

Variables and Memory Cells

Computers understand numbers, not words. 0s and 1s. But 0s and 1s are difficult for people.

High-level programming languages allow people to use symbolic names instead of just 0s and 1s.

A computer's memory is a contiguous sequence of slots, or memory cells. Data can be stored in memory cells. Each cell has a numeric address.

A **variable** is the name of a memory cell.

We call it a "variable" because the value in the cell can change.

Python and other high-level languages use the **symbol table** to map a variable name to the memory cell address it represents.

The program statement:

```
money=10000
```

causes the number 10000 to be placed in the memory cell represented by the variable name 'money'. The equals sign means "is assigned the value". Its called the *assignment operator*.

The statement

```
money=money + 1
```

is one that can mystify beginning programmers. It means:

the memory cell money is assigned its current value plus one.

The right-hand side of the assignment operator '=' is *evaluated*, then placed in the memory cell denoted by the left-side variable.

Here's a simplified example of what happens when a program is executed:

Program

```
Money=10000
interestRate = 0.2
oneYearReturn=money*interestRate
```

Main Memory

Address	Value
0	10000
4	0.2
12	2000
...	

Symbol Table

money	0
interestRate	4
oneYearReturn	12

The program puts data in contiguous memory. So the variable 'money' is at address 0, 'interestRate' at address 4, and oneYearReturn at address 12.

In-Class Questions

1. Why are the addresses in the table labeled 0,4,12 instead of 0,1,2,...

2. Given the 'trace' shown above as example, show the main memory and symbol table after execution of the following:

```
name="joe"
x= 92
y = x+77
```

3. The main memory shown is too simplistic. In a real program, variable storage would not begin at address 0. How is memory for real computer applications organized?