

Overview

Evaluating student's knowledge of the DIGI curriculum, the lab assessment acts as a critical point of their learning abilities. Accounting for 50 marks, it holds a substantial 35 percent portion in the Annual Digi Evaluation, emphasizing its crucial significance in the certification of Digi students. This underscores its main function in assessing their depth of understanding and knowledge.

Marking Scheme

Full Marks: **50 Marks**

Pass Marks: **20 Marks**

Annual Evaluation Weightage: **35 %**

Assessment Mechanism

In the computer lab, students will take part in an assessment that involves a diverse range of activities from different chapters. These tasks consist of various aspects of the curriculum, each crafted to integrate essential concepts and principles from different chapters. By engaging in hands-on experimentation and practical applications, students not only deepen their understanding but also sharpen critical thinking and problem-solving skills.

Sorting a list of numbers

Objective:

To help students understand how to take multiple user inputs, store data in a list, and use Python's sorting functions to arrange numbers in ascending and descending order.

Task:

Write a Python program that instructs the user to provide 10 numbers and sorts them in ascending and descending order.

Instructions:

1. Create an empty list named numbers.
2. Use a loop to take 10 inputs from the user.
3. Store each number in the list.
4. Use the sort() and reverse() function to sort the list in ascending and descending order.
5. Print both the sorted lists clearly with labels.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
List Creation	Correctly initializes an empty list and stores all 10 numbers.	List created but contains errors or missing items.	List not created properly or data not stored.
Loop Implementation	Correctly uses a for loop to take 10 inputs.	Loop used but with small logic or range errors.	Loop missing or not functioning correctly.
Sorting Functionality	Accurately sorts numbers in both ascending and descending order.	Only one sorting order correct or minor errors in logic.	Sorting incorrect or missing.
Output Statements	Correctly prints both sorted lists with clear labels.	Prints output but unclear or partially correct.	Output missing, incorrect, or incomplete.
Use of Methods	Uses the sort() and reverse() method correctly.	Uses one of the methods incorrectly.	Doesn't use the methods.

Multiplication Table Using While Loop

Objective:

To help students understand how to use a while loop in Python to perform repetitive tasks and display a multiplication table for a given number.

Task:

Write a Python program that asks the user to enter a number and uses a while loop to display the multiplication table of that number up to 10.

Show the result in a clear format (e.g, $5 \times 1 = 5$).

Instructions:

1. Prompt the user to input a number and store it in a variable.
2. Initialize a counter variable.
3. Use a while loop that runs until i becomes greater than 10.
4. Inside the loop, calculate the product of the number and i .
5. Print the result in the format $\text{number} \times i = \text{product}$.
6. Increment the counter by 1 after each iteration.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Variable Initialization	Correctly stores user input and initializes counter properly.	Variables created but with small mistakes or missing initialization.	Variables missing or incorrectly assigned.
Loop Implementation	Properly uses while loop with correct condition.	Loop works but with small logic or boundary errors.	Loop missing or incorrect, causing infinite or no iteration.
Logic	Correctly calculates product in each iteration using number * i.	Product calculated but with minor mistakes in the formula or variable use.	Incorrect or missing multiplication logic.
Output	Displays output clearly in format.	Output shown but lacks proper formatting or missing few lines.	Output unclear, incorrect, or missing.
Python Constructs	Uses valid Python syntax and constructs (while, print, +=, etc.) correctly without errors.	Minor syntax or construct misuse but program runs.	Major syntax errors or use of wrong constructs (e.g., for loop instead of while).

Eligibility to Vote Using If–Else Statement

Objective:

To help students understand how to use conditional statements (if–else) in Python to make decisions based on user input.

Task:

Write a Python program that asks the user to enter their name and age and uses an if–else statement to check whether the user is eligible to vote and print a suitable message based on the result.

