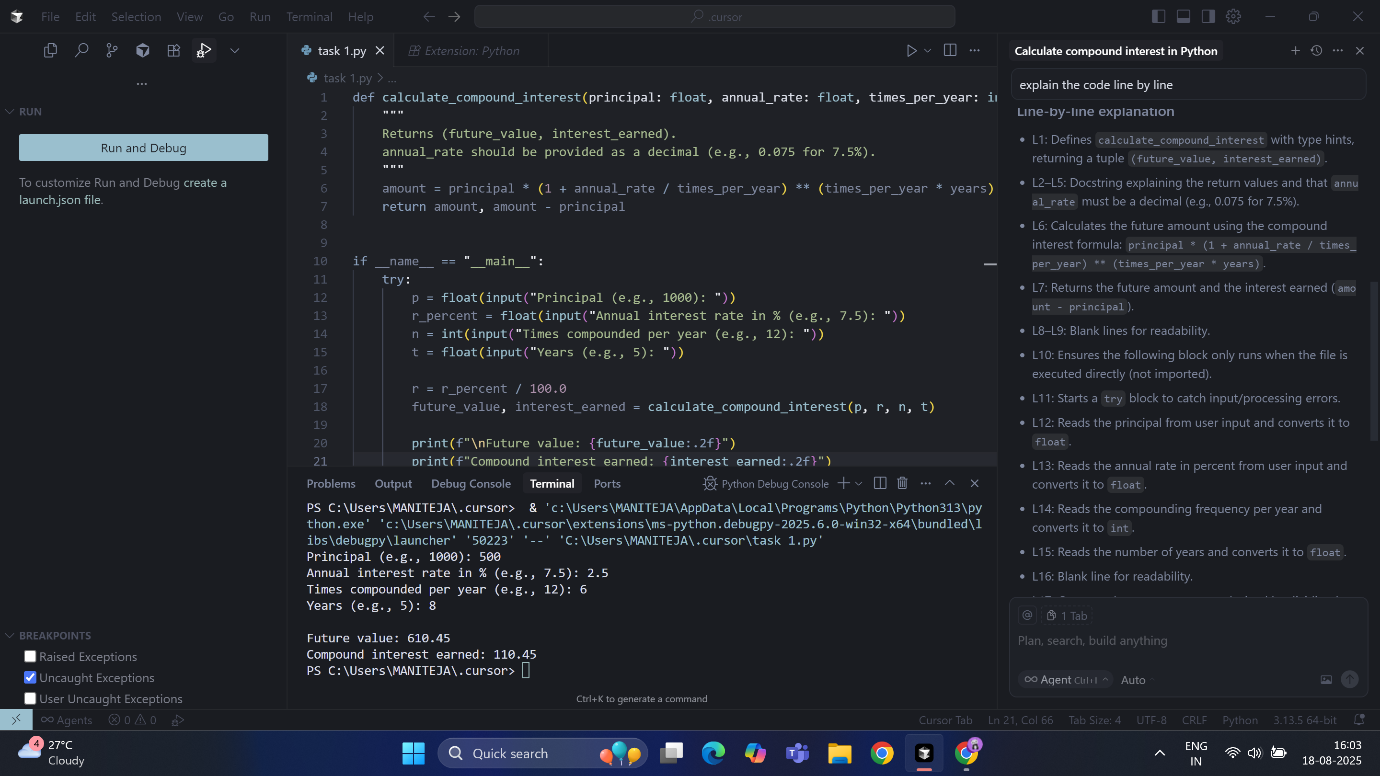
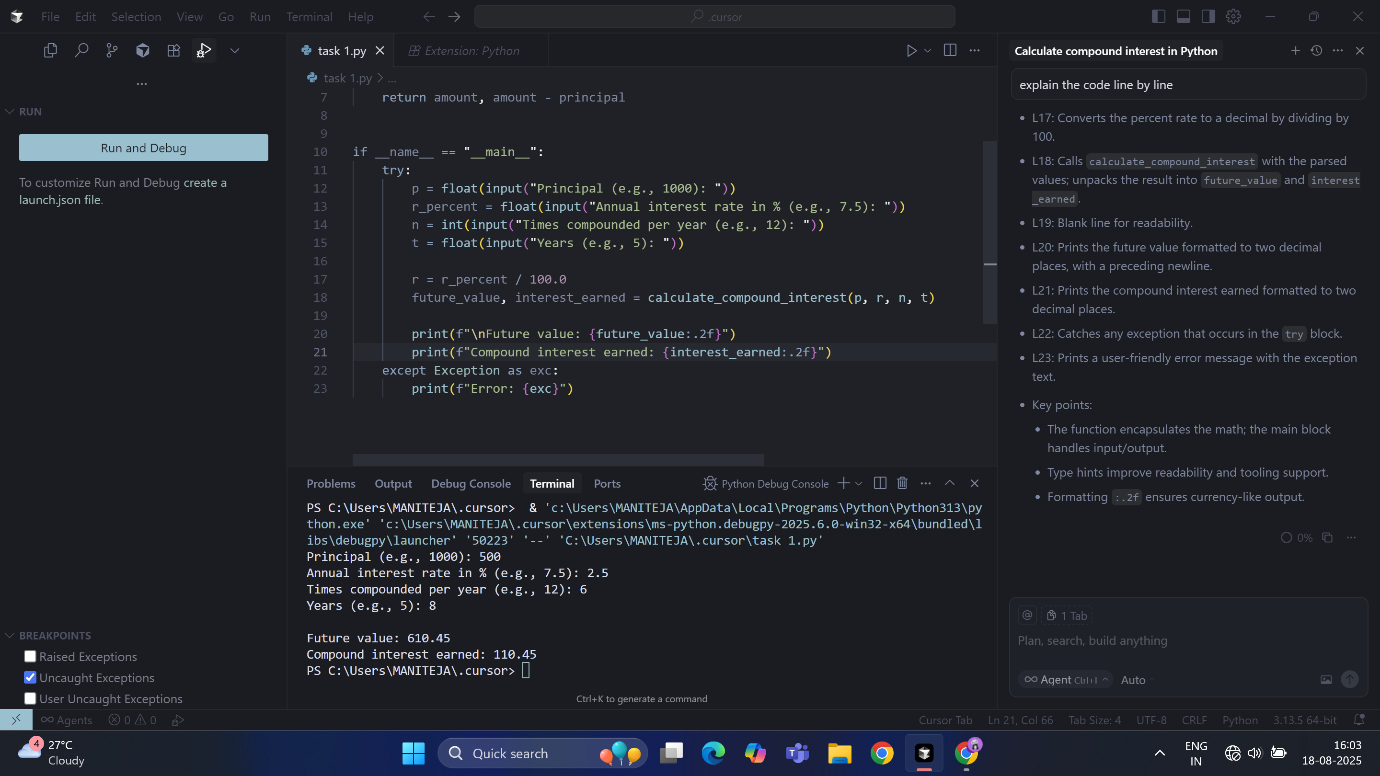
HTNO-2403A52134

Assignment-3.2

Task-1:

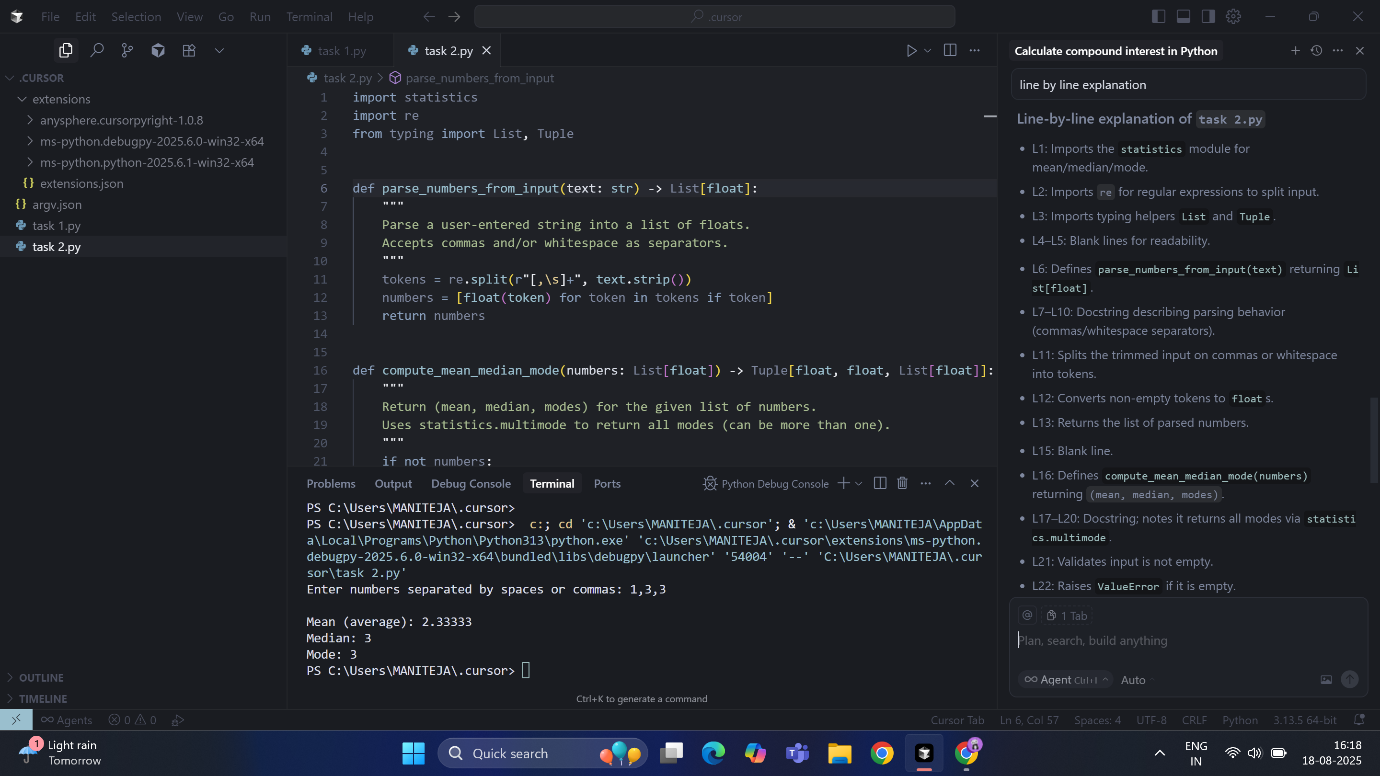
To find the compound interest

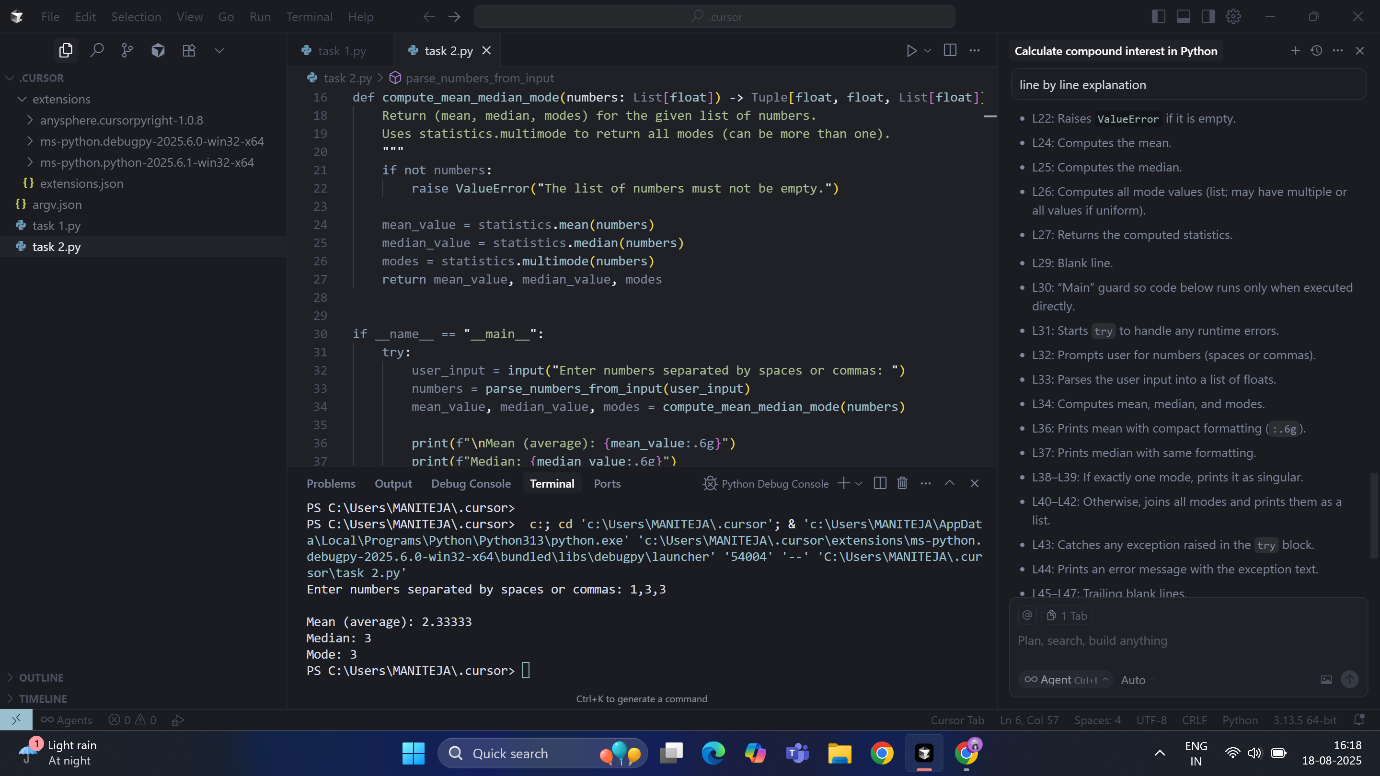


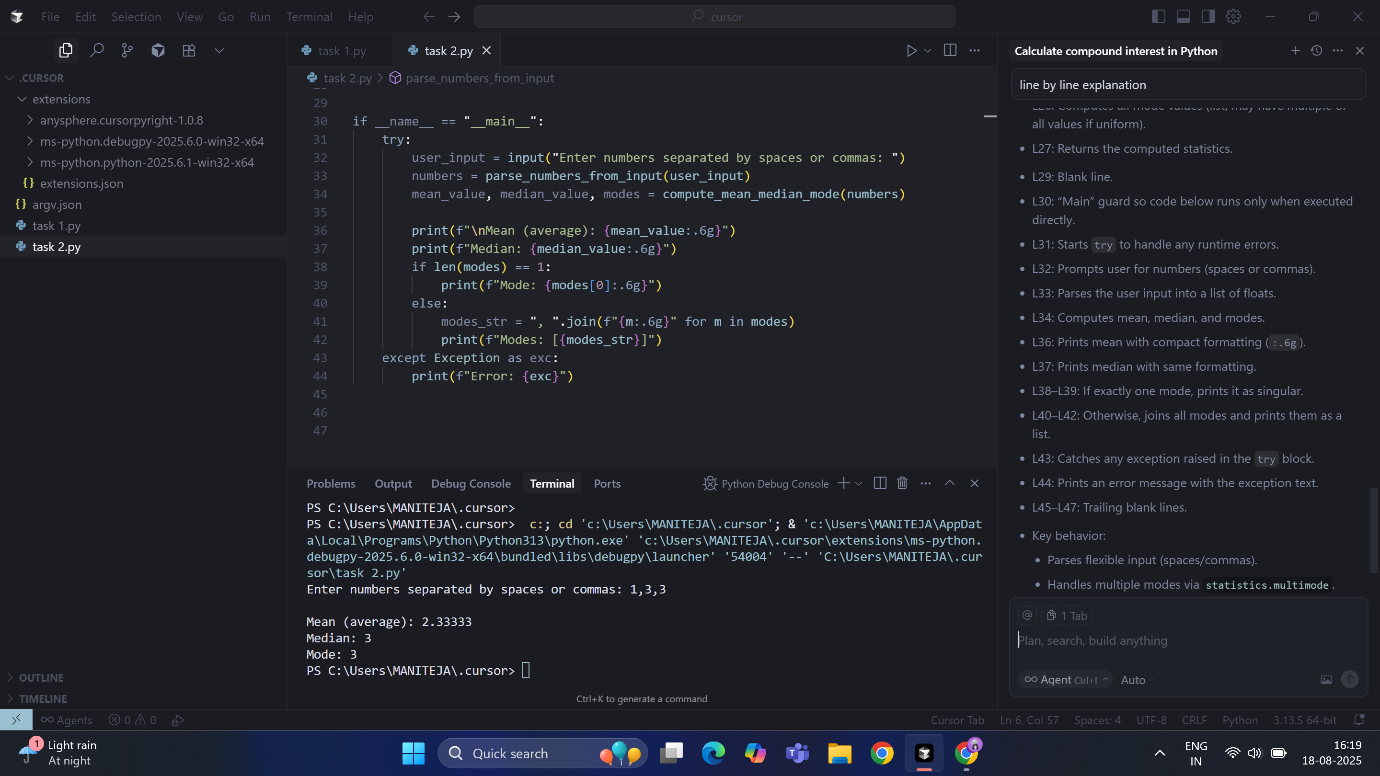


Task-2:

To calculate average, median, and mode of a list of numbers.

