Python Development Internship Task List

Task Level (Beginner): Do 5 out of 9

1. Variables

1. Create a variable named pi and store the value 22/7 in it.

Now check the data type of this variable.

Code:

```
pi=22/7
print(f"Value of pi:{pi}")
print(f"Data type of pi:{type(pi)}")
```

Output:

```
Value of pi:3.142857142857143
Data type of pi:<class 'float'>
```

2. Create a variable called for and assign it a value 4. See what happens and find out the reason behind the behavior that you see.

Code:

```
for_value=4
print(f"Value of the variable 'for_value':{for_value}")
```

Output:

```
Value of the variable 'for_value':4
```

3. Store the principal amount, rate of interest, and time in different variables and then calculate the Simple Interest for 3 years. Formula: Simple Interest = P x R x T / 100

Code:

```
principal=10000
rate_of_interest=5
time_period=3
simple_interest=(principal*rate_of_interest*time_period)/100
print(f"Simple Interest for {time_period}years:{simple_interest}")
```

Output:

Simple Interest for 3years:1500.0

2. Numbers

1. Write a function that takes two arguments, 145 and 'o', and uses the `format` function to return a formatted string. Print the result. Try to identify the representation used.

Code:

```
def format_as_octal(number):
    return format(number,'o')
result=format_as_octal(145)
print("Octal representation of 145 is:",result)
```

Output: Octal representation of 145 is: 221

2. In a village, there is a circular pond with a radius of 84 meters. Calculate the area of the pond using the formula: Circle Area = π r^2. (Use the value 3.14 for π) Bonus Question: If there is exactly 1.4 liters of water in a square meter, what is the total amount of water in the pond? Print the answer without any decimal point in it. Hint: Circle Area = π r^2 Water in the pond = Pond Area Water per Square Meter

Code:

```
def pond_area_and_water(radius):
    pi=3.14
    area=pi*radius**2
    water_per_sq_meter=1.4
    total_water=int(area*water_per_sq_meter)
    return total_water
total_water_in_pond=pond_area_and_water(84)
print("Total water in the pond(liters):",total_water_in_pond)
```

Output:

```
Total water in the pond(liters): 31018
```

3. If you cross a 490meterlong street in 7 minutes, calculate your speed in meters per second. Print the answer without any decimal point in it. Hint: Speed = Distance / Time

Code:

```
def calculate_speed(distance,time_minutes):
    time_seconds=time_minutes*60
    speed=int(distance/time_seconds)
    return speed
speed=calculate_speed(490,7)
print("Speed in meters per second:",speed)
```

Output:

```
Speed in meters per second: 1
```

3. List

1. You have a list of superheroes representing the Justice League. justice_league = ["Superman","Batman","Wonder Woman","Flash","Aquaman","Green Lantern"]

Perform the following tasks:

1. Calculate the number of members in the Justice League.

2. Batman recruited Batgirl and Nightwing as new members. Add them to your list.

3. Wonder Woman is now the leader of the Justice League. Move her to the beginning of the list.

4. Aquaman and Flash are having conflicts, and you need to separate them. Choose either "Green Lantern" or "Superman" and move them in between Aquaman and Flash.

5. The Justice League faced a crisis, and Superman decided to

assemble a new team. Replace the existing list with the following

new members: "Cyborg","Shazam","Hawkgirl","Martian Manhunter","Green Arrow".

6. Sort the Justice League alphabetically. The hero at the 0th

index will become the new leader.

(BONUS: Can you predict who the new leader will be?)

Your task is to write Python code to perform these operations on

the "justice_league" list. Display the list at each step to observe the changes.

Code:

```
justice_league = ["Superman", "Batman", "Wonder Woman", "Flash", "Aquaman", "Green Lantern"]

num_members = len(justice_league)

print(f"1. Number of Justice League members: {num_members}")

print(f"Justice League: {justice_league}\n")

justice_league.append("Batgirl")

justice_league.append("Nightwing")

print(f"2. After adding Batgirl and Nightwing: {justice_league}\n")

justice_league.remove("Wonder Woman")

justice_league.insert(0, "Wonder Woman")

print(f"3. Wonder Woman is now the leader: {justice_league}\n")

justice_league.remove("Green Lantern")

flash_index = justice_league.index("Flash")

justice_league.insert(flash_index, "Green Lantern")

print(f"4. After moving Green Lantern between Aquaman and Flash: {justice_league}\n")
```

```
justice_league = ["Cyborg", "Shazam", "Hawkgirl", "Martian Manhunter", "Green Arrow"]

print(f"5. New team after Superman assembled a new Justice League: {justice_league}\n")

justice_league.sort()

new_leader = justice_league[0]

print(f"6. Justice League sorted alphabetically: {justice_league}")

print(f"New leader: {new_leader}")
```