Instructions:

1. Prompt the user to input their name and age and store it in separate variables.
2. Convert the age input to an integer.
3. Use an if–else statement to check
 - If the age is 18 or above, print a message with the user name saying the user is eligible to vote.
 - Otherwise, print a message with the user name saying the user is eligible to vote.
4. Use proper indentation and test your program with different ages.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Input Handling	Correctly takes both name and age inputs and stores them in variables as a string.	Inputs taken but with minor errors (e.g., wrong data type or missing variable).	Input section incomplete or incorrect.
Type Conversion	Properly converts age to an integer before comparison.	Conversion done but uncalled or occasionally causes error.	No conversion or incorrect use leading to error.
Logic	Correctly uses if–else to check age eligibility.	Condition used but contains minor logical mistakes.	Missing or incorrect conditional logic.
Output	Displays correct personalized message including the user's name.	Message printed but lacks name or correct formatting.	Output unclear, incorrect, or missing.
Python Constructs	Uses proper Python syntax (if, else, print, input, etc.) without errors.	Minor syntax or indentation issues but runs correctly.	Major syntax errors or wrong constructs used.

Introduction to Variables, Input and Operators

Objective:

To help students understand the concept of variables, user input, and basic arithmetic operations in Python by calculating and displaying the Body Mass Index (BMI) with a personalized message.

Task:

Write a program that prompts the user to input their name, height(in meters) and weight (in kilograms) and displays BMI including personalized message using the user's name. Calculate the BMI using the formula:

$$BMI = \frac{weight(kg)}{height(m)^2}$$

Instructions:

1. Create three variables: name, height, and weight.
2. Use the `input()` function to take the user's name, height, and weight.
3. Convert height and weight to float before calculation.
4. Calculate the BMI using the formula given above.
5. Print a personalized message with the format: “Hello, your name, your BMI is 23”.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Variable Declaration	Correctly declares and uses variables for name, height, and weight.	Variables declared but with naming or data type errors.	Missing or incorrect variable declaration.
Conversion	Properly converts height/weight to a float.	Converted to a different data type.	Inputs missing or handled incorrectly.
Calculation	Applies the correct BMI formula and computes an accurate result.	The formula is mostly correct, but with a small calculation error.	Incorrect or missing BMI calculation.
Output	Displays BMI with a personalized message using the user name.	Output is correct without the personalized message or user name.	Output missing, unclear, or incorrect.
Use of Python Constructs	Uses valid Python syntax, operators, and expressions without errors.	Minor syntax errors that do not affect overall output.	Major syntax or logical errors that prevent program execution.

List: Accessing and Modifying Data

Objective:

To help students understand how to create, access, modify, and update lists in Python using basic list operations such as adding, removing, and printing elements.

Task:

Write a Python program that creates a list named animals containing five different animal names, prints the third item in the list, adds a new animal, removes an animal, and prints the updated list.

Instructions:

1. Create a list named animals containing five different animal names.
2. Use indexing to print the third item in the list.
3. Use the append() method to add "Rhino" to the end of the list.
4. Use the pop() or remove() method to delete the fourth animal.
5. Print the updated animal list.

Tools Used:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
List Creation	Correctly creates a list named animals with five unique animal names.	List created but with fewer items or minor errors.	List missing or incorrectly defined.
Item Access	Correctly prints the third animal using proper index.	Prints an item but uses wrong index or variable.	Incorrect or missing item access.
Item Insertion	Correctly adds the element at the end of the list.	Adds item but method or position incorrect.	Fails to add item or uses wrong syntax.
Item Removal	Removes the fourth item correctly.	Removes an item but not the correct one or with wrong method.	Fails to remove or uses incorrect syntax.
Output	Prints the updated list correctly showing all changes.	Prints list but missing or unclear changes.	Output missing, incorrect, or incomplete.

Introduction to Functions

Objective:

To help students understand how to create and call functions, use parameters and return values, and display results with formatted output in Python.

Task:

Write a Python program with two functions. The first function should take your name and print a greeting message. The second function should take three numbers, find their average percentage, and return it. Call both functions and print the percentage with a clear label.