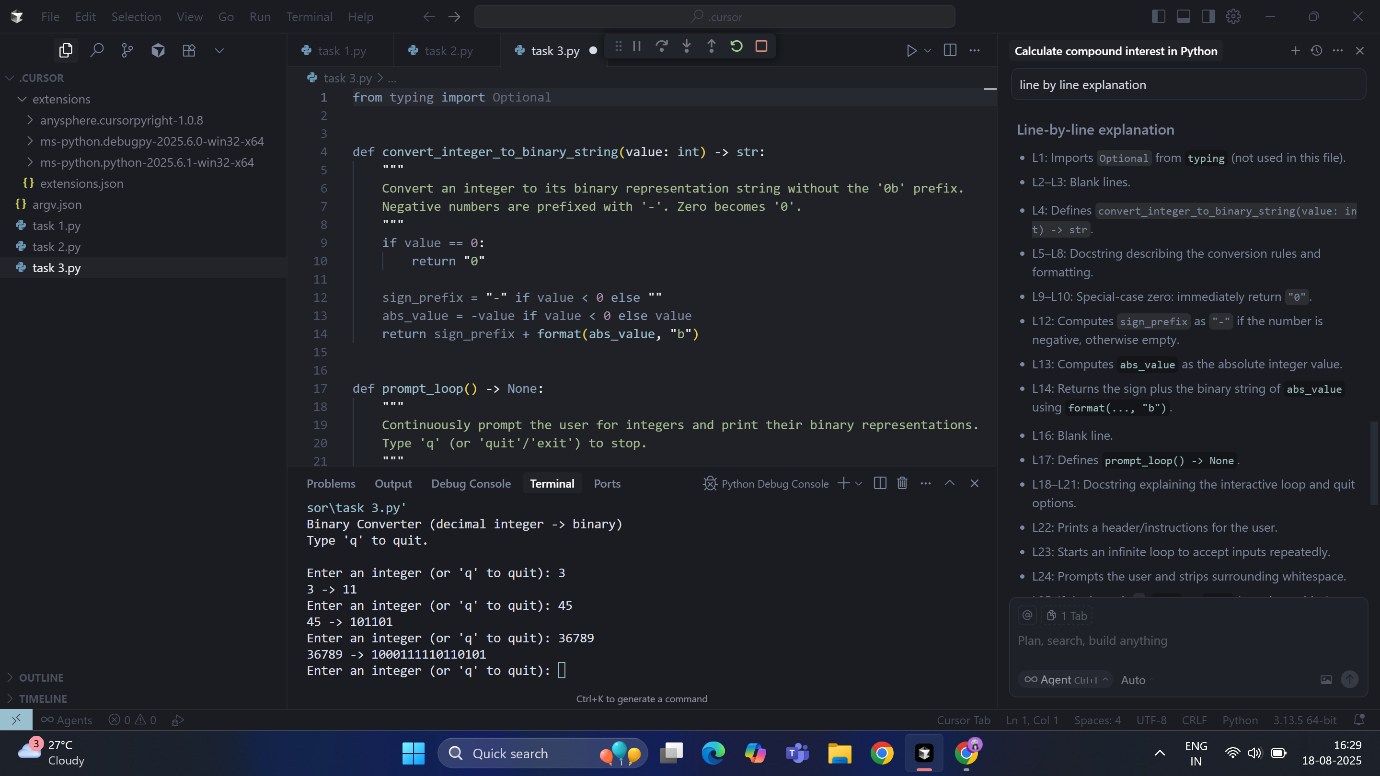


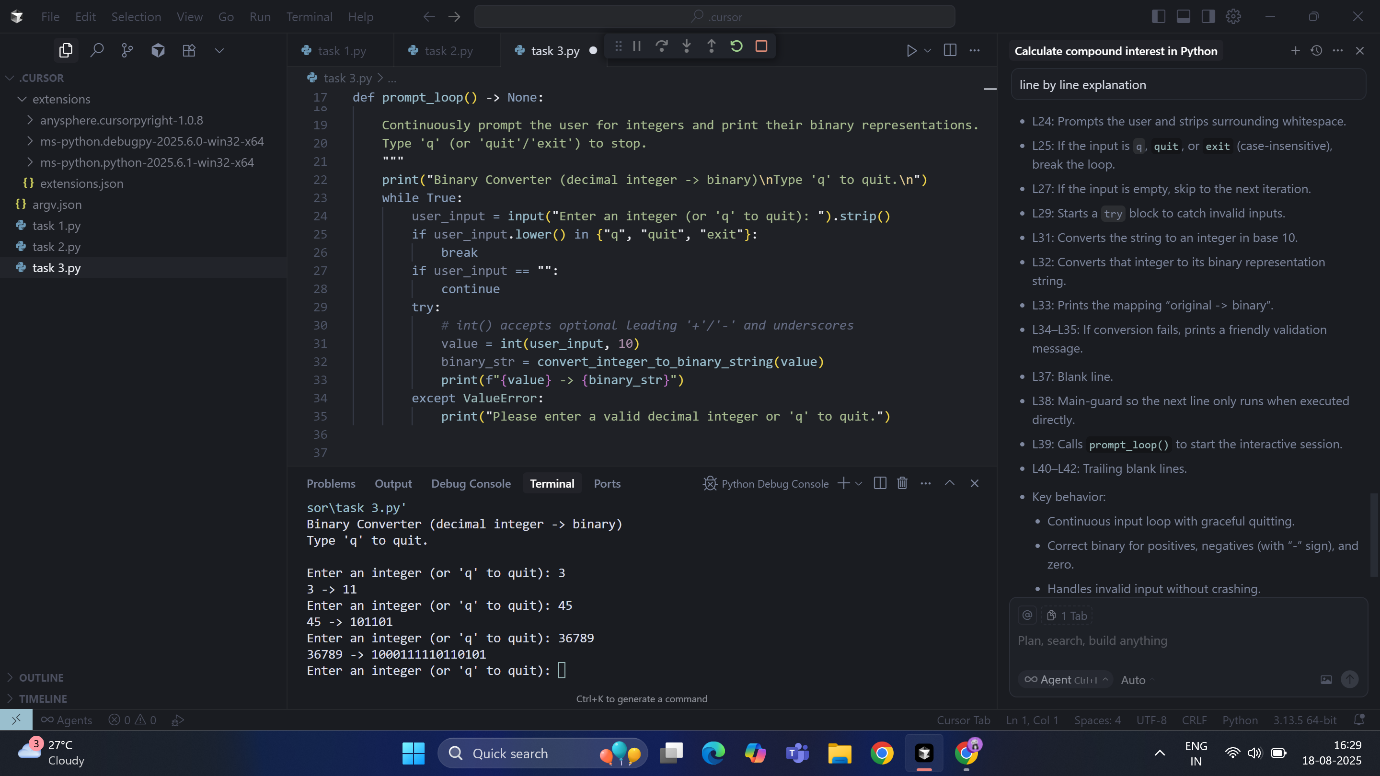


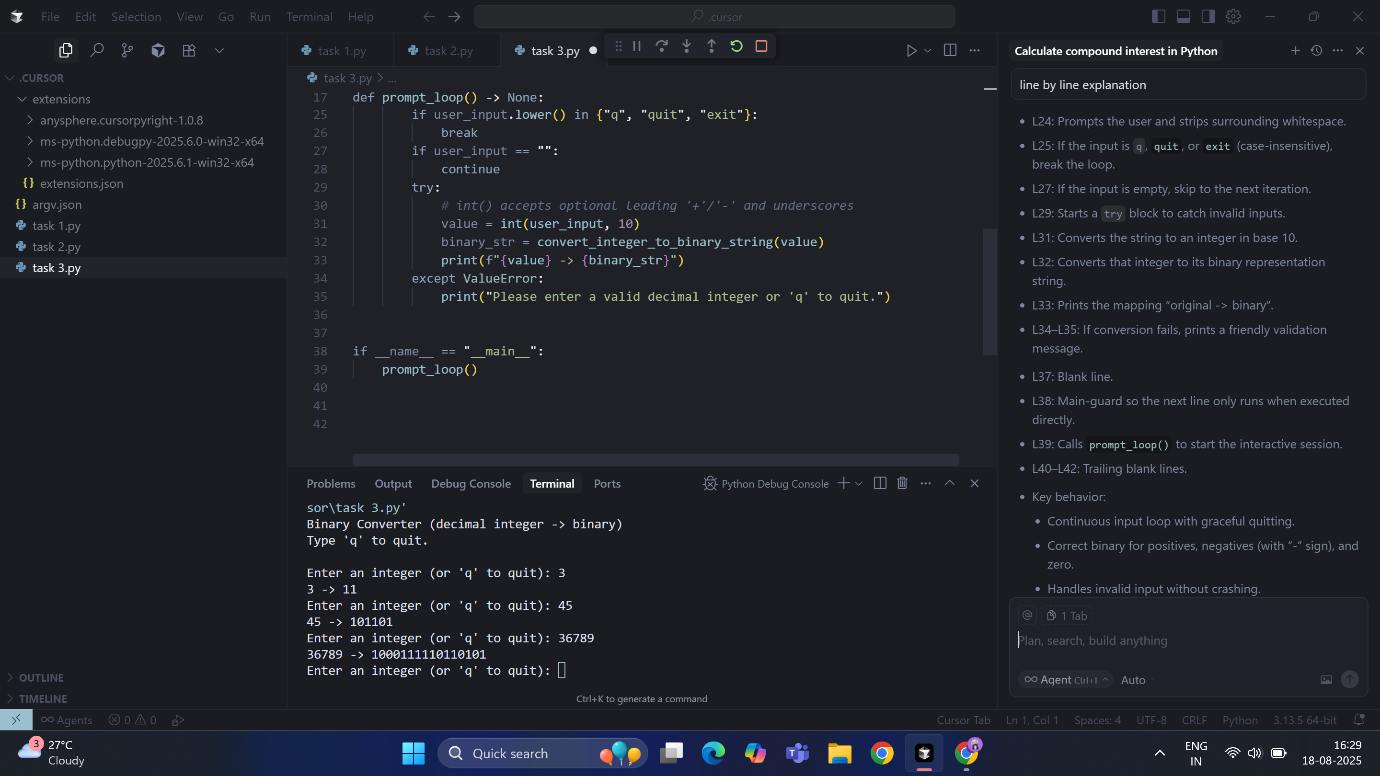


Task-3:

To convert number into binary format with input-output



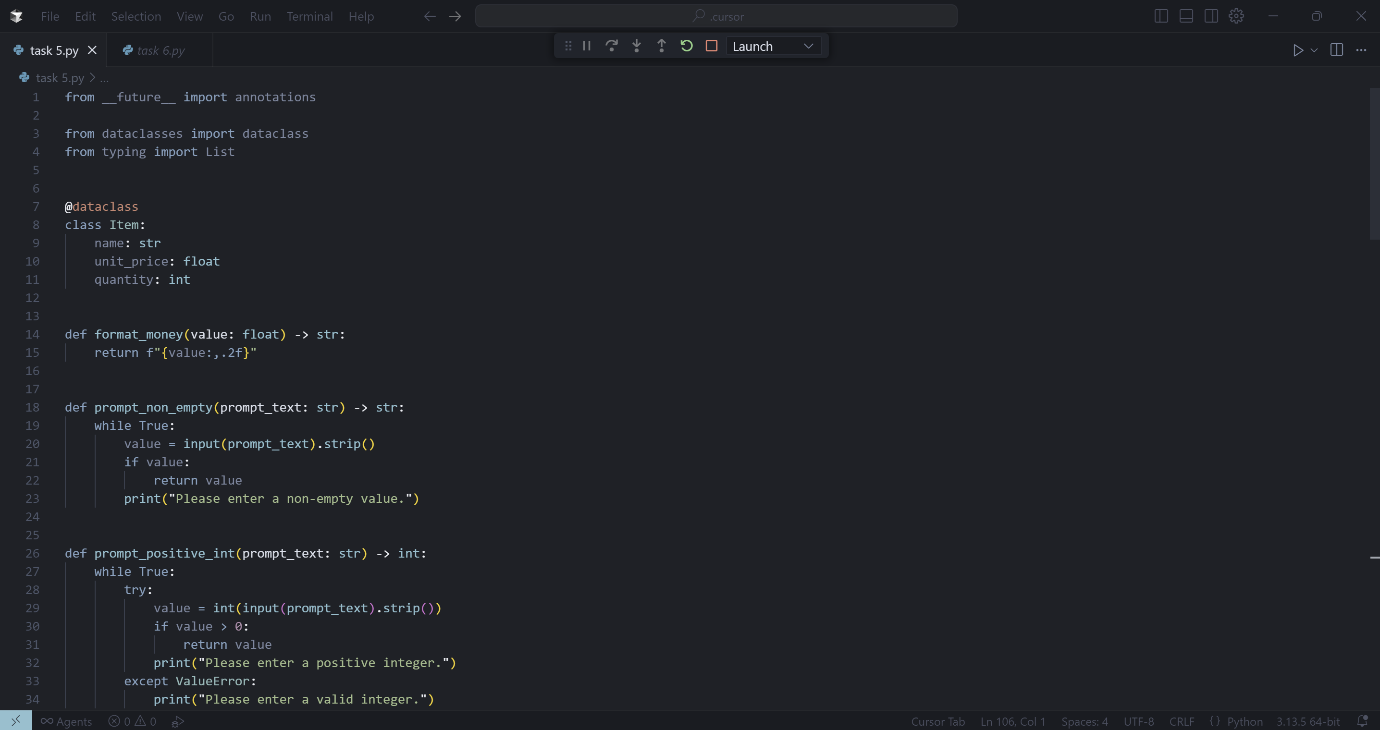


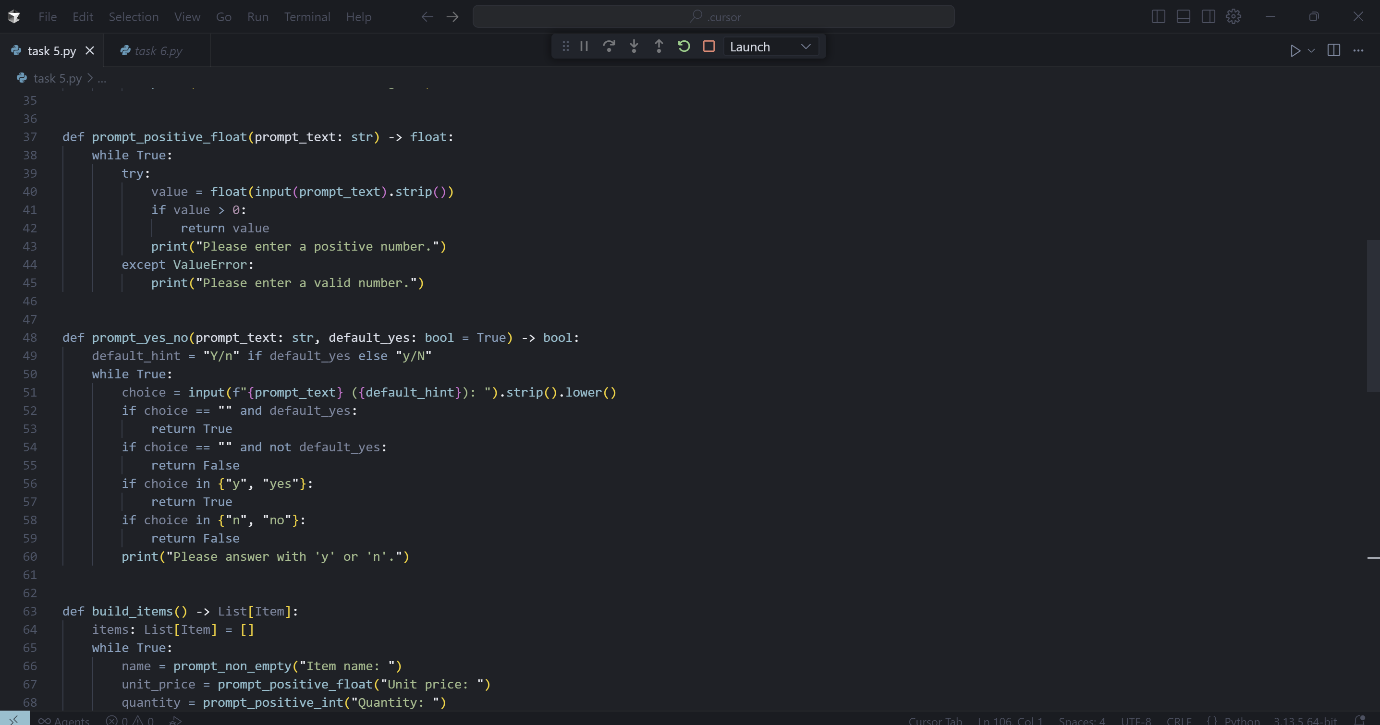


