This code will be expanded as part of a future lab! Plan ahead!

Evaluating Multiple Assignment Statements

like a mini programming language

You are **NOT** allowed use eval() or exec(). You will **fail** the assignment if you do.

Overview

- Write a webpage and/or CGI-Script(s) that work together to parse and correctly evaluate a series of assignment expressions.
- Turn in the python code you've written, as multiple files, making sure to put the Parkland Pledge at the top with your name and any sites you've taken code from. Since there are so many sites that do similar things, your code **MUST** implement the solution as I have given it.
- Put a link to your working cgi script in the message area.

You are **NOT** allowed use eval() or exec(). You will **fail** the assignment if you do.

Input requirements

- You need to implement the single line "**`DUMP`**" (all caps, no quotes) which will print the variables that are stored in memory.
- You need to implement the single line "**`NAME`**" (all caps, no quotes) which will print your name.
- Everything else will be an assignment expression to be evaluated, containing the operator, a value and a variable.
- The assignment operator may or may not have a space around it: `"x := 45.6"` or `"x:=45.6"` or `"x:= 45.6"` or `"x :=45.6"` all work.
- **Note that the lab must use := for assignment. It is two characters long. If it works with just =, you will lose points so don't so the extra work to make that happen.**

What operators you need to implement

- Assume every variable and constant is a floating point number (no integers or bool or strings)
- Assignment operator:
  - := → Assignment: number := 7.1 will store the value of 7.1 into the variable 'number'. Assignment will always be the first operator and there will never be more that one.
- More operators will be added on a later assignment.

You are **NOT** allowed use eval() or exec(). You will **fail** the assignment if you do.

How to handle variables

https://repl.it/@kurbanParkland/dictionaryExample

- Variables ALWAYS begin with a lowercase letter. If the first letter of the part you're dealing with is lowercase, that's what you have. Any variable names can work. Variable names only contain letters and numbers, never spaces.
- You'll maintain a table of variables and their values. (Python calls this a **dictionary**) http://openbookproject.net/thinkcs/python/english3e/dictionaries.html
- When you encounter an assignment (:=), the right hand side could be a value or a variable.
  - If it's a value (begins with a number) store that value in the table under the left hand side variable name.
  - If it's a variable (begins with a lower case letter), look up the variable. If the variable has a value, store it in the table, otherwise throw an error.

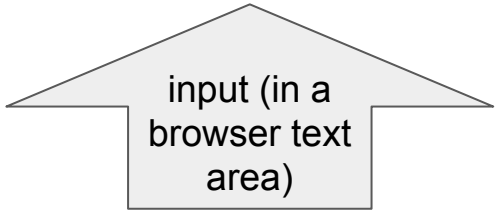
Output

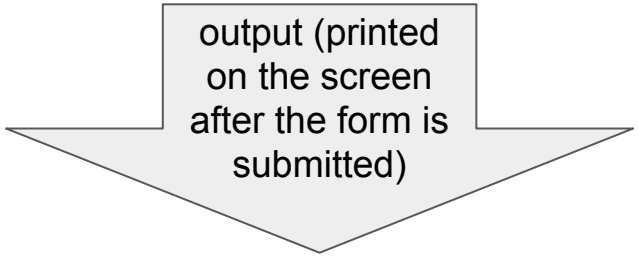
- "DUMP" and "NAME" are the only things that have to print anything. DUMP will look better in a table.
- If you want to print debugging with assignment that's fine, just begin it with the word DEBUG and I'll ignore it.
- You are to throw an error when something 'bad' happens, like trying to pop an empty stack or visiting an unassignment variable. (Your errors don't have to match my suggestions)

You are **NOT** allowed use eval() or exec(). You will **fail** the assignment if you do.

Example:

```
val1 := 100
val2 := 300
DUMP
val2 := 200
DUMP
NAME
```

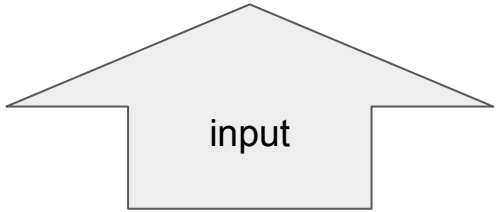


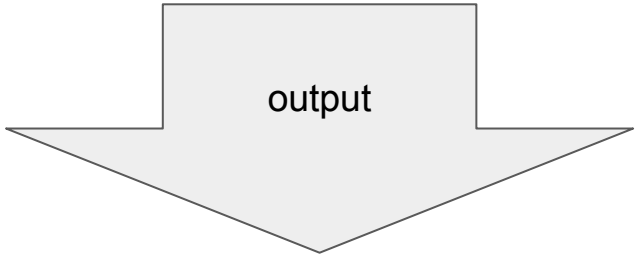



```
Printing all variables:
val1 100
val2 300
Printing all variables:
val1 100
val2 200
Ken Urban
```

Example:

```
val1 := 100
val2 := val1
val1 := 200
val3 := val1
DUMP
NAME
```

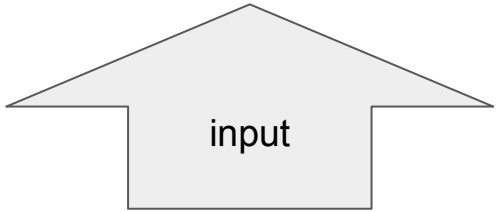


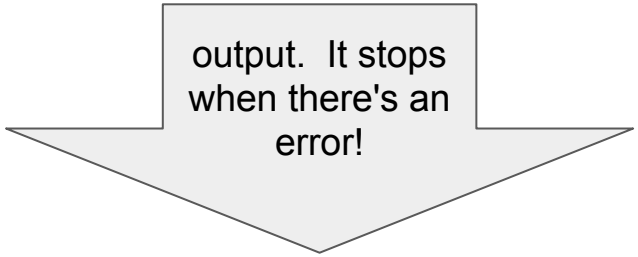



```
Printing all variables:
val1 200
val2 100
val3 200
Ken Urban
```

Example:

```
val1 := 100
val2 := val7
NAME
```






```
Error: val7 not defined
```

My thoughts on how to proceed

1. As always, set up the html and the python CGI-Script to work
2. Make the python code print the input
3. Make the python code print the input line by line (split on newlines)
4. Have the python code identify **`DUMP, NAME`** and **`var :=`**.
5. Get **`NAME`** to work.
6. Break up **`var :=`** into its pieces.
7. get lines like **`var := 56.3`** and **`DUMP`** to work together using a dictionary
8. determine if the right hand side is a value or a variable.

Consider making a 'Variable_table' class to do all the work of variables.

SHORTCUTS IN PYTHON THAT WILL CAUSE SECURITY ISSUES!

You are **NOT** allowed use eval() or exec(). You will **fail** the assignment if you do.

Python is interpreted. (Running and compiling at the same time). This allows for strings to be compiled and run.

You can take input, and run the input directly. exec("x = 345")

You can take input from the internet, and run the input directly on www.csit.parkland.edu.

You can take input from **ANYONE** ON the internet, and run the input directly on www.csit.parkland.edu.

Bad idea. exec() is BAD. eval() is BAD with ANY user input. This is a security issues.

You are NOT allowed use eval() or exec(). You will fail the assignment if you do.