Instructions:

1. Define a function named greet() that takes one parameter (name) and prints a message in the format: “Hello name, Welcome”
2. Define another function called calculate_percentage() that takes three parameters sub1, sub2, and sub3.
3. Inside the function, calculate the percentage.
4. Return the percentage value to the main program.
5. Call both functions correctly and print the percentage result clearly with a label.

Tools Required:

- Code editing software like Coding Ground, VS Code or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Function Definition	Both functions are defined correctly with parameters.	Functions are defined, but with the wrong data type in the parameter.	Missing or incorrectly defined functions.
Parameter Handling	Correctly passes and uses all parameters inside the functions.	Parameters used but with small mistakes or mismatched names.	Parameters missing or used incorrectly.
Return Statement & Calculation	Accurately calculates and returns the percentage using the correct formula.	Calculation mostly correct but minor logical or formula error.	Incorrect formula or missing return statement.
Function Calling	Both functions are called correctly and executed without errors.	One function called incorrectly or with missing arguments.	Functions not called or result not displayed.
Output Display	Prints both the greeting message and percentage result clearly and correctly.	Output printed but unclear or partially incorrect.	Output missing, unclear, or incorrect.

Array Fundamentals: Importing and Manipulation

Objective:

To help students understand how to import and use arrays in Python, including how to access, modify, and update elements using array methods.

Task:

Write a Python program that uses the array module to create and manipulate an array of student ages. The program should create an array named `ages` containing the ages of ten students, print the third age in the array, append a new age (13) to the end, remove the second age, and then print the updated array.

Instructions:

1. Import the array module from the array library.
2. Create an array named `ages` containing the ages of ten students.
3. Print the third age in the `ages` array using indexing.
4. Append a new age "13" to the end of the array.
5. Remove the second age from the array.
6. Print the updated `ages` array.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Import Statement	Correctly imports the array module using from array import * or equivalent syntax.	Module imported but syntax partially incorrect.	Import missing or incorrect.
Array Creation	Creates an ages array with exactly ten numerical values.	Array created but with less than ten or incorrect values.	Array not created or created incorrectly with less than five values.
Element Access	Correctly prints the third age using proper indexing.	Prints an element but with wrong index.	Incorrect or missing element access.
Append and Remove Methods	Correctly appends the element and removes the second element using array methods.	Adds or removes an element but with minor syntax issues.	Methods not used or incorrect implementation.
Output Display	Prints the final updated array clearly showing all changes.	Output printed but incomplete, or changes mentioned in instructions not completed.	Output missing, incomplete, or incorrect.

Calculating Subtotal Using Functions in Python

Objective:

To help students understand how to define and call functions that take a list as a parameter, perform calculations on list elements, and return results in a formatted output.

Task:

Write a Python program that defines a function named calculate_subtotal which takes a list of item prices [2.50, 15.00, 4.25] as input and returns the sum of all prices as the subtotal. Then, call the calculate_subtotal function with a list of at least four item prices and print the result, formatted as a currency amount (for example, \$25.75).

Instructions:

1. Define a function named calculate_subtotal(prices) that takes a list of item prices as its parameter.
2. Inside the function, calculate the total using the sum() function and return the result.
3. In the main program, create a list of at least four item prices.
4. Call the calculate_subtotal() function with the list as an argument and store the returned subtotal in a variable.
5. Print the subtotal using currency formatting (e.g., `print(f"Subtotal: ${subtotal:.2f}")`).

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Function Definition	Defines function correctly with one list parameter.	Function defined but with small syntax or parameter issues.	Function missing or incorrectly defined
Calculation Logic	Correctly calculates subtotal using sum() or appropriate loop.	Calculation mostly correct but with minor logic error.	Calculation incorrect or missing.
Return Statement	Properly returns the subtotal value to the main program.	Return used but placed incorrectly or returns wrong variable.	Missing or incorrect return statement.
Function Calling	Calls the function correctly with a list of at least four prices.	Function called but with wrong argument type or fewer items.	Function not called or causes an error.
Output Formatting	Prints subtotal clearly in proper currency format.	Output printed but lacks proper formatting.	Output missing, incorrect, or unclear.

