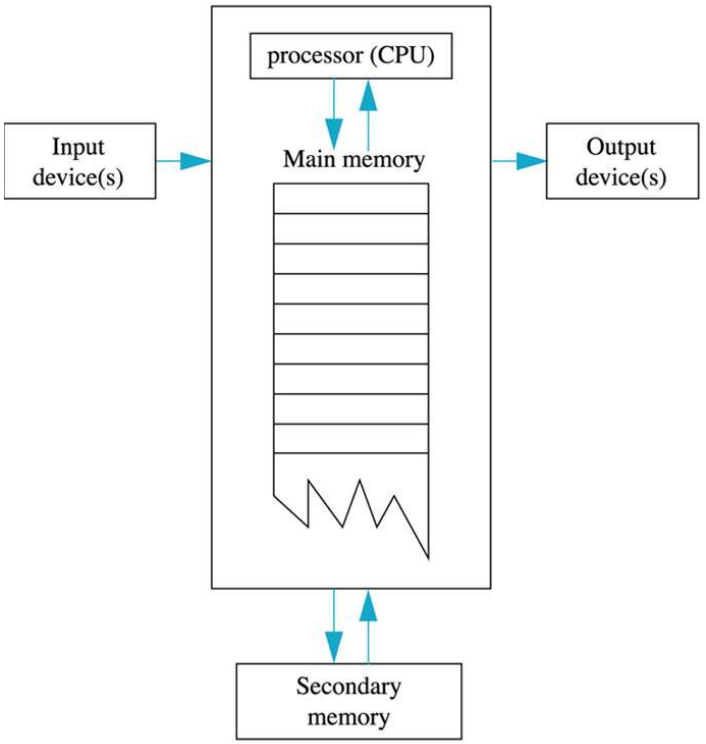
What is a Computer?

Hardware

Main Components of a Computer



Software

- A collection of computer programs.
  - E.g. Operating system

Computer Program

- Sets of instructions that control computer's processing data.

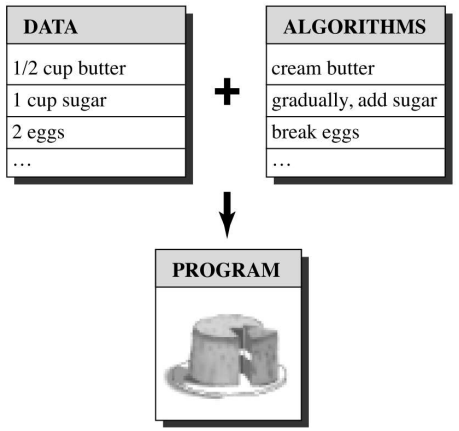
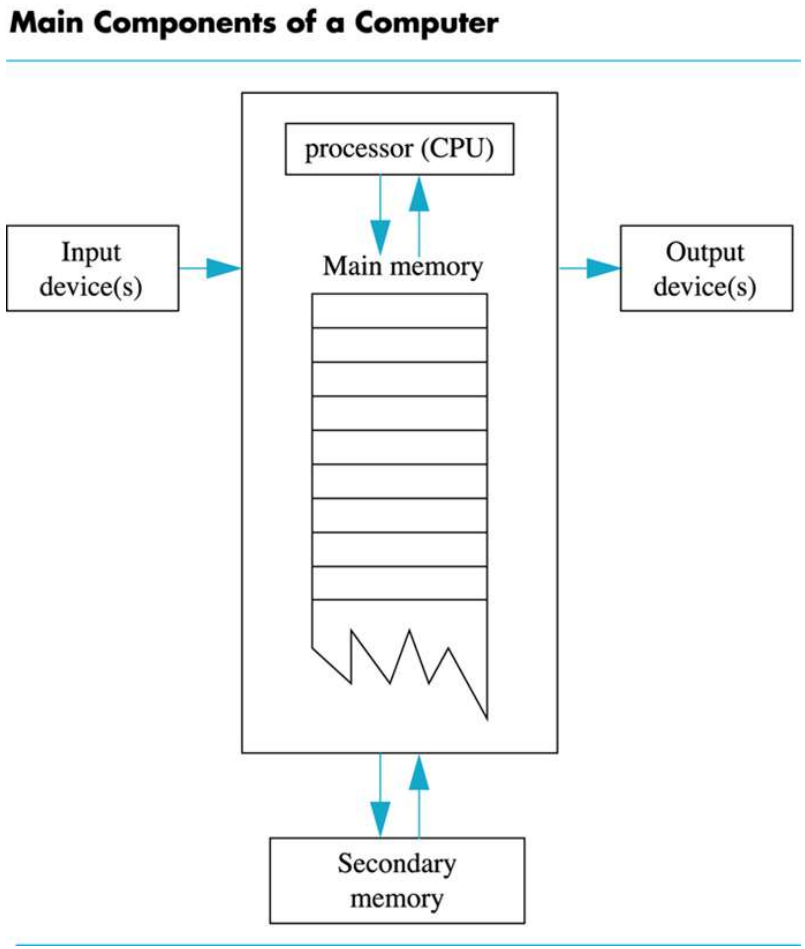


Figure 1.1 Data + algorithms = program.

