Tasks – Logical Operators (and, or, not)

Practice `and`, `or`, and `not` with conditions and variables. Create each file, run it, and check the output.

Run scripts with: `python3 script_name.py`

Part 1 – and

Task 1.1 – Both conditions true (`logic_and_basic.py`)

- Create `logic_and_basic.py`.
- Print the result of `True and True`. Then print `True and False`. Then print `False and False`.

Expected output:

```
True
False
False
```

Task 1.2 – and with comparisons (`logic_and_compare.py`)

- Create `logic_and_compare.py`.
- Print the result of `5 > 3 and 10 > 7`. Then print `5 > 3 and 2 > 10`. Use the same style: two comparisons joined by `and`.

Expected output:

```
True
False
```

Task 1.3 – and with variables (`logic_and_var.py`)

- Create `logic_and_var.py` .
- Assign `age = 25` and `has_ticket = True` . Print the result of `age >= 18 and has_ticket` . Then set `has_ticket = False` and print the same expression again (without redefining age).

Expected output:

```
True
False
```

Task 1.4 – Range check with and (`logic_and_range.py`)

- Create `logic_and_range.py` .
- Read an integer `n` . Print `True` if `n` is between 1 and 10 inclusive (i.e. `n >= 1 and n <= 10`), otherwise print `False` . Use one `print()` with that condition.

Expected output (example – input 5, then 15):

```
Enter a number: 5
True
```

(Run again with 15 to see False.)

Part 2 – or

Task 2.1 – or basic (`logic_or_basic.py`)

- Create `logic_or_basic.py` .
- Print `True or False` . Then print `False or False` . Then print `True or True` .

Expected output:

```
True
False
```

```
True
```

Task 2.2 – or with comparisons (`logic_or_compare.py`)

- Create `logic_or_compare.py` .
- Print the result of `3 > 5 or 10 > 7` . Then print `3 > 5 or 2 > 10` . At least one side must be true for the whole expression to be true.

Expected output:

```
True
False
```

Task 2.3 – or with variables (`logic_or_var.py`)

- Create `logic_or_var.py` .
- Assign `is_weekend = False` and `is_holiday = True` . Print `is_weekend or is_holiday` . Then set both to `False` and print the same expression again.

Expected output:

```
True
False
```

Task 2.4 – Discount: age or amount (`logic_or_discount.py`)

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

Expected output (example – age 30, amount 150):

```
Age: 30
Amount: 150
```

```
Discount applies
```

Part 3 – not

Task 3.1 – not basic (`logic_not_basic.py`)

- Create `logic_not_basic.py` .
- Print `not True` . Then print `not False` .

Expected output:

```
False
True
```

Task 3.2 – not with comparison (`logic_not_compare.py`)

- Create `logic_not_compare.py` .
- Print the result of `not (5 > 10)` . Then print `not (3 < 5)` . The parentheses make it clear: first evaluate the comparison, then apply `not` .

Expected output:

```
True
False
```

Task 3.3 – not with variable (`logic_not_var.py`)

- Create `logic_not_var.py` .
- Assign `empty = True` (meaning "the box is empty"). Print `not empty` . Then assign `empty = False` and print `not empty` again.

Expected output:

```
False
True
```

Part 4 – Combining and, or, not

Task 4.1 – and and or (`logic_and_or.py`)

- Create `logic_and_or.py` .
- Print the result of `(True and False) or True` . Then print `False and (True or False)` . Use parentheses to control order: `and` is evaluated before `or` if you don't use parens, but parentheses make it clear.

Expected output:

```
True
False
```

Task 4.2 – not and and (`logic_not_and.py`)

- Create `logic_not_and.py` .
- Assign `a = True` and `b = False` . Print `not (a and b)` . Then print `(not a) and b` . See how parentheses change the result.

Expected output:

```
True
False
```

Task 4.3 – Real check: valid age and password (`logic_combined.py`)

- Create `logic_combined.py` .
- Read age (int) and a password string (e.g. "secret"). If `age >= 18 and password == "secret"` , print `"Access granted"` . Else if `age < 18 and password == "secret"` , print

`"Too young"`. Else print `"Access denied"`. Use variables and one or more if/elif/else.

Expected output (example – 20, secret):

```
Age: 20
Password: secret
Access granted
```

Task 4.4 – not in condition (`logic_not_in_condition.py`)

- Create `logic_not_in_condition.py`.
- Read an integer `n`. If `not (n >= 1 and n <= 10)` (i.e. n is outside 1–10), print `"Out of range"`. Else print `"In range"`. You can also write it as `n < 1 or n > 10` (De Morgan: not (A and B) is (not A) or (not B)).

Expected output (example – 0, then 5):

```
Enter n: 0
Out of range
```

(Run with 5 to see "In range".)

Done

You've used: **and** (both true), **or** (at least one true), **not** (negate), and combined them with variables, comparisons, and if/elif/else.