String Manipulation

Objective:

To help students understand how to create and call functions that take a string as input, perform string operations such as case conversion and character counting, and display the results clearly.

Task:

Write a Python program that asks the user to enter a word or sentence, then defines a function called manipulate_string that takes the user input and converts it to uppercase and counts how many times the letter ‘e’ appears, ignoring case. The program should then print the uppercase version of the text and the number of ‘e’s, both with clear labels.

Instructions:

1. Create a Python program that prompts the user to input a string.
2. Define a function called manipulate_string that:
 - Convert the string to all uppercase (e.g., "hello" becomes "HELLO").
 - Count the occurrences of the letter 'e' and 'E' in the original string.
3. Inside the function, print the results of each manipulation (the uppercase string, and the count of 'e's).
4. Call the manipulate_string() function and pass the user input as an argument.

Tools Required:

- Code editing software like Programmiz, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
User Input	Correctly prompts and stores user input as a string.	Input taken but with wrong data type.	Input missing or handled incorrectly.
Function Definition	Defines function correctly with one parameter.	Function defined but with minor syntax or naming issues.	Function missing or incorrectly defined.
String Manipulation	Correctly converts string to uppercase and counts ‘e’s accurately.	Converts string to uppercase but counts either ‘e’ or ‘E’ in the string.	Incorrect or missing string operations.
Function Calling	Calls the function correctly and passes the user input as an argument.	Function called but argument incorrect or incomplete.	Function not called or fails to execute.
Output	Prints both uppercase string and ‘e’ count clearly with labels.	Output displayed but unclear or missing one result.	Output missing, incorrect, or confusing.

Area Calculator

Objective:

To help students learn how to create and call functions that take multiple inputs, perform calculations, and return more than one result.

Task:

Write a Python program that defines two functions: one to take two numeric inputs from the user and another to calculate both the area of a rectangle and the area of a triangle using those values. The program should return both results and print them clearly with labels.

Instructions:

1. Define a function named `get_shape_parameters()` that asks the user to input two numbers and returns them.
2. Define another function named `calculate_areas(param1, param2)` that calculates:
 - Rectangle Area: $\text{param1} \times \text{param2}$
 - Triangle Area: $0.5 \times \text{param1} \times \text{param2}$
3. Return both values as a list.
4. In the main program, call `get_shape_parameters()` to get the inputs and pass them to `calculate_areas()`.
5. Store the returned values in two variables (e.g., `rect_area`, `tri_area`) and print them clearly.
6. Ensure the output displays both area results with clear labels.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Function Definitions	Defines both functions correctly.	Functions defined but with minor syntax or parameter errors.	One or both functions missing or incorrect.
Input Handling	Correctly prompts for and returns two numeric inputs.	Input taken but not returned properly or with type issues.	Input missing or handled incorrectly.
Calculations	Accurately calculates both areas using correct formulas.	Formulas mostly correct but with small logical errors.	Formulas incorrect or missing.
Return Values	Returns both area results correctly	Returns values but with incorrect format or variable handling.	Missing or incorrect return of results.
Output Display	Prints both results clearly with descriptive labels.	Output shown but unclear or missing one label.	Output missing, incorrect, or confusing.

Boolean Logic Evaluator

Objective:

To help students understand how to take Boolean inputs, use logical operators, and perform different Boolean operations in Python.

Task:

Write a Python program that asks the user to enter two Boolean values (True or False) and a logical operator (AND, OR, NOT, NAND, NOR, or XOR). The program should read the inputs, perform the selected Boolean operation, and print the result on the screen.

