows(extracted_data)

print(" Scraping completed and data saved to shadowfox_links.csv")
```

except requests.exceptions.RequestException as e: print("Error during request:", e) except Exception as e: print("General error:", e)

Hangman: -

```
import random

def get_random_word(): words = ["python", "shadow", "developer", "hangman", "internship", "scraper", "algorithm"] return random.choice(words)

def display_hangman(tries): stages = [ """ -------- | | | O | |/ | | | / - """, """ -------- | | | O | |/ | | | - """, """ -------- | | | O | | | | |
```

- """, """ -------- | | | O | | | | |

- """, """ -------- | | | O |

|

|

- """, """ -------- | | |

|

|

|

- """] return stages[tries]

def play_game(): word = get_random_word() word_letters = set(word) guessed_letters = set()
incorrect_guesses = 0 max_tries = 6

print("Welcome to Hangman!\n")

while incorrect_guesses < max_tries and word_letters:
    print(display_hangman(incorrect_guesses))
    print("Word:", ' '.join([letter if letter in guessed_letters else '_' for letter in word]))
    print("Guessed letters:", ' '.join(sorted(guessed_letters)))
    guess = input("\nGuess a letter: ").lower()
    if not guess.isalpha() or len(guess) != 1:
        print("Please enter a single valid alphabet letter.")
        continue
    if guess in guessed_letters:
        print("You already guessed that letter.")
        continue
    guessed_letters.add(guess)
    if guess in word_letters:
        word_letters.remove(guess)
        print(f"Good job! '{guess}' is in the word.")
    else:
        incorrect_guesses += 1
        print(f"Sorry! '{guess}' is not in the word. Attempts left: {max_tries - incorrect_guesses}")

# Game conclusion
if not word_letters:

```
    print("\nCongratulations! You guessed the word:", word)
else:
    print(display_hangman(incorrect_guesses))
    print("\nGame Over! The correct word was:", word)

# Play again option
again = input("\nDo you want to play again? (yes/no): ").lower()
if again in ['yes', 'y']:
    play_game()
 Start the game

play_game()
```