Python Slip Question Answers BBA - CA

Slip No - 1

✓ A) Python program to accept n numbers in a list and remove duplicates:

```
# Function to accept n numbers and remove duplicates
def remove_duplicates():
    # Accept the number of elements in the list
    n = int(input("Enter the number of elements in the list: "))

    # Initialize an empty list
    num_list = []

    # Accept n numbers and add them to the list
    for i in range(n):
        num = int(input(f"Enter number {i + 1}: "))
        num_list.append(num)

    # Remove duplicates by converting the list to a set and back to a list
    unique_list = list(set(num_list))

    # Print the list without duplicates
    print("List without duplicates:", unique_list)

# Call the function
remove_duplicates()
```

B) Python GUI program to take birthdate and output age:

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime

# Function to calculate age
def calculate_age():
    # Get the entered birthdate
    birthdate_str = entry.get()

    try:
        # Convert the string to a datetime object
        birthdate = datetime.strptime(birthdate_str, "%Y-%m-%d")
        today = datetime.today()
        # Calculate the age
        age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))

        # Show the age in a message box
        messagebox.showinfo("Age", f"Your age is {age} years.")

    except ValueError:
        messagebox.showerror("Invalid input", "Please enter the date in YYYY-MM-DD format.")

# Create the main window
root = tk.Tk()
root.title("Age Calculator")

# Create a label and an entry widget for the birthdate input
label = tk.Label(root, text="Enter your birthdate (YYYY-MM-DD):")
label.pack(pady=10)
entry = tk.Entry(root)
entry.pack(pady=10)

# Create a button to calculate the age
button = tk.Button(root, text="Calculate Age", command=calculate_age)
button.pack(pady=10)

# Run the main event loop
root.mainloop()
```

Slip No - 2

A) Python function to calculate the number of upper case and lower case letters:

```
def count_case(string):
    upper_case = 0
    lower_case = 0
    for char in string:
        if char.isupper():
            upper_case += 1
        elif char.islower():
            lower_case += 1
    print(f"No. of Upper case characters: {upper_case}")
    print(f"No. of Lower case characters: {lower_case}")

# Sample usage
count_case('The quick Brown Fox')
```

B) Python GUI program to create a digital clock with Tkinter:

```
import tkinter as tk
from time import strftime

def time():
    current_time = strftime('%H:%M:%S %p')
    label.config(text=current_time)
    label.after(1000, time)

root = tk.Tk()
root.title('Digital Clock')

label = tk.Label(root, font=('calibri', 40, 'bold'), background='purple', foreground='white')
label.pack(anchor='center')

time()

root.mainloop()
```

Slip No - 3

A) Python program to check if a given key exists in a dictionary and replace it:

```
def check_and_replace_key(dictionary, old_key, new_key, new_value):
    if old_key in dictionary:
        dictionary.pop(old_key)
        dictionary[new_key] = new_value
    print(dictionary)

# Sample usage
my_dict = {'a': 1, 'b': 2, 'c': 3}
check_and_replace_key(my_dict, 'b', 'd', 4)
```

B) Python script to define a Student class and a Test subclass:

```
class Student:
    def __init__(self, roll_no, name, age, gender):
        self.roll_no = roll_no
        self.name = name
        self.age = age
        self.gender = gender

class Test(Student):
    def __init__(self, roll_no, name, age, gender, marks):
        super().__init__(roll_no, name, age, gender)
        self.marks = marks

    def total_marks(self):
        return sum(self.marks)

    def display(self):
        print(f"Roll No: {self.roll_no}")
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Gender: {self.gender}")
        print(f"Marks: {self.marks}")
        print(f"Total Marks: {self.total_marks()}")

# Creating objects of Test class
student1 = Test(101, "Alice", 20, "Female", [85, 90, 88])
student2 = Test(102, "Bob", 21, "Male", [75, 80, 78])
student3 = Test(103, "Charlie", 19, "Male", [65, 70, 68])