Output:

```
1. Number of Justice League members: 6

Justice League: ['Superman', 'Batman', 'Wonder Woman', 'Flash', 'Aquaman', 'Green Lantern']


2. After adding Batgirl and Nightwing: ['Superman', 'Batman', 'Wonder Woman', 'Flash', 'Aquaman', 'Green Lantern', 'Batgirl', 'Nightwing']


3. Wonder Woman is now the leader: ['Wonder Woman', 'Superman', 'Batman', 'Flash', 'Aquaman', 'Green Lantern', 'Batgirl', 'Nightwing']


4. After moving Green Lantern between Aquaman and Flash: ['Wonder Woman', 'Superman', 'Batman', 'Green Lantern', 'Flash', 'Aquaman', 'Batgirl', 'Nightwing']


5. New team after Superman assembled a new Justice League: ['Cyborg', 'Shazam', 'Hawkgirl', 'Martian Manhunter', 'Green Arrow']


6. Justice League sorted alphabetically: ['Cyborg', 'Green Arrow', 'Hawkgirl', 'Martian Manhunter', 'Shazam']
```

New leader: Cyborg

4. If Condition

1. Write a program to determine the BMI Category based on user input.

Ask the user to:

Enter height in meters

Enter weight in kilograms

Calculate BMI using the formula: BMI = weight / (height)2

Use the following categories:

If BMI is 30 or greater, print "Obesity"

If BMI is between 25 and 29, print "Overweight"

If BMI is between 18.5 and 25, print "Normal"

If BMI is less than 18.5, print "Underweight"

Example:

Enter height in meters: 1.75

Enter weight in kilograms: 70

Output: "Normal"

Code:

```
height = float(input("Enter height in meters:"))
weight = float(input("Enter weight in kilograms:"))
bmi = weight / (height ** 2)
if bmi >= 30:
    category = "Obesity"
elif 25 <= bmi < 30:
```

```
    category = "Overweight"
elif 18.5 <= bmi < 25:
    category = "Normal"
else:
    category = "Underweight"
print(f"BMI: {bmi:.2f} - {category}")
```

Output :

```
Enter height in meters:5
Enter weight in kilograms:60
BMI: 2.40 - Underweight
```

2. Write a program to determine which country a city belongs to. Given list of cities per country:

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

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

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

Ask the user to enter a city name and print the corresponding country.

Example:

Enter a city name: "Abu Dhabi"

Output: "Abu Dhabi is in UAE"

Code:

```
austraila = ["Sydney", "Melbourne", "Brisbane", "Perth"]
uae  = ["Dubai","Abu Dhabi", "Sharjah", "Ajman"]
india = ["Mumabi", "Banglore", "Chennai", "Delhi"]
```

```
city = input("Enter a City name:")
if city in austraila:
    country = "Austraila"
elif city in uae:
    country = "UAE"
elif city in india:
    country = "India"
else:
    country = "Unknown"
if country != "Unknown":
    print(f"{city} is in {country}")
else:
    print(f"{city} does not match any country in our list")
```

Output :

```
Enter a City name:Banglore
Banglore is in India
```

3. Write a program to check if two cities belong to the same country.

Ask the user to enter two cities and print whether they belong to the same country or not.

Example:

Enter the first city: "Mumbai"

Enter the second city: "Chennai"

Output: "Both cities are in India"

Example:

Enter the first city: "Sydney"

Enter the second city: "Dubai"

Output: "They don't belong to the same country

Code:

```
austraila = ["Sydney", "Melbourne", "Brisbane", "Perth"]
uae  = ["Dubai","Abu Dhabi", "Sharjah", "Ajman"]
india = ["Mumbai", "Banglore", "Chennai", "Delhi"]
city1 = input("Enter the first city:")
city2 = input("Enter the second city:")
def get_country(city):
    if city in austraila:
        return "Australia"
    elif city in uae:
        return "UAE"
    elif city in india:
        return "India"
    else:
        return "Unknown"
country1 = get_country(city1)
country2 = get_country(city2)
if country1 == country2 and country1 != "Unknown" :
    print(f"Both cities are in {country1}")
else:
```