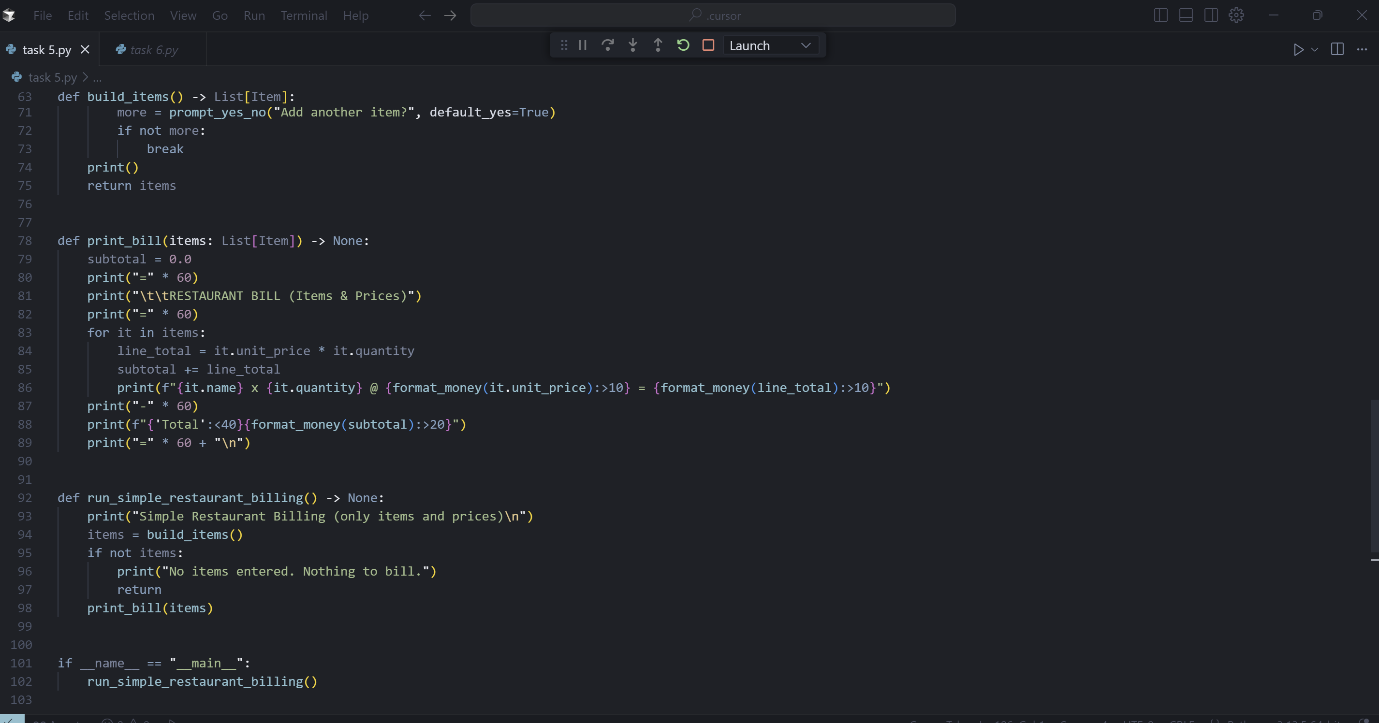
Task-4:

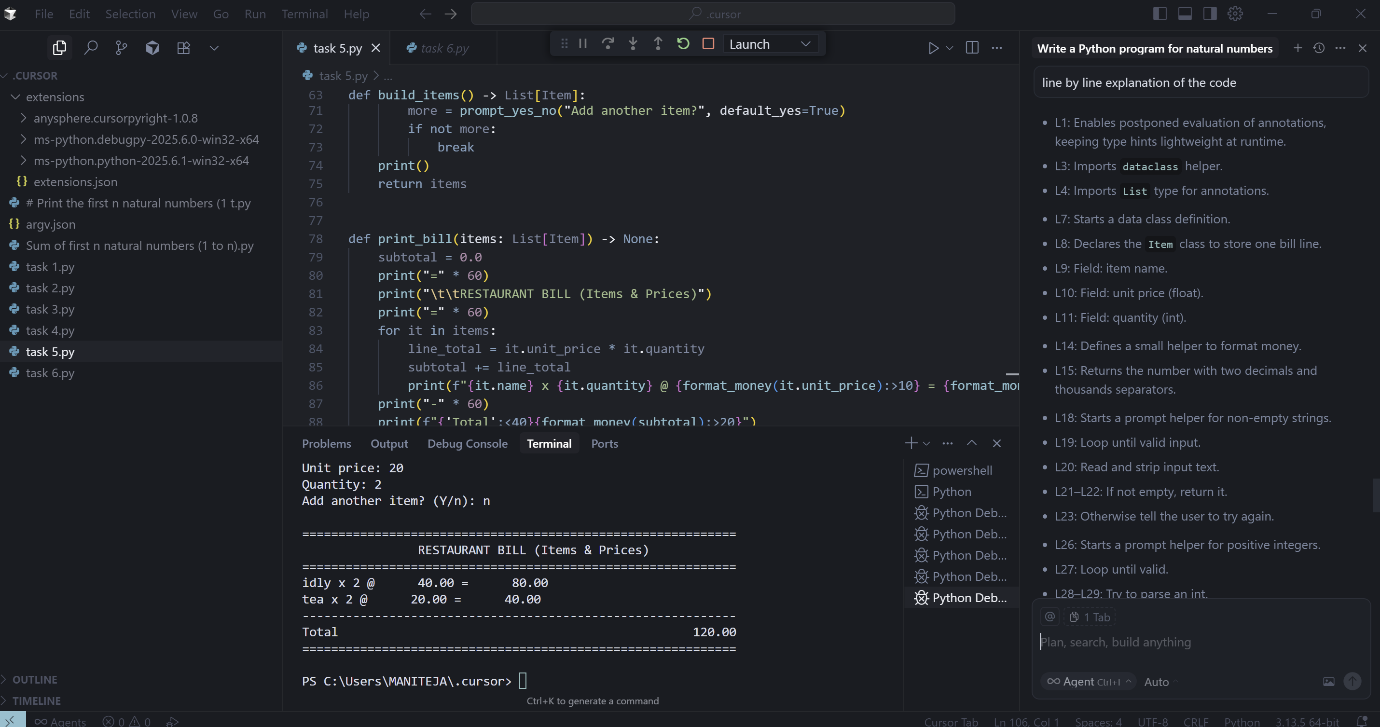
To generate hotel bill based on customer requirements

Code:







Output:

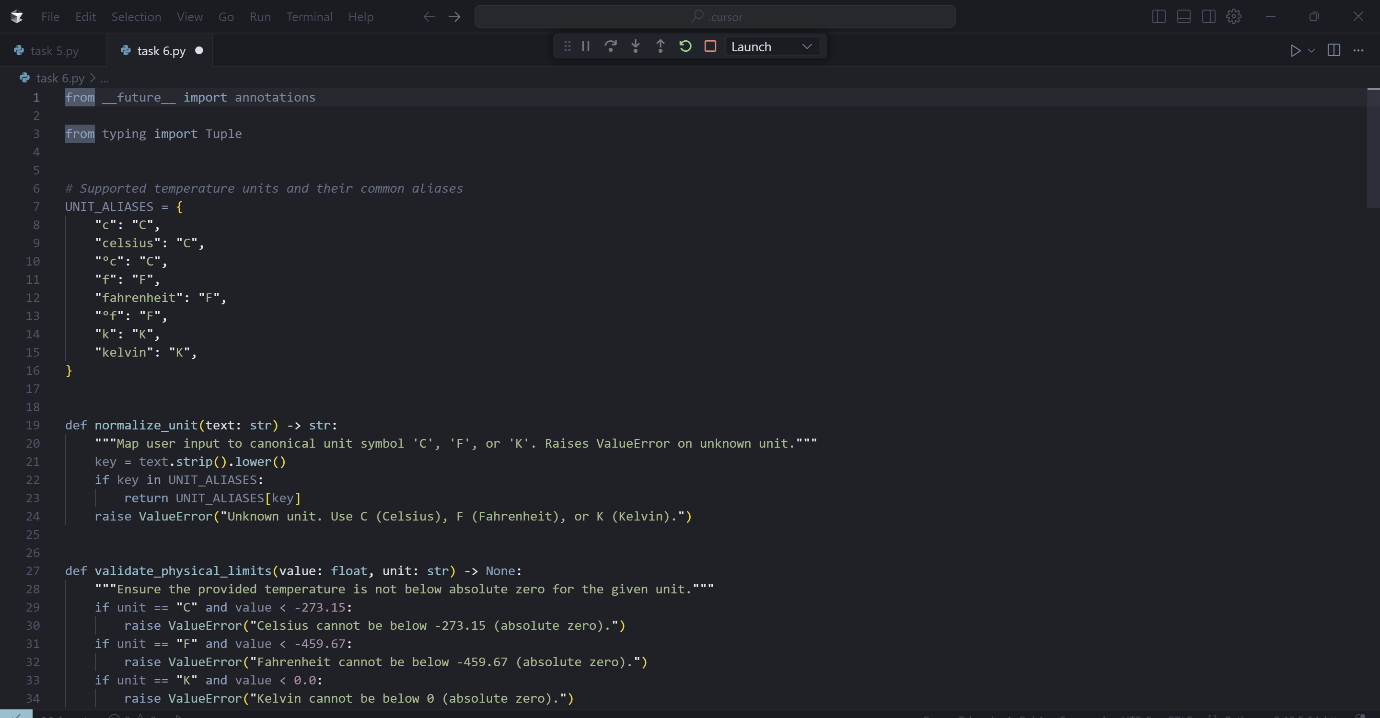
Explanation:

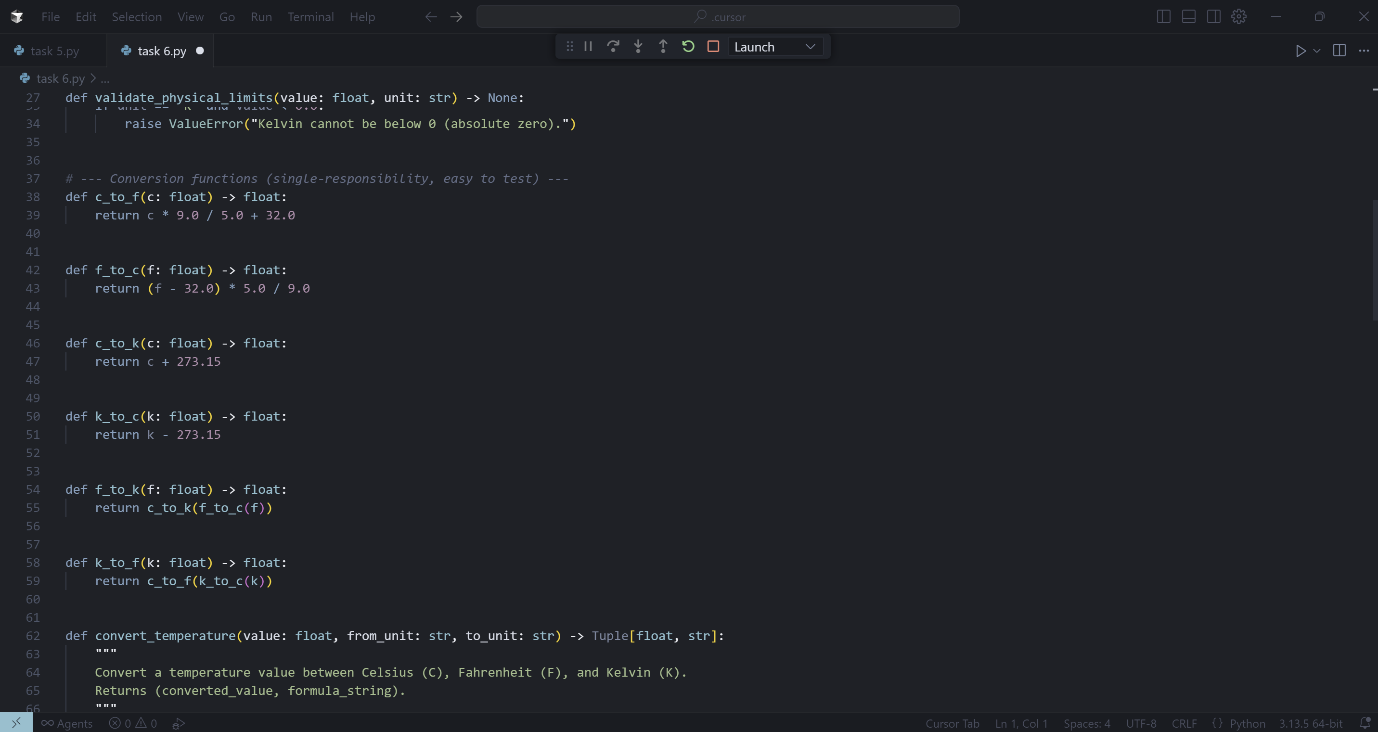
* L1: Enables postponed evaluation of annotations, keeping type hints lightweight at runtime.
* L3: Imports dataclass helper.
* L4: Imports List type for annotations.
* L7: Starts a data class definition.
* L8: Declares the Item class to store one bill line.
* L9: Field: item name.
* L10: Field: unit price (float).
* L11: Field: quantity (int).
* L14: Defines a small helper to format money.
* L15: Returns the number with two decimals and thousands separators.
* L18: Starts a prompt helper for non-empty strings.
* L19: Loop until valid input.
* L20: Read and strip input text.
* L21–L22: If not empty, return it.
* L23: Otherwise tell the user to try again.
* L26: Starts a prompt helper for positive integers.
* L27: Loop until valid.
* L28–L29: Try to parse an int.
* L30–L31: If > 0, return it.
* L32: Otherwise, instruct to enter positive integer.
* L33–L34: If parsing fails, show a validation message.
* L37: Starts a prompt helper for positive floats.
* L38: Loop until valid.
* L39–L40: Try to parse a float.
* L41–L42: If > 0, return it.
* L43: Otherwise, instruct to enter a positive number.
* L44–L45: If parsing fails, show a validation message.
* L48: Starts a yes/no prompt helper.
* L49: Shows default choice hint (“Y/n” or “y/N”).
* L50: Loop until valid.
* L51: Read answer (lowercased).
* L52–L55: Empty answer returns default.
* L56–L57: “y/yes” → True.
* L58–L59: “n/no” → False.
* L60: Otherwise, ask again.
* L63: Begins building the list of items interactively.
* L64: Initialize empty list.
* L65: Loop for multiple items.
* L66: Ask for item name (non-empty).
* L67: Ask for unit price (positive float).
* L68: Ask for quantity (positive int).
* L69: Append an Item to the list.
* L71: Ask if another item should be added (default yes).
* L72–L73: If not, break out.
* L74: Print a blank line for spacing.
* L75: Return the built items list.
* L78: Function to print the bill.
* L79: Initialize subtotal.
* L80–L82: Print header and title.
* L83: Iterate over items.
* L84: Compute line total = price × qty.
* L85: Accumulate subtotal.
* L86: Print formatted line: name, qty, unit price, line total.
* L87: Print a separator.
* L88: Print the final total, aligned.
* L89: Print closing line and a trailing newline.
* L92: Main UI function entry point.
* L93: Print a short description.
* L94: Collect items interactively.
* L95–L97: If none entered, show message and exit.
* L98: Otherwise, print the bill.
* L101: Standard “main” guard.
* L102: If run as a script, start the UI.

