# Assignment 1

**PROBLEM:**

Python Program to display name, college name, and other messages.

**SOLUTION:**

```python
def personal_details():
    name = "Simon"
    age = 19
    address = "Bangalore, Karnataka, India"

    print(f"Name: {name}\nAge: {age}\nAddress: {address}")

personal_details()
```

**OUTPUT:**

```
Name: Simon
Age: 19
Address: Bangalore, Karnataka, India
```

**PROBLEM:**

Python Program using type() function to display different basic data types in Python.

**SOLUTION:**

```python
num = 3.14
word = "Hello, World!"
flag = True

print("Type of num:", type(num))
print("Type of word:", type(word))
print("Type of flag:", type(flag))

my_tuple = (10, 'Hello', 45, 'Hi')
my_dict = {1: 'One', 2: 'Two', 3: 'Three'}

if type(my_tuple) is not type(my_dict):
    print("The variables have different object types.")
else:
    print("The variables have the same object type.")
```

**OUTPUT:**

```
Type of num: <class 'float'>
Type of word: <class 'str'>
Type of flag: <class 'bool'>
The variables have different object types.
```

# Assignment 2

**PROBLEM:**

Python Program to input two numbers then find the larger/smaller number.

**SOLUTION:**

```
1  num1 = float(input("Enter the first number: "))
2  num2 = float(input("Enter the second number: "))
3
4  if num1 > num2:
5      print(f"{num1} is larger than {num2}.")
6  elif num1 < num2:
7      print(f"{num2} is larger than {num1}.")
8  else:
9      print("Both numbers are equal.")
```

**OUTPUT:**

```
Enter the first number: 55
Enter the second number: 53
55.0 is larger than 53.0.
```

**PROBLEM:**

Python Program to determine ODD or EVEN number.

**SOLUTION:**

```python
def check_odd_or_even(number):
    if number % 2 == 0:
        return "Even"
    else:
        return "Odd"

user_input = int(input("Enter a number: "))
result = check_odd_or_even(user_input)
print(f"The number {user_input} is {result}.")
```

**OUTPUT:**

```
Enter a number: 55
The number 55 is Odd.
```

# Assignment 3

**PROBLEM:**

Python Program to print *n* numbers of prime number.

**SOLUTION:**

```python
def print_n_primes(n):
    count, num = 0, 2
    while count < n:
        if is_prime(num):
            print(num, end=" ")
            count += 1
        num += 1
def is_prime(num):
        if num <= 1:
            return False
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                return False
        return True

n = int(input("Enter the value of n: "))
print(f"The first {n} prime numbers are:")
print_n_primes(n)
```

**OUTPUT:**

```
Enter the value of n: 5
The first 5 prime numbers are:
2 3 5 7 11
```

**PROBLEM:**

Python Program to input three numbers and find the largest and smallest number.

**SOLUTION:**

```python
def find_largest_and_smallest(num1, num2, num3):
    largest = max(num1, num2, num3)
    smallest = min(num1, num2, num3)
    return largest, smallest

num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

largest_num, smallest_num = find_largest_and_smallest(num1, num2, num3)

print(f"The largest number is {largest_num}.")
print(f"The smallest number is {smallest_num}.")
```

**OUTPUT:**

```
Enter the first number: 59
Enter the second number: 69
Enter the third number: 79
The largest number is 79.0.
The smallest number is 59.0.
```

# Assignment 4

**PROBLEM:**

Python Program to determine Armstrong number

**SOLUTION:**

```python
def is_armstrong(number):
    num_str = str(number)
    num_digits = len(num_str)
    digit_sum = sum(int(digit) ** num_digits for digit in num_str)
    return digit_sum == number

user_input = int(input("Enter a number: "))
if user_input < 0:
    print("Please enter a non-negative integer.")
elif is_armstrong(user_input):
    print(f"{user_input} is an Armstrong number.")
else:
    print(f"{user_input} is not an Armstrong number.")
```

**OUTPUT:**

```
Enter a number: 152
152 is not an Armstrong number.
```

**PROBLEM:**

Python Program to determine palindrome number.