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

Output:

Enter the first city:Mumbai

Enter the second city:Chennai

Both cities are in India

5. For Loop:

1. Using a for loop, simulate rolling a sixsided die multiple times (at least 20 times).

Count and print the following statistics:

How many times you rolled a 6

How many times you rolled a 1

How many times you rolled two 6s in a row

Code:

```
import random
total_rolls = 20
six_count = 0
one_count = 0
consecutive_six_count = 0
previous_roll = 0
for i in range(total_rolls):
    roll = random.randint(1,6)
    if roll == 6:
        six_count += 1
    if roll == 1:
```

```
        one_count +=1
    if roll == 6 and previous_roll ==6:
        consecutive_six_count += 1
    previous_roll = roll
print(f"Total rolls: {total_rolls}")
print(f"Number of 6s rolled: {six_count}")
print(f"Number of 1s rolled: {one_count}")
print(f"Number of consecutive 6s rolled: {consecutive_six_count}")
```

Output:

```
Total rolls: 20
Number of 6s rolled: 4
Number of 1s rolled: 4
Number of consecutive 6s rolled: 0
```

2. Imagine you are doing a workout routine, and you have to complete 100 jumping jacks.

Write a program that:

Asks you to perform 10 jumping jacks at a time.

After each set, it asks, "Are you tired?"

If you reply "yes" or "y," it should ask if you want to skip the remaining sets.

If you reply "yes" or "y," it should break and print, "You completed a total of jumping jacks."

For example, if you did only 30 jumping jacks and answered "yes," the programwill break and print, "You completed a total of 30 jumping jacks."

If you reply "no" or "n," it should continue and display how many jumping jacks are remaining. After that, ask you again, "Are you tired?"

For example, if you answered "no," it should display that 70 jumping jacks are remaining and ask you again, "Are you tired?"

If you reply "no" or "n," it should continue and display how many jumping jacks are remaining. After that, ask you again, "Are you tired?"

For example, if you answered "no," it should display that 70 jumping jacks are remaining and ask you again, "Are you tired?"

If you complete all 100 jumping jacks, it should print, "Congratulations! You completed the workout," and stop the program

Code:

```
total_jumping_jacks = 100
set_size = 10
completed_jacks = 0
while completed_jacks < total_jumping_jacks:
    completed_jacks += set_size
    remaining_jacks = total_jumping_jacks - completed_jacks
    tired = input(f"Are you tired after completing {completed_jacks} jumping jacks? (yes/y or no/n): ").lower()
    if tired in ['yes', 'y']:
        skip = input("Do you want to skip the remaining sets? (yes/y or no/n):").lower()
        if skip in ['yes', 'y']:
            print(f"You completed a total of {completed_jacks} jumping jacks.")
            break
```

```
    if completed_jacks >= total_jumping_jacks:
        print("Congratulations! You Completed the workout.")
    else:
        print(f"{remaining_jacks} jumping jacks remaining.")
```

Output:

```
Are you tired after completing 10 jumping jacks? (yes/y or no/n): no
90 jumping jacks remaining.
Are you tired after completing 20 jumping jacks? (yes/y or no/n): no
80 jumping jacks remaining.
Are you tired after completing 30 jumping jacks? (yes/y or no/n): no
70 jumping jacks remaining.
Are you tired after completing 40 jumping jacks? (yes/y or no/n): no
60 jumping jacks remaining.
Are you tired after completing 50 jumping jacks? (yes/y or no/n): no
50 jumping jacks remaining.
Are you tired after completing 60 jumping jacks? (yes/y or no/n): no
40 jumping jacks remaining.
Are you tired after completing 70 jumping jacks? (yes/y or no/n): no
30 jumping jacks remaining.
Are you tired after completing 80 jumping jacks? (yes/y or no/n): no
20 jumping jacks remaining.
Are you tired after completing 90 jumping jacks? (yes/y or no/n): no
10 jumping jacks remaining.
Are you tired after completing 100 jumping jacks? (yes/y or no/n): no
```

Congratulations! You Completed the workout.

6. Dictionary

1. Create a list of your friends' names. The list should have at least 5 names. Create a list of tuples. Each tuple should contain a friend's name and the length of the name.