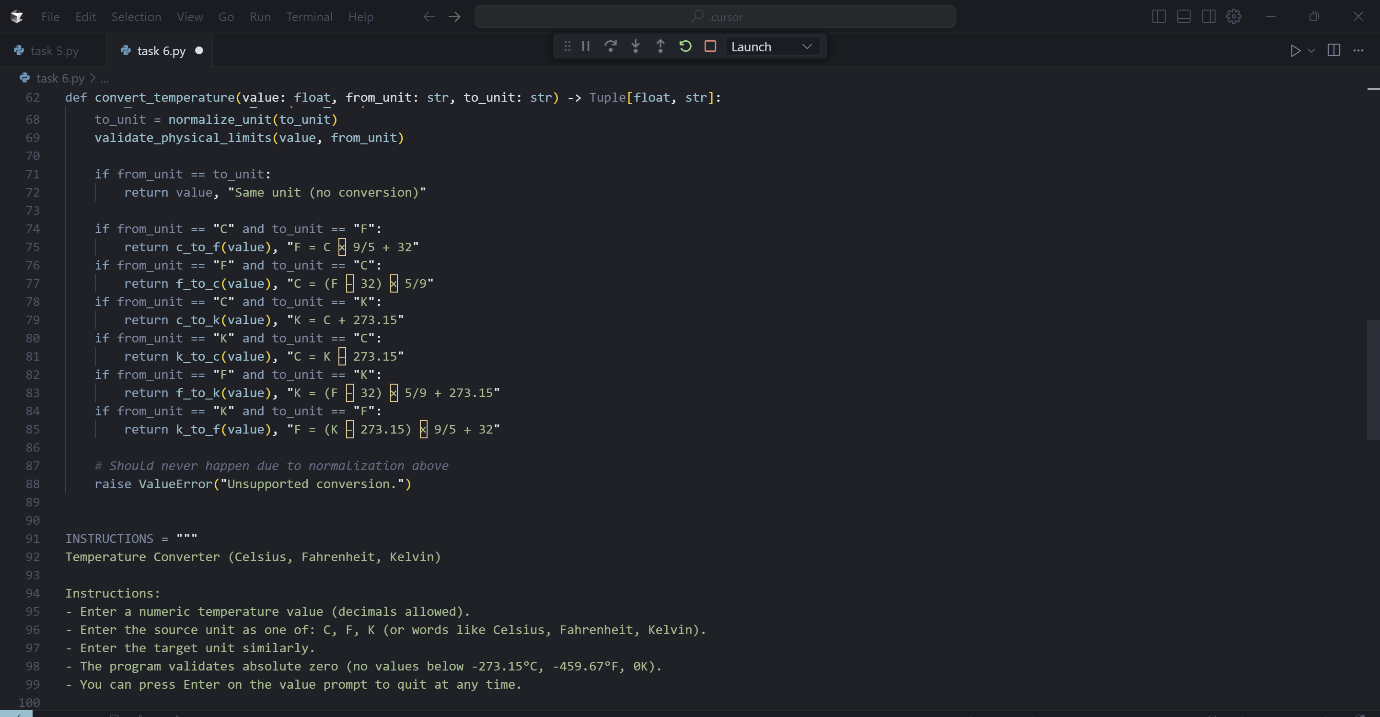
Task-5:

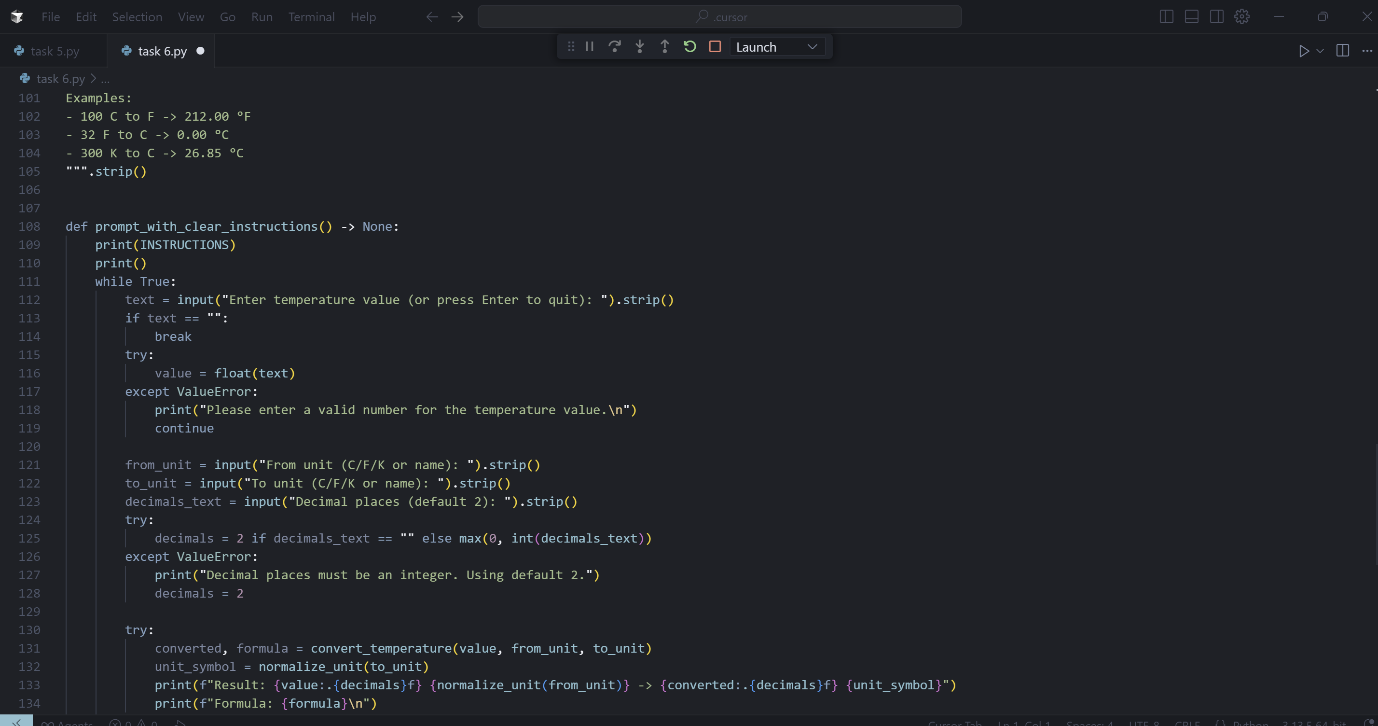
To Improving Temperature Conversion Function with Clear Instructions