Instructions:

1. Prompt the user to enter two Boolean values (True or False).
2. Prompt the user to enter a logical operator (AND, OR, NOT, NAND, NOR, XOR).
3. Read the inputs and perform the specified Boolean operation.
4. Print the result of the Boolean operation to the console.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Input Handling	Correctly prompts for two Boolean values and a logical operator.	Prompts present but missing one input or with type errors.	Inputs not handled or stored correctly.
Conditional Logic	Correctly uses if-elif-else to check the operator and perform the right operation.	Logic mostly correct but missing one operator or contains small errors.	Logic incorrect or missing.
Boolean Operations	Accurately performs all operations.	Accurately performs 3 or more operations	Performs less than 3 operations correctly.
Output	Prints the result clearly with a descriptive label.	Output printed but unclear or missing label.	Output missing or incorrect.
Code Accuracy & Syntax	Code runs smoothly without syntax or logical errors.	Minor syntax or logic issues but runs with partial output.	Major errors prevent program from running.

Understanding Tuples

Objective:

To help students understand how to create and use tuples in Python, access elements, use built-in functions, and recognize that tuples are immutable.

Task:

Write a Python program that creates a tuple named `week_days` containing the seven days of the week, starting with “Sunday” and ending with “Saturday.” The program should print the fourth day in the tuple, count how many times “Sunday” appears, and print that result. Then, try to change “Sunday” to “Funday” and observe the error message that appears, showing that tuples cannot be modified. Finally, print the total number of days in the tuple.

Instructions:

1. Create a tuple named `week_days` containing the seven names of the days of the week, starting with "Sunday" and ending with "Saturday."
2. Print the fourth day in the `week_days`.
3. Use a built-in function to count how many times the name "Sunday" appears in the `week_days` tuple. Print the result.
4. Attempt to change the first day ("Sunday") in the tuple to "Funday." Comment and explain the message when you run the code.
5. Print the length (total number of days) of the original `week_days` tuple.

Tools Required:

- Code editing software like Coding Ground, VS Code, or Notepad.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Tuple Creation	Correctly creates a tuple with seven-day names in order.	Tuple created but missing days or minor syntax errors.	Tuple missing or defined incorrectly.
Element Access	Correctly prints the fourth day using the right index.	Prints a day but with incorrect index or minor errors.	Fails to access the fourth day or fails to use the count method correctly.
Built-in Functions	Correctly uses .count() and len() functions with accurate output.	Functions used but with minor syntax or logic errors.	Functions missing or used incorrectly.
Tuple Immutability	Attempts to change tuple value and identifies immutability error clearly.	Attempts change but explanation unclear or incomplete.	Does not attempt change or misinterprets error.
Output Display	Prints all required outputs clearly and with proper labeling.	Output printed but missing labels or unclear formatting.	Output incomplete, incorrect, or missing.

Data Analysis

Objective:

Perform basic data analysis using spreadsheet software and visually present findings through graphical representations.

Task:

Perform simple data analysis on the provided dataset. Carry out the specified calculations and represent the results visually using appropriate charts or graphs.

Instructions:

ID	Age	Distance (km)	Time (mins)	Favorite Shoe Brand	Calories Burned
1	17	5.5	35	Nike	350
2	15	3.2	21	Adidas	200
3	16	3.2	25	Puma	210
4	15	8.1	58	Adidas	600
5	17	5.5	44	Puma	400
6	16	6.2	32	Nike	550
7	15	3.2	31	Skechers	250

8	17	8.1	41	Puma	550
9	16	6.2	40	Nike	430

1. Calculate the statistical measures
 - For the Distance (km), calculate the Mean, Median, and Range.
 - For the Time (minutes), calculate the Mean and the Mode.
 - Identify the maximum and minimum values in the Calories Burned column.
2. Represent the data using appropriate charts or graphs for the Favorite Shoes Brand.