For example, if someone's name is Aditya, the tuple would be: ('Aditya', 6)

Code:

```
friends = ['Ramesh','Vinay','Surendra','Praveen','Altaf']
friends_tuples = [(friend,len(friend)) for friend in friends]
print("List of friends and their name lenghts:")
for friend_tuple in friends_tuples:
    print(friend_tuple)
```

Output:

```
List of friends and their name lenghts:
('Ramesh', 6)
('Vinay', 5)
('Surendra', 8)
('Praveen', 7)
('Altaf', 5)
```

2.You and your partner are planning a trip, and you want to track expenses. Create two dictionaries, one for your expenses and one for your partner's expenses. Each dictionary should contain at least 5 expense categories and their corresponding amounts.

For example:

Your expenses

your_expenses = {

"Hotel": 1200,

"Food": 800,

"Transportation": 500,

"Attractions": 300,

"Miscellaneous": 200

}

Your partner's expenses

partner_expenses = {

"Hotel": 1000,

"Food": 900,

"Transportation": 600,

"Attractions": 400,

"Miscellaneous": 150

}

Calculate the total expenses for each of you and print the results.

Determine who spent more money overall and print the result.

Find out the expense category where there is a significant difference in spending between you and your partner.

Print the category and the difference.

Code:

```
your_expenses = {
    "Hotel": 1200,
    "Food": 2000,
    "Transportation": 5000,
    "Attractions": 3000,
    "Miscellaneous": 2000
}
partner_expenses = {
    "Hotel": 1000,
    "Food": 1150,
    "Transportation": 6000,
    "Attractions": 4000,
    "Miscellaneous": 1500
}
def calculate_total(expenses):
    return sum(expenses.values())
def find_significant_difference(exp1, exp2):
    max_diff = 0
    diff_category = ""
    for category in exp1:
        diff = abs(exp1[category] - exp2[category])
        if diff > max_diff:
            max_diff = diff
```

```
            diff_category = category
    return diff_category, max_diff
your_total = calculate_total(your_expenses)
partner_total = calculate_total(partner_expenses)
print(f"Your total expenses: ₹{your_total}")
print(f"Partner's total expenses: ₹{partner_total}")
if your_total > partner_total:
    print("You spent more than your partner.")
elif your_total < partner_total:
    print("Your partner spent more than you.")
else:
    print("Both spent the same amount.")
category, difference = find_significant_difference(your_expenses, partner_expenses)
print(f"The category with the biggest difference is '{category}' with a difference of ₹{difference}.")
combined_total = your_total + partner_total
equal_share = combined_total / 2
print(f"\nCombined total expenses: ₹{combined_total}")
print(f"Each person's share: ₹{equal_share}")
if your_total > equal_share:
    print(f"You paid more. Your partner owes you ₹{your_total - equal_share}.")
elif partner_total > equal_share:
    print(f"Your partner paid more. You owe your partner ₹{equal_share - your_total}.")
else:
```

```
print("Both of you have already paid an equal share.")
```

OUTPUT:

```
Your total expenses: ₹13200

Partner's total expenses: ₹13650

Your partner spent more than you.

The category with the biggest difference is 'Transportation' with a difference of ₹1000.

Combined total expenses: ₹26850

Each person's share: ₹13425.0

Your partner paid more. You owe your partner ₹225.0.
```

8. Classes and Objects:

1.Avengers is a Marvel's American Superheroes team, Now you want to showcase your programming skills by representing the Avengers team using classes. Create a class called Avenger and create these six superheroes using this class.

2. super_heroes = ["Captain America", "Iron Man", "Black Widow", "Hulk", "Thor", "Hawkeye"]

3. Your Avenger class should have these properties:

1. Name

2. Age

3. Gender

4. Super Power

5. Weapon

4. Captain America has Super strength, Iron Man has Technology, Black Widow is superhuman, Hulk has Unlimited Strength, Thor has super Energy and Hawkeye has fighting skills as superpowers.