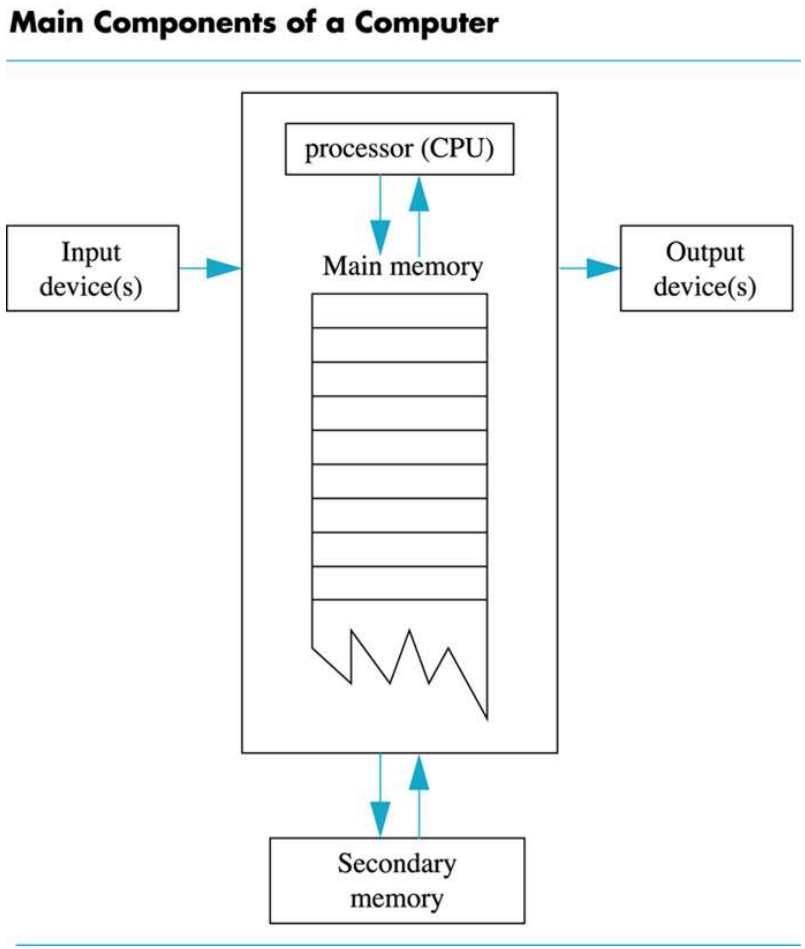
Computer Hardware

- Input devices
  - Allow communication to the computer
  - Mouse, keyboard, microphone,…
- Output devices
  - Allow communication to the user
  - Monitor, printer,…
- Processor (CPU)
  - Brain of the computer
  - Perform simple instructions (add, subtract, multiply, divide, move data from location to location)



Computer Hardware

- Main memory
  - Like a scratch of paper for CPU
  - Used to load and run programs.
  - Random access (fast)
  - Contains bit (binary digit)
    - A digit that can only be 0 or 1.
    - 8 bits = 1 byte.
  - Each memory location has an address
- Secondary memory
  - Sequential access (slower)
  - More permanent storage



Computer Hardware

What kind of device is a touch-enabled display screen?

A. Input device

B. Output device

C. Both input and output

D. A computer by itself

Computer programs

- A program describes a set of procedures for the computer to follow to produce a particular outcome, much as a recipe prescribes a set of procedures for a cook to follow to produce a cake.