# Displaying details
```

```
student1.display()
student2.display()
student3.display()
```

Slip No - 4

A) Python GUI program to create background with changing colors:

```
import tkinter as tk
import random

def change_bg_color():
    colors = ["red", "green", "blue", "yellow", "pink", "purple", "orange", "white"]
    window.config(bg=random.choice(colors))
    window.after(1000, change_bg_color)

window = tk.Tk()
window.title("Background Color Changer")
window.geometry("400x300")

change_bg_color()

window.mainloop()
```

B) Python script to define Employee and Manager classes:

```
class Employee:
    def __init__(self, emp_id, name, department, salary):
        self.emp_id = emp_id
        self.name = name
        self.department = department
        self.salary = salary

    def accept(self):
        self.emp_id = input("Enter employee ID: ")
        self.name = input("Enter employee name: ")
        self.department = input("Enter employee department: ")
        self.salary = float(input("Enter employee salary: "))

    def display(self):
        print(f"ID: {self.emp_id}")
        print(f"Name: {self.name}")
        print(f"Department: {self.department}")
        print(f"Salary: {self.salary}")

class Manager(Employee):
    def __init__(self, emp_id, name, department, salary, bonus=0):
        super().__init__(emp_id, name, department, salary)
        self.bonus = bonus

    def accept(self):
        super().accept()
        self.bonus = float(input("Enter manager bonus: "))

    def display(self):
        super().display()
        print(f"Bonus: {self.bonus}")
        print(f"Total Salary: {self.salary + self.bonus}")

# Creating n objects and finding manager with max total salary
n = int(input("Enter the number of managers: "))
managers = []

for i in range(n):
    print(f"\nEnter details for manager {i + 1}:")
    m = Manager("", "", "", 0)
    m.accept()
    managers.append(m)

# Find manager with maximum total salary
max_salary_manager = max(managers, key=lambda m: m.salary + m.bonus)

print("\nManager with maximum total salary:")
max_salary_manager.display()
```

Slip No -5

A) Python script using a class with methods get_String and print_String:

```
class StringManipulation:
    def __init__(self):
        self.user_string = ""

    def get_String(self):
        self.user_string = input("Enter a string: ")

    def print_String(self):
        print(self.user_string.upper())

# Creating an object of the class and using the methods
string_obj = StringManipulation()
string_obj.get_String()
string_obj.print_String()
```

B) Python script to generate Fibonacci terms using a generator function:
python

```
def fibonacci_generator():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

# Using the generator to generate Fibonacci terms
fib_gen = fibonacci_generator()
n = int(input("Enter the number of Fibonacci terms: "))
for _ in range(n):
    print(next(fib_gen))
```

Slip No -6

A) Python script using a package to calculate area and volume of cube and sphere:

Package structure:

markdown

```
geometry/
    __init__.py
    cube.py
    sphere.py
```

geometry/cube.py:

python

```
def area_of_cube(side):
    return 6 * (side ** 2)

def volume_of_cube(side):
    return side ** 3
```

geometry/sphere.py:

python

```
import math

def area_of_sphere(radius):
    return 4 * math.pi * (radius ** 2)

def volume_of_sphere(radius):
    return (4/3) * math.pi * (radius ** 3)
```

Main script:

python

```
from geometry.cube import area_of_cube, volume_of_cube
from geometry.sphere import area_of_sphere, volume_of_sphere

# Cube calculations
side = float(input("Enter the side length of the cube: "))
print(f"Area of Cube: {area_of_cube(side)}")
print(f"Volume of Cube: {volume_of_cube(side)}")
```

```
# Sphere calculations
radius = float(input("Enter the radius of the sphere: "))
print(f"Area of Sphere: {area_of_sphere(radius)}")
print(f"Volume of Sphere: {volume_of_sphere(radius)}")
```

B) Python GUI program to create a label and change the label font style using check buttons:

```
import tkinter as tk
from tkinter import font

def update_font():
    current_font = "Arial"
    font_size = 12
    font_weight = "normal"

    if bold_var.get() == 1:
        font_weight = "bold"

    if size_var.get() == 1:
        font_size = 20

    label.config(font=(current_font, font_size, font_weight))

root = tk.Tk()
root.title("Font Style Changer")

# Create a label
label = tk.Label(root, text="Sample Text", font=("Arial", 12))
label.pack(pady=20)

# Bold check button
bold_var = tk.IntVar()
bold_check = tk.Checkbutton(root, text="Bold", variable=bold_var, command=update_font)
bold_check.pack()

# Font size check button
size_var = tk.IntVar()
size_check = tk.Checkbutton(root, text="Increase Size", variable=size_var, command=update_font)
size_check.pack()

# Run the main loop
root.mainloop()
```

Slip No - 7

A) Python class to perform addition of two complex numbers using + operator overloading:

```
class ComplexNumber:
    def __init__(self, real, imag):
        self.real = real
        self.imag = imag

    # Overloading the + operator
    def __add__(self, other):
        return ComplexNumber(self.real + other.real, self.imag + other.imag)

    # String representation for easy printing
    def __str__(self):
        return f"{self.real} + {self.imag}i"

# Creating two complex number objects
c1 = ComplexNumber(2, 3)
c2 = ComplexNumber(4, 5)

# Adding the two complex numbers
result = c1 + c2
print("Sum of complex numbers:", result)
```

B) Python GUI program to generate a random password with upper and lower case letters:

```
import tkinter as tk
import random
import string

def generate_password():
    length = 8
    characters = string.ascii_letters  # Upper and lower case letters
    password = ''.join(random.choice(characters) for _ in range(length))
    label.config(text=f"Generated Password: {password}")

# Create the main window
root = tk.Tk()
root.title("Random Password Generator")

# Create a button to generate password
button = tk.Button(root, text="Generate Password", command=generate_password)
button.pack(pady=20)

# Create a label to display the generated password
```

```
label = tk.Label(root, text="")
label.pack(pady=20)

# Run the main loop
root.mainloop()
```

Slip No -8

A) Python script to find the repeated items of a tuple:

```
def find_repeated_items(tup):
    repeated_items = set([item for item in tup if tup.count(item) > 1])
    return repeated_items

# Sample tuple
my_tuple = (1, 2, 3, 2, 4, 5, 6, 3, 7, 8, 5)
repeated = find_repeated_items(my_tuple)
print("Repeated items:", repeated)
```

B) Python class with get_String, print_String, and reverse string functionality:
python

```
class StringManipulation:
    def __init__(self):
        self.user_string = ""

    def get_String(self):
        self.user_string = input("Enter a string: ")

    def print_String(self):
        print(self.user_string.upper())

    def reverse_and_print_lower(self):
        reversed_string = ' '.join(self.user_string.split()[::-1])
        print(reversed_string.lower())

# Creating an object and using the methods
string_obj = StringManipulation()
string_obj.get_String()
string_obj.print_String()  # Print in uppercase
string_obj.reverse_and_print_lower()  # Print reversed string in lowercase
```

Slip No - 9

A) Python script using class to reverse a string word by word:
python

```
class StringManipulation:
    def __init__(self):
        self.user_string = ""

    def get_String(self):
        self.user_string = input("Enter a string: ")

    def reverse_string(self):
        reversed_string = ' '.join(self.user_string.split()[::-1])
        print("Reversed string:", reversed_string)

# Creating an object and using the methods
string_obj = StringManipulation()
string_obj.get_String()
string_obj.reverse_string()
```

B) Python GUI program to accept a number and check if it is Prime, Perfect, or Armstrong:
python

```
import tkinter as tk
from tkinter import messagebox

# Function to check if the number is prime
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# Function to check if the number is perfect
def is_perfect(num):
    return num == sum(i for i in range(1, num) if num % i == 0)

# Function to check if the number is Armstrong
def is_armstrong(num):
    digits = [int(d) for d in str(num)]
    return num == sum(d ** len(digits) for d in digits)