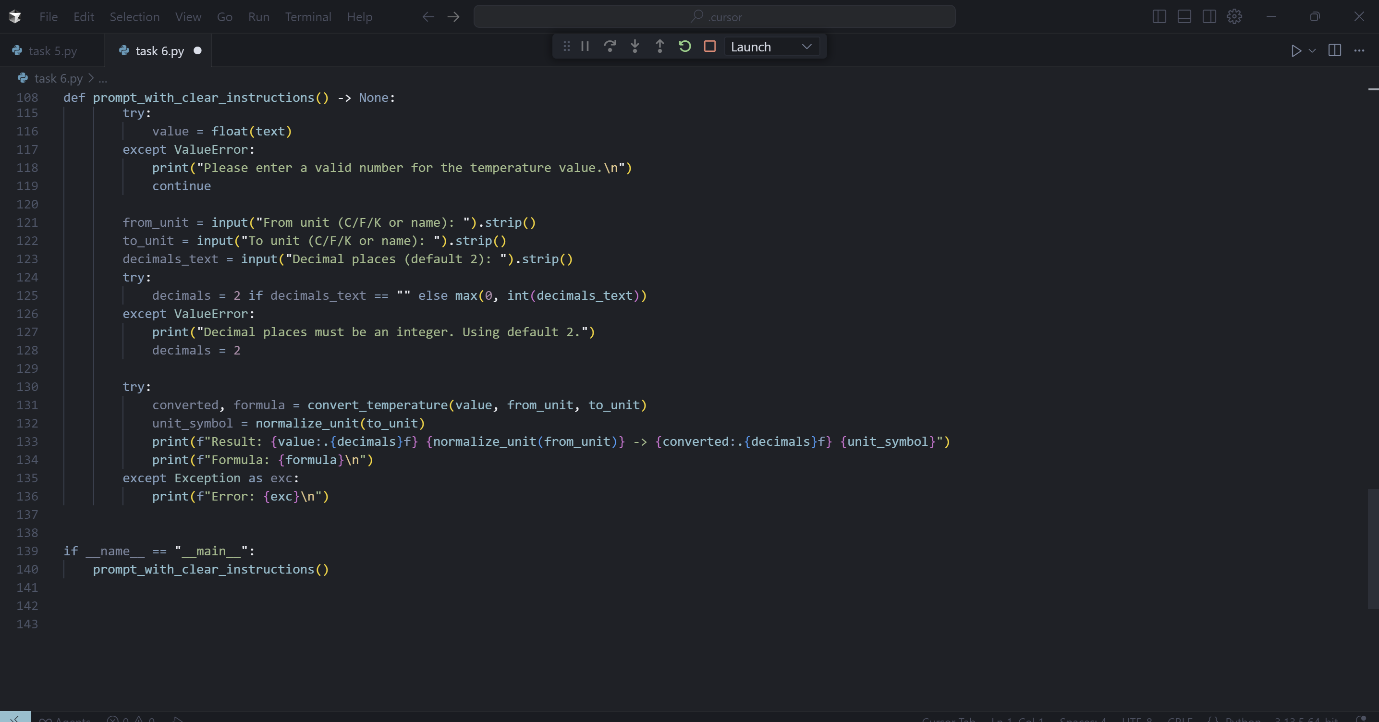
Code:

****

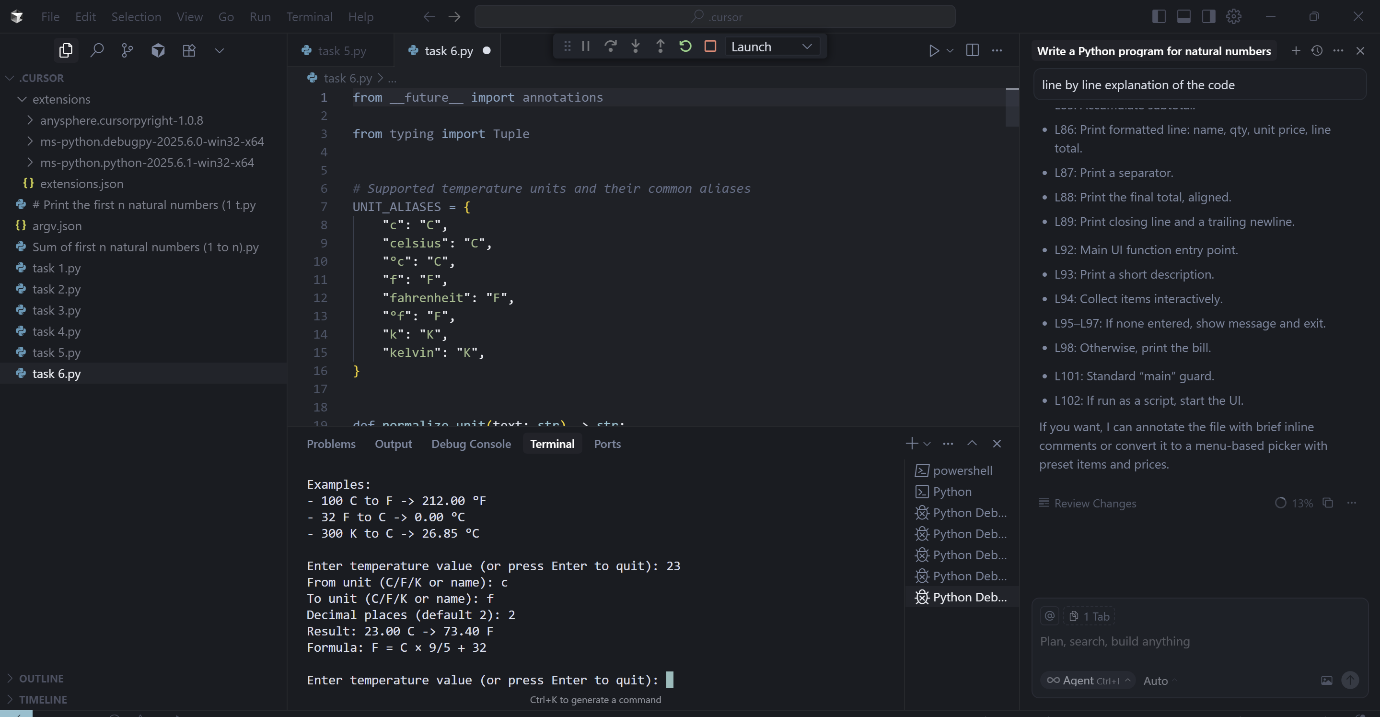
****







Output:



Explanation:

* L1: Enables postponed evaluation of type annotations for better forward-ref support.
* L3: Imports Tuple for typed return values.
* L6: Comment describing supported units and aliases.
* L7–L16: UNIT\_ALIASES maps various user inputs (like “celsius”, “°f”) to canonical symbols C, F, K.
* L19: Defines normalize\_unit(text) to canonicalize unit input.
* L20: Docstring explaining behavior and errors.
* L21: Normalizes input by trimming and lowercasing.
* L22–L23: If the normalized key exists, return its canonical symbol.
* L24: Otherwise raise a helpful ValueError.
* L27: Defines validate\_physical\_limits(value, unit) to enforce absolute zero.
* L28: Docstring.
* L29–L34: For each unit, raises ValueError if below absolute zero (−273.15 C, −459.67 F, 0 K).
* L37: Comment header for conversion helpers.
* L38–L39: c\_to\_f: Celsius to Fahrenheit.
* L42–L43: f\_to\_c: Fahrenheit to Celsius.
* L46–L47: c\_to\_k: Celsius to Kelvin.
* L50–L51: k\_to\_c: Kelvin to Celsius.
* L54–L55: f\_to\_k: Fahrenheit to Kelvin via F→C→K.
* L58–L59: k\_to\_f: Kelvin to Fahrenheit via K→C→F.
* L62: Defines convert\_temperature(value, from\_unit, to\_unit) -> Tuple[float, str].
* L63–L66: Docstring noting conversions and that it returns the value plus the formula string.
* L67–L68: Canonicalize both units using normalize\_unit.
* L69: Enforce absolute-zero for the input side using validate\_physical\_limits.
* L71–L72: If both units are identical, return the original value and a note.
* L74–L85: Route each supported conversion to the right helper and return both result and human-readable formula.
* L87–L88: Fallback error (shouldn’t occur due to earlier normalization).
* L91–L105: INSTRUCTIONS multi-line string: title, usage steps, absolute-zero note, and examples; then stripped of leading/trailing whitespace.
* L108: Defines prompt\_with\_clear\_instructions() (interactive UI).
* L109–L110: Print instructions and a blank line.
* L111: Start an input loop.
* L112: Prompt for temperature value or Enter to quit.
* L113–L114: Quit if blank input.
* L115–L119: Try to parse the numeric value; on failure, show message and continue loop.
* L121–L122: Prompt for source and target units (free-form; aliases accepted).
* L123: Prompt for decimal places (optional).
* L124–L128: Parse decimal places; default to 2 and clamp to non-negative; on error, warn and keep default.
* L130: Begin error-handled conversion block.
* L131: Convert using convert\_temperature.
* L132: Normalize the target for display.
* L133: Print the formatted result using chosen decimal places.
* L134: Print the human-readable formula used.
* L135–L136: On any error (e.g., bad unit or below absolute zero), show message and continue.
* L139: Standard main-guard for script execution.
* L140: When run directly, start the interactive prompt