**SOLUTION:**

```python
def is_palindrome(number):
    num_str = str(number)
    return num_str == num_str[::-1]

user_input = int(input("Enter a number: "))
if user_input < 0:
    print("Please enter a non-negative integer.")
elif is_palindrome(user_input):
    print(f"{user_input} is a palindrome number.")
else:
    print(f"{user_input} is not a palindrome number.")
```

**OUTPUT:**

```
Enter a number: 123321
123321 is a palindrome number.
```

**PROBLEM:**

Python Program to reverse words in a given String in Python.

**SOLUTION:**

```python
def reverse_words(input_string):
    words = input_string.split()

    reversed_words = words[::-1]
    reversed_string = ' '.join(reversed_words)
    return reversed_string

input_string = "Hello World"
reversed_string = reverse_words(input_string)
print("Original String:", input_string)
print("Reversed Words String:", reversed_string)
```

**OUTPUT:**

```
Original String: Hello World
Reversed Words String: World Hello
```

# Assignment 5

**PROBLEM:**

Python Program using 5 string function in python.

**SOLUTION:**

```python
def string_functions_demo(input_string):
    words = input_string.split()
    print("Split String into Words:", words)

    joined_string = '-'.join(words)
    print("Joined String with Hyphens:", joined_string)

    replaced_string = input_string.replace('World', 'Universe')
    print("String after Replacement:", replaced_string)

    upper_string = input_string.upper()
    print("Uppercase String:", upper_string)


# Example usage
input_string = "  Hello World  "
string_functions_demo(input_string)
```

**OUTPUT:**

```
Split String into Words: ['Hello', 'World']
Joined String with Hyphens: Hello-World
String after Replacement:   Hello Universe
Uppercase String:   HELLO WORLD
```

**PROBLEM:**

Python Program to display the terms of a Fibonacci series.

**SOLUTION:**

```
def fibonacci_series(n):
    a, b = 0, 1

    for i in range(n):
        print(a, end=" ")
        a, b = b, a + b

num_of_terms = 10
print(f"Fibonacci series up to {num_of_terms} terms:")
fibonacci_series(num_of_terms)
```

**OUTPUT:**

```
Fibonacci series up to 10 terms:
0 1 1 2 3 5 8 13 21 34
```

# Assignment 6

**PROBLEM:**

Python Program to find largest & smallest number in a tuple.

**SOLUTION:**

```python
def find_largest_and_smallest(numbers):
    largest = max(numbers)
    smallest = min(numbers)
    return largest, smallest

# Example usage
numbers = (3, 41, 52, 26, 38, 57, 9, 49)
largest, smallest = find_largest_and_smallest(numbers)
print("Largest number:", largest)
print("Smallest number:", smallest)
```

**OUTPUT:**

```
Largest number: 57
Smallest number: 3
```

**PROBLEM:**

Python Program to find largest & smallest number in a list

**SOLUTION:**

```python
def find_largest_and_smallest(numbers):
    largest = numbers[0]
    smallest = numbers[0]

    for number in numbers:
        if number > largest:
            largest = number
        if number < smallest:
            smallest = number

    return largest, smallest

numbers = [3, 41, 52, 26, 38, 57, 9, 49]
largest, smallest = find_largest_and_smallest(numbers)
print("Largest number:", largest)
print("Smallest number:", smallest)
```

**OUTPUT:**

```
Largest number: 57
Smallest number: 3
```

# Assignment 7

**PROBLEM:**

Python Program to Sort Python Dictionaries by Key or Value.

**SOLUTION:**

```
myDict = {'ravi': 10, 'rajnish': 9,
          'sanjeev': 15, 'yash': 2, 'suraj': 32}

myKeys = list(myDict.keys())
myValue = list(myDict.items())
myValue.sort(key=lambda item: item[1])
myKeys.sort()
sorted_dict_key = {i: myDict[i] for i in myKeys}
sorted_dict_value = {i[0]: i[1] for i in myValue}

print(sorted_dict_key)
print(sorted_dict_value)
```

**OUTPUT:**

```
{'rajnish': 9, 'ravi': 10, 'sanjeev': 15, 'suraj': 32, 'yash': 2}
{'yash': 2, 'rajnish': 9, 'ravi': 10, 'sanjeev': 15, 'suraj': 32}
```

**PROBLEM:**

Python Program to Merging two Dictionaries.