5. Weapons: Shield, Armor, Batons, No Weapon for hulk, Mjölnir, Bow, and Arrows

6. Create methods to get the information about each superhero

7. Create a method is_leader() which will tell if the superhero is a leader or not.

Code:

```
class Avenger:
    def __init__(self, name, age, gender, super_power, weapon):
        self.name = name
        self.age = age
        self.gender = gender
        self.super_power = super_power
        self.weapon = weapon
    def get_info(self):
        return (f"Name: {self.name}\n"
                f"Age: {self.age}\n"
                f"Gender: {self.gender}\n"
                f"Super Power: {self.super_power}\n"
                f"Weapon: {self.weapon}")
    def is_leader(self):
        return self.name == "Captain America"
super_heroes = [
    Avenger("Captain America", 100, "Male", "Super Strength", "Shield"),
    Avenger("Iron Man", 48, "Male", "Technology", "Armor"),
    Avenger("Black Widow", 35, "Female", "Superhuman", "Batons"),
```

```
    Avenger("Hulk", 50, "Male", "Unlimited Strength", "No Weapon"),
    Avenger("Thor", 1000, "Male", "Super Energy", "Mjölnir"),
    Avenger("Hawkeye", 45, "Male", "Fighting Skills", "Bow and Arrows")
]
for hero in super_heroes:
    print(hero.get_info())
    print("Is Leader:", hero.is_leader())
    print("-" * 40)
```

OUTPUT:

```
Name: Captain America
Age: 100
Gender: Male
Super Power: Super Strength
Weapon: Shield
Is Leader: True
----------------------------------------
Name: Iron Man
Age: 48
Gender: Male
Super Power: Technology
Weapon: Armor
Is Leader: False
----------------------------------------
Name: Black Widow
```

Age: 35

Gender: Female

Super Power: Superhuman

Weapon: Batons

Is Leader: False

Name: Hulk

Age: 50

Gender: Male

Super Power: Unlimited Strength

Weapon: No Weapon

Is Leader: False

Name: Thor

Age: 1000

Gender: Male

Super Power: Super Energy

Weapon: Mjölnir

Is Leader: False

Name: Hawkeye

Age: 45

Gender: Male

Super Power: Fighting Skills

Weapon: Bow and Arrows

Is Leader: False

---

9. Inheritence

1. Create inheritance using MobilePhone as base class and Apple & Samsung as child class

1. The base class should have properties:

1. ScreenType = Touch Screen

2. NetworkType = 4G/5G

3. DualSim = True or False

4. FrontCamera = (5MP/8MP/12MP/16MP)

5. rearCamera = (8MP/12MP/16MP/32MP/48MP)