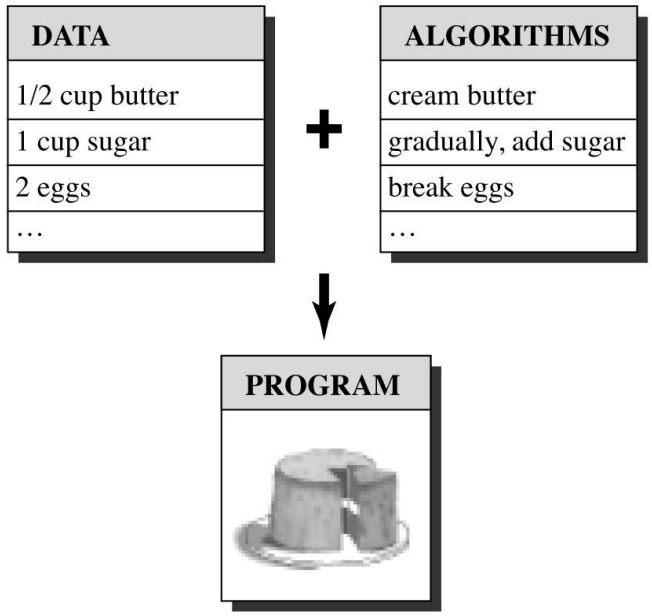


Figure 1.1 Data + algorithms = program.

Computer programs

- In the cooking analogy, which one correspond to the recipe?

  A. Computer

  B. Programmer

  C. Data

  D. Algorithm

Computer programs

- In the cooking analogy, which one correspond to the ingredients?

  A. Computer

  B. Programmer

  C. Data

  D. Algorithm

Computer programs

- In the cooking analogy, which one correspond to the cook?

  A. Computer

  B. Programmer

  C. Data

  D. Algorithm

Computer programs

- In the cooking analogy, which one correspond to the cook?

  A. Computer

  B. Programmer

  C. Data

  D. Algorithm

Who is the one who wrote the recipe?

Problem Solving

- The most important skill for a computer scientist or even a programmer is problem solving
  - The ability to formulate problem
  - Find a solution
  - Express the solution clearly and accurately (the result will be the algorithm)
- Algorithm
  - A step by step list of instruction (if followed exactly will solve the problem)
  - Once we find this solution, computer can be used to automate the execution

Programming Languages

- We use them to create programs
  - Just like English is used to write the recipe.

Programming Languages

- High level programming language:
  - Resemble human languages and uses math-like operations
  - Are designed to be easy to read and write
  - Example: `balance = credit - debit`
  - high level programming languages: `C++, Java, Python,` …

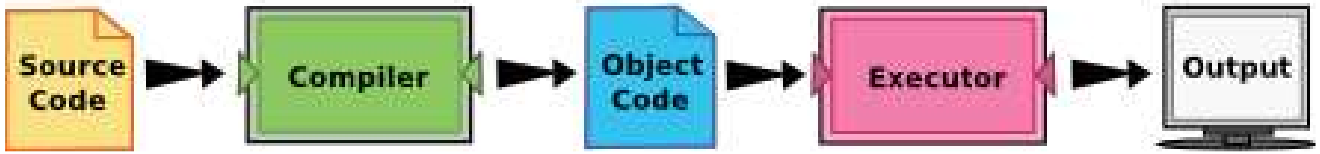
Programming Languages

- High level programming language:
  - Resemble human languages and uses math-like operations
  - Are designed to be easy to read and write
  - Example: `balance = credit - debit`
  - high level programming languages: `C++, Java, Python,` …
- Low level programming languages:
  - Assembly languages `SUBTRACT d c b`
  - Machine-level languages `0100  1001  1010  1011`
    - Hardware dependent
  - Computers can only execute programs written in machine languages

Programming Languages

- High level language should be translated to machine level
- Two kinds of programs process high-level languages into low-level languages:
  - Interpreters: reads a high-level program and executes it.

  - Compilers: reads the program and translates it completely before the program starts running.
    - Source code: The original program in a high level language
    - Object code: The translated version in machine language



Python

- Python uses both processes (interpret and compile)
  - Because of the way programmers interact with it, it is usually considered an interpreted language.
- Two ways to use Python interpreter
  - Shell mode: type Python expressions into the Python shell, and the interpreter immediately shows the result
  - Program mode: write your program in Python, store it and run it (command line or in a GUI)

Review Questions?

- List the five main components of the computer?
  - Input devices, output devices, CPU, memory, secondary memory
- Describe the work of a compiler and interpreter? What is the difference?
- Define source code? Define object code?

Review questions

- Source code is another name for:

  A. the instructions in a program, stored in a file.

  B. the language that you are programming in (e.g., Python).

  C. the environment/tool in which you are programming.

  D. the number (or "code") that you must input at the top of each program to tell the computer how to execute your program.

Review questions

- What is the difference between a high-level programming language and a low-level programming language?

  A. It is high-level if you are standing and low-level if you are sitting.

  B. It is high-level if you are programming for a computer and low-level if you are programming for a phone or mobile device.

  C. It is high-level if the program must be processed before it can run, and low-level if the computer can execute it without additional processing.