**SOLUTION:**

```python
def merge_dicts(dict1, dict2):
    merged_dict = dict1.copy()
    merged_dict.update(dict2)
    return merged_dict

# Example usage
dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}
merged_dict = merge_dicts(dict1, dict2)
print("Merged Dictionary:", merged_dict)
```

**OUTPUT:**

```
Merged Dictionary: {'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

# Assignment 8

**PROBLEM:**

Python Program to perform guess the number generated by Random function.

**SOLUTION:**

```python
import random

def guess_the_number():
    secret_number = random.randint(1, 100)

    print("I have chosen a number between 1 and 100. Can you guess it?")

    attempts = 0
    while True:
        guess = int(input("Enter your guess (between 1 and 100): "))
        attempts += 1

        if guess < secret_number:
            print("Too low! Try again.")
        elif guess > secret_number:
            print("Too high! Try again.")
        else:
            print(f"Congratulations! You guessed the number {secret_number}
                correctly!")
            print(f"It took you {attempts} attempts.")
            break

guess_the_number()
```

**OUTPUT:**

```
I have chosen a number between 1 and 100. Can you guess it?
Enter your guess (between 1 and 100): 50
Too low! Try again.
Enter your guess (between 1 and 100): 75
Too low! Try again.
Enter your guess (between 1 and 100): 80
Too low! Try again.
Enter your guess (between 1 and 100): 90
Too high! Try again.
Enter your guess (between 1 and 100): 85
Too low! Try again.
Enter your guess (between 1 and 100): 87
Too low! Try again.
Enter your guess (between 1 and 100): 88
Too low! Try again.
Enter your guess (between 1 and 100): 89
Congratulations! You guessed the number 89 correctly!
It took you 8 attempts.
```

# Assignment 9

**PROBLEM:**

Python Program to perform Single Inheritance in python.

**SOLUTION:**

```python
class Parent:
    def __init__(self, name):
        self.name = name

    def show_info(self):
        print("Name:", self.name)

class Child(Parent):
    def __init__(self, name, age):
        super().__init__(name)
        self.age = age

    def show_info(self):
        super().show_info()
        print("Age:", self.age)

parent_obj = Parent("ParentName")
child_obj = Child("ChildName", 10)
parent_obj.show_info()
child_obj.show_info()
```

**OUTPUT:**

```
Name: ParentName
Name: ChildName
Age: 10
```

# Assignment 10

**PROBLEM:**

Python Program to perform Multiple Inheritance in python.

**SOLUTION:**

```
class Parent1:
    def method1(self):
        print("Method 1 from Parent 1")

class Parent2:
    def method2(self):
        print("Method 2 from Parent 2")

class Child(Parent1, Parent2):
    def method3(self):
        print("Method 3 from Child")

child_obj = Child()

child_obj.method1()
child_obj.method2()
child_obj.method3()
```

**OUTPUT:**

```
Method 1 from Parent 1
Method 2 from Parent 2
Method 3 from Child
```

# Assignment 11

**PROBLEM:**

Python Program to perform Multilevel Inheritance in python.

**SOLUTION:**

```
class Grandparent:
    def method1(self):
        print("Method 1 from Grandparent")

class Parent(Grandparent):
    def method2(self):
        print("Method 2 from Parent")

class Child(Parent):
    def method3(self):
        print("Method 3 from Child")

child_obj = Child()
child_obj.method1()
child_obj.method2()
child_obj.method3()
```

**OUTPUT:**

```
Method 1 from Grandparent
Method 2 from Parent
Method 3 from Child
```

# Assignment 12

**PROBLEM:**

Python Program to perform Hierarchical Inheritance in python.

**SOLUTION:**

