Tasks – Intermediate Conditions (Input + Variables)

Use `input()`, variables, and `if` / `elif` / `else` together. Tasks are harder than the basic conditions lab: multiple inputs, ranges, compound conditions (`and` / `or`), and nested logic. Create each file, run it, and test with different values.

Run scripts with: `python3 script_name.py`

Part 1 – One input, variables, and conditions

Task 1.1 – Number range with variable (`cond_age_group.py`)

- Create `cond_age_group.py`.
- Ask for age (one `input()`, convert to `int`), store in a variable. If age < 13 print `"Child"`, elif age < 20 print `"Teen"`, elif age < 65 print `"Adult"`, else print `"Senior"`. Run and test with 10, 15, 30, 70.

Expected output (example – input 15):

```
Enter your age: 15
Teen
```

Task 1.2 – Positive / zero / negative (`cond_sign.py`)

- Create `cond_sign.py`.
- Read one number (int or float) into a variable. If it is greater than 0 print `"Positive"`, elif it is less than 0 print `"Negative"`, else print `"Zero"`. Test with 5, -3, 0.

Expected output (example – input -3):

```
Enter a number: -3
Negative
```

Task 1.3 – Even or odd with variable (`cond_even_odd.py`)

- Create `cond_even_odd.py`.
- Read one integer into a variable. Use `%` to decide: if remainder when divided by 2 is 0 print `"Even"`, else print `"Odd"`. Print the number and the result (e.g. "7 is Odd").

Expected output (example – input 7):

```
Enter a number: 7
7 is Odd
```

Task 1.4 – Score to letter grade (`cond_grade.py`)

- Create `cond_grade.py`.
- Read a score (0–100) into a variable. If score >= 90 print `"A"`, elif >= 80 `"B"`, elif >= 70 `"C"`, elif >= 60 `"D"`, else `"F"`. Also print the score in the message (e.g. "Score 85: B").

Expected output (example – input 85):

```
Enter score (0-100): 85
Score 85: B
```

Part 2 – Two inputs and conditions

Task 2.1 – Larger of two numbers (`cond_max_two.py`)

- Create `cond_max_two.py`.
- Read two numbers (int or float) into two variables. Using only `if` / `elif` / `else` (no `max()`), print which one is larger, or `"Equal"` if they are equal. Use variables in the

conditions.

Expected output (example – 10 and 7):

```
First number: 10
Second number: 7
Larger: 10
```

Task 2.2 – Both positive? (`cond_both_positive.py`)

- Create `cond_both_positive.py`.
- Read two integers. If both are positive (each > 0) print `"Both positive"`. Elif both are negative print `"Both negative"`. Else print `"Mixed signs"`. Use variables and `and` in conditions.

Expected output (example – 3 and -2):

```
First: 3
Second: -2
Mixed signs
```

Task 2.3 – In range together (`cond_in_range.py`)

- Create `cond_in_range.py`.
- Read two numbers. If both are between 1 and 10 (inclusive), print `"Both in 1-10"`. Else print `"Not both in range"`. Use one variable for each input and conditions like `a >= 1 and a <= 10`.

Expected output (example – 5 and 8):

```
First (1-10): 5
Second (1-10): 8
Both in 1-10
```

Task 2.4 – Simple “login” (username and password) (`cond_login.py`)

- Create `cond_login.py`.
- Ask for username and password (two inputs, store in variables). Choose a fixed username and password (e.g. `admin` / `secret`). If both match, print `"Welcome"`. Else print `"Invalid"`. Use string comparison (e.g. `user == "admin" and pwd == "secret"`).

Expected output (example – wrong password):

```
Username: admin
Password: wrong
Invalid
```

Part 3 – Compound conditions (and / or)

Task 3.1 – Age and ticket (`cond_age_ticket.py`)

- Create `cond_age_ticket.py`.
- Read age (int) and then a “has ticket” answer (e.g. ask "Do you have a ticket? (yes/no)" and read string). If age >= 18 and answer is "yes" (compare lowercase: `answer.lower() == "yes"`), print `"Entry allowed"`. Else print `"Entry denied"`.

Expected output (example – 20 and yes):

```
Age: 20
Do you have a ticket? (yes/no): yes
Entry allowed
```

Task 3.2 – Discount by age and amount (`cond_discount.py`)

- Create `cond_discount.py`.
- Read age and purchase amount (float). If age >= 65 OR amount >= 100, print `"Discount applies"`. Else print `"No discount"`. Use variables and `or`.