  D. It is high-level if it easy to program in and is very short; it is low-level if it is really hard to program in and the programs are really long.

Review questions

- Pick the best replacements for 1 and 2 in the following sentence: *When comparing compilers and interpreters, a compiler is like **1** while an interpreter is like **2**.*

  A. 1 = a process, 2 = a function

  B. 1 = translating an entire book, 2 = translating a line at a time

  C. 1 = software, 2 = hardware

  D. 1 = object code, 2 = byte code

A few definitions…

- **Bug :** A mistake (error in a program).

- **Debugging**: Eliminating mistakes in programs.

Types of errors

- Syntax errors
  - Violation of the grammar rules of the language
  - Discovered by the compiler or interpreter
    - Error messages may not always show correct location of errors
- Runtime errors
  - Error conditions detected by the computer at run-time
    - Example: running out of memory or division by zero
- Logic errors (semantic errors)
  - Errors in the program's algorithm
  - Most difficult to diagnose
  - Computer does not recognize a logical error

Types of errors

- What kind of error is produced if you forget a punctuation symbol such as a colon at the end of a statement where one is required?
  A. Syntax
  B. Runtime
  C. Logical
  D. Numerical

Types of errors

- What kind of error is produced when program runs but produces incorrect results?
  A. Syntax
  B. Runtime
  C. Logical

Types of errors

- Which kind of errors are detected by the compiler/ interpreter?
  A. Syntax
  B. Runtime
  C. Logical
  D. Numerical

Types of errors

- Which of the following is a run-time error?

  A. Attempting to divide by 0.

  B. Forgetting a colon at the end of a statement where one is required.

  C. Forgetting to divide by 100 when printing a percentage amount.

Types of errors

- Which of the following is a semantic error?

  A. Attempting to divide by 0.

  B. Forgetting a colon at the end of a statement where one is required.

  C. Forgetting to divide by 100 when printing a percentage amount.

Python Basics

A simple program: Hello world!

- Hello world! is the first program written in a new language

```
print “Hello world!”
```

```
Hello world!
```

A simple program: Hello world!

- Compare it to a program (same functionality) in C++!!

print “Hello world!”

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Hello world!" << std::endl;
6      return 0;
7  }
```

```
Hello world!
```

A simple program: Hello world!

- Hello world! is the first program written in a new language

```
print “Hello world!”
```

```
Hello world!
```

  - This is an example of using the `print` function, which displays a value on the screen
    - Here the value is a phrase. The quotation marks in the program mark the beginning and end of the value. They don’t appear in the result.

Comments

- In Python # indicates that everything following it until the end of the line is a comment; it is ignored by the interpreter
- This is new version of `Hello, world!` program

```
# New version of Hello world!