```
class Grandparent:
    def method1(self):
        print("Method 1 from Grandparent")

class Parent(Grandparent):
    def method2(self):
        print("Method 2 from Parent")

class Child(Grandparent):
    def method3(self):
        print("Method 3 from Child")

child_obj = Child()
parent_obj = Parent()

child_obj.method1()
child_obj.method3()

parent_obj.method1()
parent_obj.method2()
```

**OUTPUT:**

```
Method 1 from Grandparent
Method 3 from Child
Method 1 from Grandparent
Method 2 from Parent
```

# Assignment 13

**PROBLEM:**

Python Program to perform Hybrid Inheritance in python.

**SOLUTION:**

```
class A:
    def method_A(self):
        print("Method A from class A")

class B(A):
    def method_B(self):
        print("Method B from class B")

class C(A):
    def method_C(self):
        print("Method C from class C")

class D(B, C):
    def method_D(self):
        print("Method D from class D")

d_obj = D()
d_obj.method_A()
d_obj.method_B()
d_obj.method_C()
d_obj.method_D()
```

**OUTPUT:**

```
Method A from class A
Method B from class B
Method C from class C
Method D from class D
```

# Assignment 14

**PROBLEM:**

Python Program to create class/objects in Python and perform movie ticket booking.

**SOLUTION:**

```python
class MovieTicket:
    def __init__(self, movie_name, theater_name, show_time, num_tickets):
        self.movie_name = movie_name
        self.theater_name = theater_name
        self.show_time = show_time
        self.num_tickets = num_tickets

    def book_tickets(self, num_tickets):
        if num_tickets <= 0:
            print("Number of tickets should be greater than zero.")
            return
        if num_tickets <= self.num_tickets:
            print(f"Successfully booked {num_tickets} ticket(s) for the movie {self
                .movie_name}.")
            self.num_tickets -= num_tickets
        else:
            print("Sorry, tickets are not available.")

    def show_details(self):
        print("Movie:", self.movie_name)
        print("Theater:", self.theater_name)
        print("Show Time:", self.show_time)
        print("Available Tickets:", self.num_tickets)

movie_ticket = MovieTicket("Avengers: Endgame", "Regal Cinemas", "7:00 PM", 100)
movie_ticket.show_details()
movie_ticket.book_tickets(3)
movie_ticket.show_details()
```

**OUTPUT:**

```
Movie: Avengers: Endgame
Theater: Regal Cinemas
Show Time: 7:00 PM
Available Tickets: 100
Successfully booked 3 ticket(s) for the movie Avengers: Endgame.
Movie: Avengers: Endgame
Theater: Regal Cinemas
Show Time: 7:00 PM
Available Tickets: 97
```

**PROBLEM:**

Python Program to work with class constructors and other elements of OOP in Python.

**SOLUTION:**

```python
class Car:
    num_cars = 0
    def __init__(self, brand, model, year, color):
        self.brand = brand
        self.model = model
        self.year = year
        self.color = color
        Car.num_cars += 1

    def display_info(self):
        print("Brand:", self.brand)
        print("Model:", self.model)
        print("Year:", self.year)
        print("Color:", self.color)

    @staticmethod
    def total_cars():
        print("Total cars:", Car.num_cars)

car1 = Car("Toyota", "Camry", 2018, "Red")
car2 = Car("Honda", "Civic", 2020, "Blue")

print("Car 1 Information:")
car1.display_info()
print("\nCar 2 Information:")
car2.display_info()
Car.total_cars()
```

**OUTPUT:**

```
Car 1 Information:
Brand: Toyota
Model: Camry
Year: 2018
Color: Red
```

```
Car 2 Information:
Brand: Honda
Model: Civic
Year: 2020
Color: Blue
Total cars: 2
```

# Assignment 15

**PROBLEM:**

Analyzing Sales Data with Python using panda, numpy and matplotlib Library.

**SOLUTION:**

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Read sales data from CSV file
sales_data = pd.read_csv('sales_data.csv')

# Calculate total sales per month
sales_data['Date'] = pd.to_datetime(sales_data['Date'])
sales_data['Month'] = sales_data['Date'].dt.month
monthly_sales = sales_data.groupby('Month')['Sales'].sum()

# Plot the monthly sales data
plt.plot(monthly_sales.index, monthly_sales.values, marker='o')
plt.title('Monthly Sales')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct'
    , 'Nov', 'Dec'])
plt.grid(True)
plt.show()
```

**OUTPUT:**