University of Colombo School of Computing
SCS1303 - Introduction to Software Engineering
IDE-Based RAD Activities

Tutorial 03

---

1. Project Setup with Virtual Environments and Module Structure
   - I. Create a new folder named RADProject.
   - II. In the integrated terminal, set up a Python virtual environment:

     python -m venv venv

2. **Rapid Calculator Prototype (Basic Prototyping)**

I. Create a Python script in VS Code that:

Asks the user to input two numbers.
Asks for the operation: add, subtract, multiply, divide.
Displays the result.

II. Requirements:

Use *input()* for user data

Use *if-elif* conditions for operations

Handle invalid inputs

Sample Code:

```
print("Welcome to the Rapid Calculator!")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
operation = input("Choose operation (+, -, *, /): ")

if operation == '+':
    result = num1 + num2
elif operation == '-':
    result = num1 - num2
elif operation == '*':
    result = num1 * num2
elif operation == '/':
    if num2 != 0:
        result = num1 / num2
```

```
        else:
            result = "Cannot divide by zero!"
    else:
        result = "Invalid operation."
    print(f"Result: {result}")
```

3. Temperature Converter Prototype (Unit Conversion Tool)

I. Create a Python script in VS Code that:

- ★ Asks the user to input a temperature value.
- ★ Asks whether they want to convert from **Celsius to Fahrenheit** or **Fahrenheit to Celsius**.
- ★ Displays the converted result.

II. Requirements:

- ★ Use input() for user data
- ★ Use if-elif for conversion logic
- ★ Handle invalid selections or non-numeric inputs

Sample Output:

```
PS T:\python> python temperature_converter.py
Enter temperature value: 100
Convert to (C/F): C
Converted temperature: 37.78°C
```

4. Basic Unit Price Calculator (Practical Business Tool)

I. Create a Python script in VS Code that:

- ★ Asks the user for the **item name**, **total cost**, and **quantity**.
- ★ Calculates and displays the **unit price**

II. Requirements;

- ★ Use input() for data
- ★ Handle zero or negative quantity input
- ★ Display formatted results

Sample Output:

```
PS T:\python> python unit_price_calculator.py
Enter item name: Pencil
Enter total cost: 120
Enter quantity: 10
Unit price for Pencil is: 12.00
```

5. Student Grade Tracker (User-Centric Design & Iteration - Modular Development)

I. Create a script to input and store 3 students' names and their grades. Calculate and display:

★ Average grade
★ Pass/Fail (assume pass ≥ 50)

After the first run, ask users what features they'd like to add (e.g., highest grade, alphabetical order). Then, implement **at least one improvement**.

II. Requirements:

★ Use *lists* or *dictionaries*
★ Prompt user for suggestions
★ Implement one enhancement

Sample Code:

```
students = {}
num_students = 3

for _ in range(num_students):
    name = input("Enter student name: ")
    grade = float(input(f"Enter grade for {name}: "))
    students[name] = grade

# Display average
average = sum(students.values()) / len(students)
print(f"\nAverage Grade: {average:.2f}")

# Display pass/fail
print("\nResults:")
for name, grade in students.items():
    status = "Pass" if grade >= 50 else "Fail"
    print(f"{name}: {grade} - {status}")

# Sample improvement: Show highest scorer
highest = max(students, key=students.get)
print(f"\nHighest Scorer: {highest} with {students[highest]}")
```

6. Modular To-Do List App (Modular Design)

I. Build a command-line To-Do List App with the following modules:

add_task()
view_tasks()
remove_task()
Let users choose options from a menu to perform actions.

II. Requirements:

Use functions for modularity
Use a loop for the menu
Store tasks in a list

Sample Code:

```
tasks = []

def add_task():
    task = input("Enter new task: ")
    tasks.append(task)
    print("Task added!\n")

def view_tasks():
    if tasks:
        print("\nYour Tasks:")
        for i, task in enumerate(tasks, 1):
            print(f"{i}. {task}")
    else:
        print("\nNo tasks found.")
    print()

def remove_task():
    view_tasks()
    try:
        task_num = int(input("Enter task number to remove: "))
        if 1 <= task_num <= len(tasks):
            removed = tasks.pop(task_num - 1)
            print(f"Removed: {removed}\n")
        else:
            print("Invalid task number.\n")
    except ValueError:
        print("Please enter a valid number.\n")
```

```
while True:
    print("1. Add Task\n2. View Tasks\n3. Remove Task\n4. Exit")
    choice = input("Choose an option: ")

    if choice == '1':
        add_task()
    elif choice == '2':
        view_tasks()
    elif choice == '3':
        remove_task()
    elif choice == '4':
        print("Exiting To-Do List App. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.\n")
```

7. RAD Form with Basic Validation (Quick GUI with Tkinter – Prototype GUI development)

I. Create a new file ui.py.

II. Use the tkinter library to build a simple form that takes the user's name, age, and color preference and displays a greeting.

Sample Code:

```
import tkinter as tk

def submit():
    name = entry_name.get()
    age = entry_age.get()
    color = entry_color.get()
    label_result.config(text=f"Hello, {name}. You are {age} and love {color}!")

root = tk.Tk()
root.title("RAD Greeting App")

tk.Label(root, text="Name").pack()
entry_name = tk.Entry(root)
entry_name.pack()
```

```
tk.Label(root, text="Age").pack()
entry_age = tk.Entry(root)
entry_age.pack()

tk.Label(root, text="Favorite Color").pack()
entry_color = tk.Entry(root)
entry_color.pack()

tk.Button(root, text="Submit", command=submit).pack()
label_result = tk.Label(root, text="")
label_result.pack()

root.mainloop()
```

Submission:

- **Time allowed:** 1.00 PM – 3.00 PM.
- **Submission method:** Upload your report to the **VLE (Virtual Learning Environment)** using the provided link.
- **Report**: Include screenshots of your work with the VS Code activities.
- **File format:** PDF
- **File naming format:** **YourName with Initials_IndexNumber_RADActivity.pdf** *(e.g., Weerawardhana APK_122344_RADActivity.pdf)*