Task 2 : Implement Conditional, Control and Looping Statements

Dt: 13/08/25

2.1 You are developing a simple grade management system for a school. The system needs to determine the grade of a student based on their score in a test. The grading system follows these rules:

If the score is 90 & above, the grade is "A".
If the score is between 80 and 89, the grade is "B".
If the score is between 70 and 79, the grade is "C".
If the score is between 60 and 69, the grade is "D".
If the score is below 60, the grade is "F".

Aim :-

To implement conditional, control and looping statements using Python

Algorithm :-

1. start
2. Get the input mark from the user
3. with the use of an if-elif-else statement do
   - If the marks >=90 print grade "A".
   - If the mark is between 80 and 89 print grade "B".
   - If the mark is between 70 and 79 print grade "C",
   - If the mark is between 60 and 69 print grade "D".
   - If the mark is below 60, Print grade "F".

Program:-

Input:-

```
Score = int (input ("Enter the score:"))
if score >=90;
  Print ("The Grade is A")
elif (score <=89 and score >=80):
  Print ("The Grade is B")
elif (score <=79 and score >=70):
  Print ("the Grade is c")
elif (score <=69 and Score >=60):
  Print ("The Grade is D")
else:
  Print ("The Grade is F")
```

Output:

```
Enter the score :60
The Grade is D
```

	EX NO.
	PERFORMANCE (5)
	RESULT AND ANALYSIS (3)
	VIVA VOCE (2)

2.2 The electronics maintenance team at a data center needs a tool to access the health status of UPS back up batteries based on their current charge percentage. you are asked to develop a python program that accepts the battery charge percentage as input and categorizes the battery health using the following conditions:

- If the percentage is greater than or equal to 90, display:
  - "Excellent Battery Health"
- If the percentage is between 70 and 89, display:
  - "Good Battery Health"
- If the percentage is between 40 and 69, display:
  - "Average Battery Health"
- If the percentage is below 40, display:
  - "Poor Battery Health"

Task:

write a python program that: Uses laddered if-elif-else statements

Algorithm:

1. Accept battery percentage from the user
2. Use laddered if-elif-else to determine the health category:
   - If percentage ≥ 90 → "Excellent Battery Health"
   - If $70 \leq$ percentage < 90 → "Good Battery Health"
   - If $40 \leq$ Percentage < 70 → "Average Battery Health"
   - If Percentage < 40 → "Poor Battery Health"

Python Program :-

```
#Battery Health checker
Percentage = int(input("Enter battery percentage: "))
if percentage >=90:
Print ("Excellent Battery Health")
elif percentage >=70:
    Print (" Good Battery Health")
elif percentage >=40:
    Print ("Average Battery Health")
else:
Print("Poor Battery Health")
```

Input:-

Battery Charge percentage (integer)

Output:

Enter Battery percentage :85

Good Battery Health

2.3 You're coding a system at an amusement park that checks the height of each visitor

- If the height is 120 cm or more, print "Allowed".
- otherwise, print "NOT allowed".

Repeat this for 5 visitors

Algorithm :

1. Start the program
2. Set the total number of visitors to 5
3. Loop from visitor 1 to visitor 5
   - Accept the height of the visitor as input (in cm)
   - If height is greater than or equal to 120, print "Allowed".
   - Else, print "Not allowed".
4. End the loop after 5 visitors have been checked
5. Stop the program.

VEL TECH	
EX.NO.2	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

Result :

Thus, The Python Program was successfully implemented using Conditional Statement (if-else), control flow and looping ~~statement~~

Program :-

```
for i in range(1,6):
    height = int(input("Enter height of visitor "+str(i)+" in cm: "))
    if height >= 120:
        print("Allowed to ride.")
    else:
        print("Not allowed to ride.")
```

Sample Inputs :-

Enter height of visitor 1 in cm : 130
Enter height of visitor 2 in cm : 110
Enter height of visitor 3 in cm : 150
Enter height of visitor 4 in cm : 90
Enter height of visitor 5 in cm : 125

Sample Output :-

Allowed
Not allowed
Allowed
Not allowed
All allowed

Task 3 Importing and Creating Python modules and Packages in Python Program

Dt: 20/08/25

Aim: To implement and demonstrate the process of importing built-in modules, creating user defined modules, and organizing code into Packages in python, there by promoting code reusability, modularity and maintainability.

3.1

1. Perform Common math and random opelations
2. Work with the opelating System (create/change directories, list contents) and read the Python version.
3. Compute basic statistics (mean, media, mode, standard deviation).