# Function to handle button press
```

```
def check_number():
    try:
        n = int(entry.get())
        if var.get() == 1:
            result = "Prime" if is_prime(n) else "Not Prime"
        elif var.get() == 2:
            result = "Perfect" if is_perfect(n) else "Not Perfect"
        else:
            result = "Armstrong" if is_armstrong(n) else "Not Armstrong"
        messagebox.showinfo("Result", result)
    except ValueError:
        messagebox.showerror("Error", "Please enter a valid number")

# Creating the main window
root = tk.Tk()
root.title("Number Checker")

# Create an entry field to accept a number
entry_label = tk.Label(root, text="Enter a number:")
entry_label.pack(pady=10)
entry = tk.Entry(root)
entry.pack(pady=10)

# Create radio buttons for selecting Prime, Perfect, or Armstrong check
var = tk.IntVar()
prime_radio = tk.Radiobutton(root, text="Prime", variable=var, value=1)
prime_radio.pack()
perfect_radio = tk.Radiobutton(root, text="Perfect", variable=var, value=2)
perfect_radio.pack()
armstrong_radio = tk.Radiobutton(root, text="Armstrong", variable=var, value=3)
armstrong_radio.pack()

# Create a button to check the number
check_button = tk.Button(root, text="Check", command=check_number)
check_button.pack(pady=10)

# Run the main event loop
root.mainloop()
```

Slip No -10

A) Python GUI program to display an alert message when a button is pressed:
python

```
import tkinter as tk
from tkinter import messagebox

def show_alert():
    messagebox.showinfo("Alert", "Button was pressed!")

# Create the main window
root = tk.Tk()
root.title("Alert Message")

# Create a button that triggers the alert
button = tk.Button(root, text="Press Me", command=show_alert)
button.pack(pady=20)

# Run the main event loop
root.mainloop()
```

B) Python class to find the validity of a string of parentheses:
python

```
class ParenthesesValidator:
    def __init__(self):
        self.pairings = {')': '(', '}': '{', ']': '['}

    def is_valid(self, s):
        stack = []
        for char in s:
            if char in self.pairings.values():
                stack.append(char)
            elif char in self.pairings.keys():
                if not stack or stack.pop() != self.pairings[char]:
                    return False
        return len(stack) == 0

# Create an object of the ParenthesesValidator class
validator = ParenthesesValidator()

# Sample usage
test_strings = ["()", "()[]{}", "[)", "({[)]", "{{{"]
for string in test_strings:
    print(f"{string}: {validator.is_valid(string)}")
```

Slip No -11

A) Python program to compute element-wise sum of given tuples:
python

```
def elementwise_sum(tuples):
    return tuple(sum(values) for values in zip(*tuples))

# Original tuples
tuple1 = (1, 2, 3, 4)
tuple2 = (3, 5, 2, 1)
tuple3 = (2, 2, 3, 1)

# Compute element-wise sum
result = elementwise_sum((tuple1, tuple2, tuple3))
print("Element-wise sum of the said tuples:", result)
```

B) Python GUI program to add a menu bar to change the background color:
python

```
import tkinter as tk

def change_color(color):
    window.config(bg=color)

# Create the main window
window = tk.Tk()
window.title("Color Menu Example")

# Create a menu bar
menu_bar = tk.Menu(window)

# Create a submenu for colors
color_menu = tk.Menu(menu_bar, tearoff=0)
color_menu.add_command(label="Red", command=lambda: change_color("red"))
color_menu.add_command(label="Green", command=lambda: change_color("green"))
color_menu.add_command(label="Blue", command=lambda: change_color("blue"))
color_menu.add_command(label="Yellow", command=lambda: change_color("yellow"))
color_menu.add_command(label="White", command=lambda: change_color("white"))

# Add the color menu to the menu bar
menu_bar.add_cascade(label="Colors", menu=color_menu)

# Configure the main window to use the menu bar
```

```
window.config(menu=menu_bar)