6. RAM = (2GB/3GB/4GB)

7. Storage = (16GB/32GB/64GB)

2. Create basic mobile phone functionalities in the classes like: make_call, recieve_call, take_a_picture, etc.

3. Use super() constructor for calling parent class's constructor

4. Make some objects of Apple class with different properties

5. Make some objects of Samsung class with different properties

CODE:

```
class MobilePhone:
    """
    Base class representing a mobile phone with common features.
    """
    def __init__(self, screen_type: str = "Touch Screen",
                 network_type: str = "4G/5G",
                 dual_sim: bool = False,
                 front_camera: str = "12MP",
                 rear_camera: str = "48MP",
                 ram: str = "4GB",
                 storage: str = "64GB",
                 battery_capacity: int = 4000,
                 charging_speed: int = 25,
                 operating_system: str = "Generic OS",
                 price: float = 999.99):
        """
        Initialize the MobilePhone with common properties.
        """
        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
        self.battery_capacity = battery_capacity
        self.charging_speed = charging_speed
        self.operating_system = operating_system
        self.price = price
    def make_call(self, number: str) -> None:
        """
        Simulate making a call to the given number.
        """
        print(f"Calling {number} from {self.operating_system} phone...")
    def receive_call(self, caller: str) -> None:
        """
        Simulate receiving a call from the given caller.
        """
        print(f"Receiving a call from {caller} on {self.operating_system} phone...")
    def take_a_picture(self, camera_type: str = "rear") -> None:
        """
        Simulate taking a picture with either the front or rear camera.
        """
        if camera_type.lower() == "front":
            print(f"Taking a picture with the {self.front_camera} front camera.")
        else:
```

```
        print(f"Taking a picture with the {self.rear_camera} rear camera.")
    def phone_details(self) -> dict:
        """
        Return the phone's details as a dictionary.
        """
        return {
            "Screen Type": self.screen_type,
            "Network Type": self.network_type,
            "Dual SIM": self.dual_sim,
            "Front Camera": self.front_camera,
            "Rear Camera": self.rear_camera,
            "RAM": self.ram,
            "Storage": self.storage,
            "Battery Capacity (mAh)": self.battery_capacity,
            "Charging Speed (Watts)": self.charging_speed,
            "Operating System": self.operating_system,
            "Price (USD)": f"${self.price:.2f}"
        }
    def __str__(self) -> str:
        """
        String representation of the mobile phone.
        """
        details = self.phone_details()
        return "\n".join(f"{key}: {value}" for key, value in details.items())
```

```
class Apple(MobilePhone):
    """
    Class representing an Apple mobile phone.
    """
    def __init__(self, front_camera: str = "12MP",
                 rear_camera: str = "48MP",
                 ram: str = "4GB",
                 storage: str = "64GB",
                 battery_capacity: int = 3000,
                 charging_speed: int = 20,
                 price: float = 1199.99):
        """
        Initialize Apple phone with specific properties.
        """
        super().__init__(dual_sim=False, front_camera=front_camera,
                         rear_camera=rear_camera, ram=ram, storage=storage,
                         battery_capacity=battery_capacity,
                         charging_speed=charging_speed,
                         operating_system="iOS", price=price)
class Samsung(MobilePhone):
    """
    Class representing a Samsung mobile phone.
    """
    def __init__(self, dual_sim: bool = True,
```

```
            front_camera: str = "16MP",
            rear_camera: str = "32MP",
            ram: str = "6GB",
            storage: str = "128GB",
            battery_capacity: int = 5000,
            charging_speed: int = 45,
            price: float = 1099.99):
        """
        Initialize Samsung phone with specific properties.
        """
        super().__init__(dual_sim=dual_sim, front_camera=front_camera,
                         rear_camera=rear_camera, ram=ram, storage=storage,
                         battery_capacity=battery_capacity,
                         charging_speed=charging_speed,
                         operating_system="Android", price=price)
if __name__ == "__main__":
    iphone_12 = Apple(front_camera="12MP", rear_camera="48MP", ram="4GB", storage="64GB", price=1099.99)
    iphone_13_pro = Apple(front_camera="12MP", rear_camera="48MP", ram="6GB", storage="128GB", price=1299.99)
    samsung_galaxy_s21 = Samsung(dual_sim=True, front_camera="16MP", rear_camera="32MP", ram="8GB", storage="256GB", price=1199.99)
    samsung_galaxy_note = Samsung(dual_sim=False, front_camera="10MP", rear_camera="64MP", ram="12GB", storage="512GB", price=1499.99)
    print("Apple iPhone 12 Details:\n", iphone_12)
```

```
print("\nApple iPhone 13 Pro Details:\n", iphone_13_pro)
print("\nSamsung Galaxy S21 Details:\n", samsung_galaxy_s21)
print("\nSamsung Galaxy Note Details:\n", samsung_galaxy_note)
print("\n--- Using functionalities ---")
iphone_12.make_call("801-972-0900")
samsung_galaxy_s21.receive_call("Mom")
iphone_13_pro.take_a_picture("front")
samsung_galaxy_note.take_a_picture("rear")
```

OUTPUT :

```
Apple iPhone 12 Details:
 Screen Type: Touch Screen
Network Type: 4G/5G
Dual SIM: False
Front Camera: 12MP
Rear Camera: 48MP
RAM: 4GB
Storage: 64GB
Battery Capacity (mAh): 3000
Charging Speed (Watts): 20
Operating System: iOS
Price (USD): $1099.99
```

Apple iPhone 13 Pro Details:

Screen Type: Touch Screen

Network Type: 4G/5G

Dual SIM: False

Front Camera: 12MP

Rear Camera: 48MP

RAM: 6GB

Storage: 128GB

Battery Capacity (mAh): 3000

Charging Speed (Watts): 20

Operating System: iOS

Price (USD): $1299.99

Samsung Galaxy S21 Details:

Screen Type: Touch Screen

Network Type: 4G/5G

Dual SIM: True

Front Camera: 16MP

Rear Camera: 32MP

RAM: 8GB

Storage: 256GB

Battery Capacity (mAh): 5000

Charging Speed (Watts): 45

Operating System: Android

Price (USD): $1199.99

Samsung Galaxy Note Details:

Screen Type: Touch Screen

Network Type: 4G/5G

Dual SIM: False

Front Camera: 10MP

Rear Camera: 64MP

RAM: 12GB

Storage: 512GB

Battery Capacity (mAh): 5000

Charging Speed (Watts): 45

Operating System: Android

Price (USD): $1499.99

--- Using functionalities ---

Calling 801-972-0900 from iOS phone...

Receiving a call from Mom on Android phone...

Taking a picture with the 12MP front camera.

Taking a picture with the 64MP rear camera.

END POINT

Name: **D.Vishnu Vardhan Reddy**

Institute: **Parul University**

Internship Development:**Python Development**