Algorithm:-

1. Import required modules: math, random, os, sys, statistics, pathlib.
2. Math & random:
   - Compute sqrt(5), radians(30), a random float in [0.0, 10.0), a random integer in [2, 6] (inclusive), π, ceil(2.3), floor(2.3), factorial(5), gcd(5, 15), abs(-10), pow(3, 5), log base 3 of 24, log 10(a) for a=100, and check NaN/Infinity.
3. OS & Sys:
   - create C:\pythonlab if not present and print the current working directory
   - create C:\pythonlab\214 if not present and change the current working directory to it.
   - List all files/directories in the new current directory
   - Print python interpreter version
4. Statistics:
   - on lists: [5, 6, 8, 10] and [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6] Compute mean, median, mode, stdev.
5. Print neatly formatted results.

```
import math
import random
import os
import sys
import statistics as stats
from pathlib import Path

Print("\n--- MATH & RANDOM ---")
Print("sqrt(5) =", math.sqrt(5))
Print("radians(30) =", math.radians(30))
Print("random() in [0,1) =", random.random())
Print("randint(2,6) =", random.randint(2,6)) # inclusive
Print("pi =", math.pi)
Print("ceil(2.3) =", math.ceil(2.3))
Print("floor(2.3) =", math.floor(2.3))
Print("factorial(5) =", math.factorial(5))
Print("gcd(5, 15) =", math.gcd(5, 15))
Print("abs(-10) =", abs(-10))

Print("pow(3,5) =", pow(3,5))
Print("log base 3 to 2 =", math.log(2,3))
~~Print f~~"
a_val = 100
Print(f'log10({a_val}) =", math.log10(a_val))
inf_val = float('inf')
nan_val = float('nan')
Print(f"isinf(oo) = {math.isinf(inf_val)}, isnan(NaN) = {math.isnan(nan

Print("\n--- os & sys ---")
Path_pythonlab = Path(r"c:\pythonlab")
Path_pythonlab.mkdir(parents=True, exist_ok=True)
Print(f"created/ensured: {Path_pythonlab}")
Print("current working directory:", os.getcwd())
target_dir = Path(r"c:\pythonslot S2L4")
target_dir.mkdir(parents=True, exist_ok=True)
os.chdir(target_dir)
Print(f"changed into: {target_dir}")
Print("Directory contents:", os.listdir())
Print("python version:", sys.version)
```

```
Print C"\n--- STATISTICS ---")
data 1 = [5,6,8,10]
data 2 = [2,5,3,2,8,3,9,4,2,5,6]
Print (f"mean (§data1§) =", stats. mean (data1))
Print (f"median (§data1§) =", stats. median (data1))
Print (f"mode (§data 2§) =", stats. mode (data2))
Print (f"stdev (§data2§) = ", stats, stdev (data2))
```

--MATH & RANDOM---

```
Sqrt (5) = 2.23606797749979
radians (30) = 0.5235987755982988 <- will vary
randint (2, 6) = 6
Pi = 3.141592653589793
ceil (2.3) = 3
floor (2.3) = 2
factorial (5) = 120
gcd (6, 15) = 5
ads (-10) = 10
Pow (3,5) = 243
bg base 3 of 2 = 0.6309297535714574
log 10 (100) = 2.0
isinf (∞) = True, isnan (NAN) = True
```

---OS & SYS---

```
Created/ensured : c:\python lab
Current working directory : C:\... (your current path)
created / ensured & changed into : C:\Python slot S2L4
Directory contents of C:\python slot S2L4 : []
Python version : 3.x.x C... details...)
```

--- STATISTICS ---

```
mean ([5,6,8,10]) = 7.25
median ([5,6,8,10]) = 7.0
mode ([2,5,3,2,8,3,9,4,2,5,6]) = 2
stdev ([2,5,3,2,8,3,9,4,2,5,6]) = 2.2715633383201093
```

3.2

Create a python package named cardpack containing a module card fun that imports the random module. Assign a range of cards, call a function from the module, and display a random sample of cards.

Algorithm :-

Step 1 : Start

Step 2 : To create a package cardpack

Step 3 : To create a module cardfun and import random functio

Step 4 : Assign a cards range

Step 5 : Call a module function.

Step 6 : Display the random sample cards Step 7 : Stop

Program :

Cardfun

```
import random
def func():
 cards = []
 for i in range (1, 53):
 cards. append (i)
 Shuffled_cards= random. sample (cards, k =52)
 Print ("\n\n", shuffled_cards, "\n\n")
```

Mymod. py

```
import Cardfun
CardFun . func()
```

Output :

RESTART :

C: \users \student. MAT2VC6833 \AppData \Local \Programs \Python \Py
\Lib \site packages \CardPack \My Mod. py

[5, 24, 13, 22, 20, 41, 38, 51, 4, 7, 34, 49, 14, 50, 37, 40, 15, 35, 17, 18, 33, 39, 3
12, 16, 16, 19, 48, 29, 2, 27, 11, 31, 46, 28, 21, 32, 8, 25, 30, 23, 26, 10, 43, 6
44, 52, 1, 45, 9]