# Run the main loop
window.mainloop()
```

Slip No - 12

A) Python GUI program to create a label and change the label font style using Tkinter:
python

```
import tkinter as tk

def update_font():
    font_name = font_name_var.get()
    font_size = font_size_var.get()
    font_weight = "bold" if bold_var.get() else "normal"

    label.config(font=(font_name, font_size, font_weight))

# Create the main window
root = tk.Tk()
root.title("Change Label Font Style")

# Create a label
label = tk.Label(root, text="Sample Text", font=("Arial", 12))
label.pack(pady=20)

# Font name selection
font_name_var = tk.StringVar(value="Arial")
font_name_menu = tk.OptionMenu(root, font_name_var, "Arial", "Helvetica", "Times", "Courier", command=lambda _: update_font())
font_name_menu.pack(pady=5)

# Font size selection
font_size_var = tk.IntVar(value=12)
font_size_scale = tk.Scale(root, from_=8, to=32, orient=tk.HORIZONTAL, variable=font_size_var, label="Font Size", command=lambda _: update_font())
font_size_scale.pack(pady=5)

# Bold checkbox
bold_var = tk.BooleanVar()
bold_check = tk.Checkbutton(root, text="Bold", variable=bold_var, command=update_font)
bold_check.pack(pady=5)

# Run the main loop
root.mainloop()
```

B) Python program to count repeated characters in a string:

python

```
from collections import Counter

def count_repeated_characters(s):
    # Count characters and filter only those that are repeated
    counts = Counter(s)
    repeated = {char: count for char, count in counts.items() if count > 1}
    return repeated

# Sample string
sample_string = 'thequickbrownfoxjumpsoverthelazydog'
repeated_characters = count_repeated_characters(sample_string)

# Display the output
for char, count in repeated_characters.items():
    print(f"{char}-{count}")
```

Slip No - 13

A) Python program to input a positive integer with exception handling:

python

```
def input_positive_integer():
    while True:
        try:
            number = int(input("Enter a positive integer: "))
            if number <= 0:
                raise ValueError("The number must be positive.")
            print(f"You entered: {number}")
            break  # Exit the loop if input is valid
        except ValueError as e:
            print(f"Invalid input: {e}. Please try again.")

# Call the function to execute
input_positive_integer()
```

B) Program to implement the concept of queue using a list:

python

```
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item):
        self.queue.append(item)
        print(f"Enqueued: {item}")

    def dequeue(self):
        if not self.is_empty():
            item = self.queue.pop(0)
            print(f"Dequeued: {item}")
            return item
        else:
            print("Queue is empty!")

    def is_empty(self):
        return len(self.queue) == 0

    def size(self):
        return len(self.queue)

    def display(self):
        print("Queue:", self.queue)

# Sample usage of the Queue class
my_queue = Queue()
my_queue.enqueue(1)
my_queue.enqueue(2)
my_queue.enqueue(3)
my_queue.display()

my_queue.dequeue()
my_queue.display()

my_queue.dequeue()
my_queue.dequeue()
my_queue.dequeue()  # Attempt to dequeue from an empty queue
```

Slip No -14

A) Python GUI program to accept dimensions of a cylinder and display the surface area and volume:
python

```
import tkinter as tk
import math

def calculate_cylinder():
    try:
        radius = float(radius_entry.get())
        height = float(height_entry.get())

        # Calculate surface area and volume
        surface_area = 2 * math.pi * radius * (radius + height)
        volume = math.pi * (radius ** 2) * height

        # Display results
        surface_area_label.config(text=f"Surface Area: {surface_area:.2f}")
        volume_label.config(text=f"Volume: {volume:.2f}")
    except ValueError:
        surface_area_label.config(text="Invalid input! Please enter numbers.")
        volume_label.config(text="")

