UVSim Class Document

Currently our UVSim code is one big class called UVSim

Class Structures(UVSim):

- Stores in Memory
- Runs Program
- Gets the input from users
- Show output
- Loads Program

Properties (Instance Variables):

- Memory: Lists of 100 integers starts at zero
  - Stores program instructions and data
  - Valid 0-99
  - Valid numbers: -9999 to 9999

- Accumulator : this is what stores the numbers
  - Holds onto results from math problems
  - Will be used over and over for calculations

- Instruction_pointer: is the integer
  - Points to current Instruction address
  - this ranges from 0-99
  - starts at 0

- Running: boolean
  - show whether program is executing
  - set to true when program is running
  - set to false by HALT instruction

- Input_function
  - this function is used for READ operations
  - Defaults to _default_input if not provided
  - can be swapped out for testing

- output_function
  - this function displays the results to the user
  - can also be swapped out for testing
  - used for WRITE operations

Methods

- __init__
  - Runs automatically when you create new UVSim
  - Sets everything to zero
  - Connects all input and output functions

- Set_memory
  - stores values at specific memory locations
  - validate the address range 0-99
  - uses IndexError for invalid address
  - uses Value Error for values out of range

- get_memory
  - Retrieves the value from specific memory address
  - validates address range 0 - 99
  - uses IndexError for invalid address
  - returns an integer value

- _truncate_to_word
  - Makes sure the value is within 4 digit range
  - Keeps the sign of the value
  - this is used after operations are done to prevent overflow

- _default_input
  - prompt the user for input of integer
  - validates range -9999 - 9999
  - loops until a valid number is typed in the input
  - ValueError for non integer input
  - Returns the valid integer

- _default_output
  - prints value to the console

Program Loading

- load
  - Resets memory, accumulator, and instructor pointer to initial value
  - Accepts a list of strings and integers
  - Validate each line of commands using regex pattern
  - Requires 1-4 digits
  - Handles empty strings as zero
  - IndexError if more than 100 lines
  - ValueError for bad inputs
  - TypeError for bad datatypes

Program Execution

- run
  - sets running to true
  - continues until HALT instruction or an error occurs
  - runs instructions one by one until the program stops
  - starts at instruction 0

- _execute
  - Looks at instruction number
  - Splits into two parts
    - Opcode reads first 2 digits - what to do
    - Operand reads last 2 digits - where to do it
  - Example instruction 1020 opcode = 10 operand = 20
  - What it does depends on opcode

How everything works together

1. Load a program: instructions put into memory boxes
2. Starts Running: starts at instruction 0
3. Repeat these steps
   - Read instructions
   - Read opcode + operand
   - Do what instructions are given
   - Move to next instruction
4. Stops when you hit HALT instruction

Errors that could happen

- IndexError - tried to use a memory box that does not exist
- ValueError - tried to store number that is to big or too small
- TypeError - You put something in the input that is not the correct type of value
- ZeroDivisionError - trying to divide by zero
- RuntimeError - used an instruction code that doesn't exist