Expected output (example – 30 and 150):

```
Age: 30
Purchase amount: 150
```

```
Discount applies
```

Task 3.3 – Valid triangle sides (`cond_triangle.py`)

- Create `cond_triangle.py`.
- Read three numbers (sides a, b, c). A triangle is valid if each side is positive and the sum of any two sides is greater than the third. If valid print `"Valid triangle"`, else print `"Invalid"`. Use variables and `and` (e.g. `a > 0 and b > 0 and c > 0 and a+b>c and b+c>a and a+c>b`).

Expected output (example – 3, 4, 5):

```
Side a: 3
Side b: 4
Side c: 5
Valid triangle
```

Task 3.4 – Menu choice (1–4) (`cond_menu.py`)

- Create `cond_menu.py`.
- Ask for a number 1–4. Store in a variable. If 1 print `"Option A"`, elif 2 print `"Option B"`, elif 3 print `"Option C"`, elif 4 print `"Option D"`, else print `"Invalid option"`. Run and test with 3 and with 9.

Expected output (example – input 3):

```
Choose 1-4: 3
Option C
```

Part 4 – Nested conditions and more logic

Task 4.1 – Temperature and unit (`cond_temp_unit.py`)

- Create `cond_temp_unit.py`.

- Ask for temperature (number) and unit (string: "C" or "F"). Store both in variables. If unit is "C": if temp < 0 print "Freezing", elif temp < 20 print "Cold", else print "Warm/Hot". If unit is "F": use similar ranges (e.g. 32 for freezing, 68 for cold). Else print "Unknown unit". Use nested if/elif.

Expected output (example – 15 and C):

```
Temperature: 15
Unit (C/F): C
Cold
```

Task 4.2 – Leap year (simplified) (`cond_leap.py`)

- Create `cond_leap.py`.
- Read a year (int). A year is a leap year if divisible by 4, but if divisible by 100 it must also be divisible by 400. Use variables and nested conditions (or `and` / `or`): if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0) then "Leap year", else "Not leap year". Print the result.

Expected output (example – 2024):

```
Year: 2024
Leap year
```

Task 4.3 – Three numbers: smallest (`cond_min_three.py`)

- Create `cond_min_three.py`.
- Read three numbers into variables. Using only if/elif/else (no min()), determine the smallest and print it (e.g. "Smallest: 2"). Compare variables in conditions.

Expected output (example – 7, 2, 9):

```
First: 7
Second: 2
Third: 9
Smallest: 2
```

Task 4.4 – Password length and content (`cond_password.py`)

- Create `cond_password.py`.
- Read a password (string). Store its length in a variable (e.g. `length = len(password)`). If length < 6 print "Too short". Elif length > 12 print "Too long". Elif length >= 6 and length <= 12 print "Length OK". Optionally: if length is OK and password equals a secret word, print "Correct"; else "Wrong". Keep it to length checks if you prefer.

Expected output (example – 4 chars):

```
Password: abcd
Too short
```

Part 5 – Multi-step with variables

Task 5.1 – BMI category (simplified) (`cond_bmi.py`)

- Create `cond_bmi.py`.
- Read weight (kg) and height (m) as floats. Compute `bmi = weight / (height ** 2)` and store in a variable. If bmi < 18.5 print "Underweight", elif bmi < 25 print "Normal", elif bmi < 30 print "Overweight", else print "Obese". Print the BMI value too (e.g. "BMI 22.5: Normal").

Expected output (example – 70 kg, 1.75 m):

```
Weight (kg): 70
Height (m): 1.75
BMI 22.9: Normal
```

Task 5.2 – Day name from number (`cond_day.py`)

- Create `cond_day.py`.
- Read a number 1–7 (1=Monday, 7=Sunday or 1=Sunday, 7=Saturday – choose one and state in prompt). Store in a variable. Use if/elif/else to print the day name. If not 1–7

print "Invalid".

Expected output (example – 3):

```
Day (1-7): 3
Wednesday
```

Task 5.3 – Two numbers: order and sum (`cond_order_sum.py`)

- Create `cond_order_sum.py`.
- Read two integers. If the first is less than the second, print "Ascending" and then print their sum. If the first is greater than the second, print "Descending" and print their difference (first - second). If equal, print "Equal" and print the product. Use variables for both numbers and for the computed result.

Expected output (example – 5 and 10):

```
First: 5
Second: 10
Ascending
15
```

Done

You’ve combined input(), variables, if/elif/else, and/or, nested conditions, and multi-step logic. Use these patterns in bigger programs later.