# Create the main window
root = tk.Tk()
root.title("Cylinder Calculator")

# Create labels and entries for radius and height
radius_label = tk.Label(root, text="Enter Radius:")
radius_label.pack(pady=5)
radius_entry = tk.Entry(root)
radius_entry.pack(pady=5)

height_label = tk.Label(root, text="Enter Height:")
height_label.pack(pady=5)
height_entry = tk.Entry(root)
height_entry.pack(pady=5)

# Create a button to calculate
calculate_button = tk.Button(root, text="Calculate", command=calculate_cylinder)
calculate_button.pack(pady=20)

# Create labels to display surface area and volume
surface_area_label = tk.Label(root, text="")
surface_area_label.pack(pady=5)
volume_label = tk.Label(root, text="")
```

```
volume_label.pack(pady=5)

# Run the main loop
root.mainloop()
```

B) Python program to display plain text and cipher text using Caesar encryption:
python

```
def caesar_encrypt(text, shift):
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            shift_base = 65 if char.isupper() else 97
            encrypted_char = chr((ord(char) + shift - shift_base) % 26 + shift_base)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char
    return encrypted_text

# Get user input
plain_text = input("Enter plain text: ")
shift_value = int(input("Enter shift value (1-25): "))

# Encrypt the plain text
cipher_text = caesar_encrypt(plain_text, shift_value)

# Display the results
print(f"Plain Text: {plain_text}")
print(f"Cipher Text: {cipher_text}")
```

Slip No - 15

A) Python class named Student with attributes student_name and marks:
python

```
class Student:
    def __init__(self, student_name, marks):
        self.student_name = student_name
        self.marks = marks

    def display(self):
        print(f"Student Name: {self.student_name}, Marks: {self.marks}")

# Create an instance of the Student class
student = Student("Alice", 85)

# Print original values
print("Original values:")
student.display()

# Modify attribute values
student.student_name = "Bob"
student.marks = 90

# Print modified values
print("Modified values:")
student.display()
```

B) Python program to accept a string and remove characters with odd index values using a user-defined function:
python

```
def remove_odd_index_chars(input_string):
    return ''.join(char for index, char in enumerate(input_string) if index % 2 == 0)

# Accepting input from the user
user_input = input("Enter a string: ")

# Removing characters with odd index values
result_string = remove_odd_index_chars(user_input)

# Display the result
print("String after removing characters at odd index values:", result_string)
```

Slip No - 16

A) Python script to create a class Rectangle with methods to compute area and perimeter:
python

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

# Example usage
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

rectangle = Rectangle(length, width)

print(f"Area of the rectangle: {rectangle.area()}")
print(f"Perimeter of the rectangle: {rectangle.perimeter()}")
```

B) Python GUI program to manage items in a listbox:
python

```
import tkinter as tk
from tkinter import messagebox

def add_item():
    item = entry.get()
    if item:
        listbox.insert(tk.END, item)
        entry.delete(0, tk.END)
    else:
        messagebox.showwarning("Warning", "Please enter an item.")

def print_selected_item():
    try:
        selected_item = listbox.get(listbox.curselection())
        print(f"Selected Item: {selected_item}")
    except tk.TclError:
        messagebox.showwarning("Warning", "Please select an item to print.")
```

```
def delete_selected_item():
    try:
        listbox.delete(listbox.curselection())
    except tk.TclError:
        messagebox.showwarning("Warning", "Please select an item to delete.")

# Create the main window
root = tk.Tk()
root.title("Listbox Manager")

# Create an entry widget
entry = tk.Entry(root)
entry.pack(pady=10)

# Create buttons to add, print, and delete items
add_button = tk.Button(root, text="Add Item", command=add_item)
add_button.pack(pady=5)

print_button = tk.Button(root, text="Print Selected Item", command=print_selected_item)
print_button.pack(pady=5)

delete_button = tk.Button(root, text="Delete Selected Item", command=delete_selected_item)
delete_button.pack(pady=5)

# Create a listbox to display items
listbox = tk.Listbox(root)
listbox.pack(pady=10)

# Run the main loop
root.mainloop()
```