Tools Required:

- Microsoft Excel for figuring out present findings graphically.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Data Entry	Dataset entered accurately and neatly formatted in the spreadsheet.	Minor data entry errors or inconsistent formatting.	Major errors or missing entries.
Statistical Calculations	Correctly calculates all required measures.	Most calculations are correct, with minor formula or rounding errors.	Incorrect or missing calculations.
Chart Creation	Creates a clear, accurate, and well-labeled chart showing shoe brand distribution.	The chart is present but with missing labels or unclear visuals.	Chart missing or incorrect.
Presentation of Results	Results and charts are clearly labeled and easy to interpret.	Results shown, but unclear or missing titles/labels.	Results are disorganized or incomplete.
Neatness & Accuracy	Spreadsheet is well-organized, labeled, and error-free.	Mostly neat but lacks consistent labeling or formatting.	Untidy, incomplete, or error-prone.

Data Validation

Objective:

Design and implement data validation rules in a spreadsheet to ensure the data entered is accurate and adheres to specific constraints.

Task:

Set up data validation rules for a simple enrollment form table to prevent incorrect entries for age, grade, and gender.

Instructions:

Column Header	Data Type	Validation Rule
Student Name	Text	Must be less than 15.
Age	Whole Number	Must be a whole number between 13 and 18, inclusive.
Grade Level	List	Must be selected from a dropdown list containing only: 9, 10, 11, 12.
Gender	Text	Must be selected from a dropdown list containing only: Male, Female, Others
Score	Decimal	Must be a decimal number between 0 and 100, inclusive.

1. Create the table with the five column headers listed above in a new Excel or Google Sheets file.
2. Apply Validation as mentioned above.
3. Test the validation and record the ones that fail.
4. Record any validation failures and note the type of error displayed.

Tools Required:

- Spreadsheet software like Excel or Google Sheets.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Table Creation	Table created accurately with all required headers.	Table created but missing minor details or formatting.	Table incomplete or incorrectly formatted.
Data Validation Setup	Correct validation rules applied to all columns.	Most validation rules applied, minor errors present.	Validation rules missing or incorrectly applied.
Dropdown Lists	Dropdown lists created correctly.	Dropdowns created but with missing or extra options.	Dropdowns missing or not working.
Testing and Error Checking	Validation tested properly with accurate identification of failed entries.	Three or more testing done but incomplete or unclear.	Two or less testing done or no testing or incorrect results recorded.
Presentation & Accuracy	Spreadsheet is neat, labeled, and fully functional.	Spreadsheet mostly neat but with small inconsistencies.	Spreadsheet messy, inaccurate, or incomplete.

Data Validation

Objective:

To help students learn how to represent data visually by creating bar and pie charts using spreadsheet software.

Task:

Use spreadsheet software (Microsoft Excel or Google Sheets) to enter a dataset showing the number of students participating in different school clubs. Then, create a bar chart and a pie chart to display the data visually.

Instructions:

1. Open Microsoft Excel or Google Sheets.
2. Enter the following dataset:

Club Name	Number of Students
Science Club	25
Music Club	18
Art Club	15
Sports Club	30
Debate Club	12

3. Select all the data and create:

- A Bar Chart showing club names on the X-axis and number of students on the Y-axis.
 - A Pie Chart showing the percentage of total students in each club.
4. Add proper titles, axis labels, and data labels to both charts.
 5. Format the charts with colors of your choice.
 6. Save your file as “Club_Charts”.

Tools Required:

- Spreadsheet software like Excel or Google Sheets.

Marking Rubrics

Criteria	Excellent (7-10)	Good (4-7)	Poor (0-4)
Data Entry	Data entered correctly and neatly formatted.	Minor data entry or spacing errors.	Data incomplete or poorly formatted.
Bar Chart	Bar chart created correctly with title and labels.	Chart created but missing labels or title.	Chart missing or incorrect.
Pie Chart	Pie chart created accurately with labels and percentages.	Pie chart present but with unclear formatting.	Pie chart missing or inaccurate.
Chart Formatting	Both charts well-colored, labeled, and easy to read.	Some formatting missing or inconsistent.	Poorly formatted charts.
Presentation & Accuracy	File neat, well-organized, and correctly saved.	Mostly neat but lacks naming or formatting accuracy.	Untidy or incorrectly saved file.