print “Hello world!”  # very easy!!
```

```
Hello world!
```

- It still prints the same phrase
  - Comments and new lines are ignored by the interpreter but makes your program more readable for humans

Basics

- The fundamental building blocks of Python
  - Values and Data types
  - Variables
    - Assignment statements
  - Operations

Values and Data types

- A value is one of the fundamental things a program works with
  - A letter, a number, a phrase (sequence of letters)
    - ‘r’ , 3, “Hello, world!”
  - Values belong to different types
    - 3 is an integer, “Hello, world!” is a string (string are enclosed in quotation marks)
    - function print can print different types of values (try it!)
  - If you do not know the type of a value, Python has a function `type`
    - try in the shell mode: `type(“Hello, world!”)`
    - or use function print if you are not in interactive mode:
      `print type(“Hello, world!”)`

Values and Data types

- More examples:
  - `type(17)`
  - `type(3.4)`
  - `type("3.4")`
- Strings in Python can be enclosed in either single quotes (') or double quotes ("), or three of each (''' or """)
  - Try these in the shell mode:
    - `type('This is a string.')`
    - `type("And so is this.”)`
    - `type("""and this.""")`
    - `type('''and even this...''')`

Values and Data types

- Try this:
  - `print 1,111,111`
  - What do you expect? what is the output?

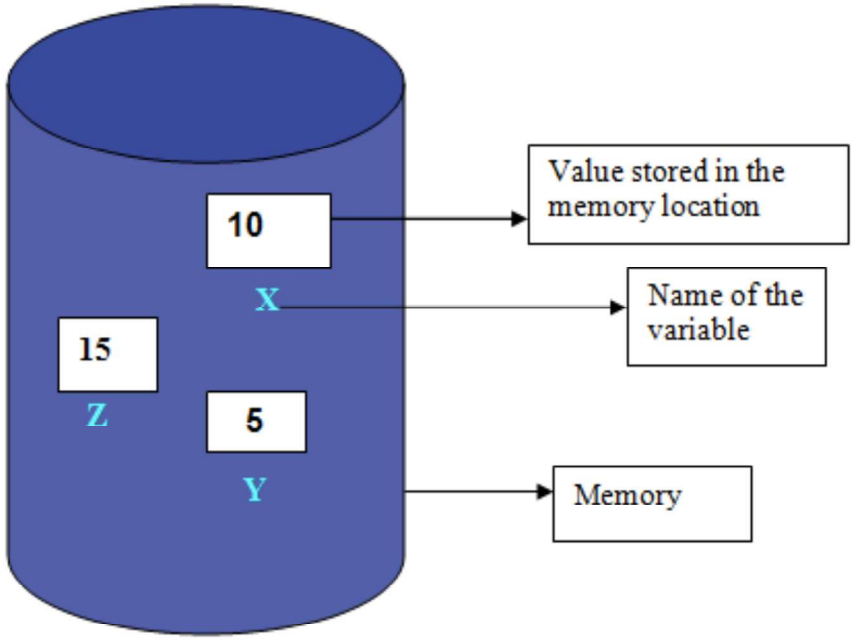
Values and Data types

- Try this:
  - `print 1,111,111`
  - What do you expect? what is the output?
- It is not a legal integer in Python, but it does mean something else which is legal!
- Python treat it as a triple of values
  - We will get back to it later
  - Just remember not to put comma or space in your integers

Variables

- In programming, a variable is a (symbolic) name representing some value (data) stored in computer memory.

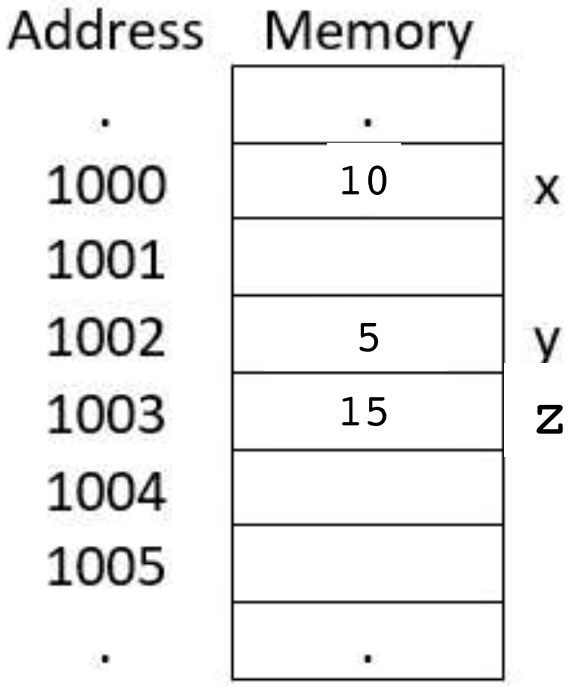
- Think of it as a location in the memory paired with an associate name (an identifier)
  - Variables are names for memory locations
  - We can write a value in them, change it later, ...



Variables

- In programming, a variable is a (symbolic) name representing some value (data) stored in computer memory.

- Think of it as a location in the memory paired with an associated name (an identifier)
  - Variables are names for memory locations
  - We can write a value in them, change it later, ...



Variables

- One of the most powerful features of a programming language is the ability to manipulate variables.
- In Python, an assignment statement creates new variables and gives them values
  - Try these in shell mode:
  - `message = "Python is really cool!"`
  - `x = 9`
  - `pi = 3.14159`
- To display the value of a variable you can use function print: `print pi`

Variables

- Type of variables is determined by the type of value it refers to
  - Check the type of variables we defined in previous slide (using function `type()`)

Variables

- Type of variables is determined by the type of value it refers to
  - Check the type of variables we defined in previous slide (using function `type()`)
- Note: in many other programming languages, you need to declare the type of variables before assigning any value to them
  - for example in C++ or Java:

```
int x;
float pi;
x = 17;
pi = 3.14159;
```

Variable Names

- In programming, variable names are called identifiers
- Choosing variable names
  - Use meaningful names that represent data to be stored
    - balance, debit, credit, acceleration, mass, time, rate, age, ...
  - String of characters (letters, digits, underscores)
    - Cannot begin with digit
    - Cannot contain spaces or punctuation marks or symbols
  - Case sensitive
    - rate, RATE, Rate : are not the same
- Keywords (reserved words) cannot be used as identifiers

Variable Names

- Python reserves 33 keywords:

and	**del**	**from**	**None**	**True**
as	**elif**	**global**	**nonlocal**	**try**
assert	**else**	**if**	**not**	**while**
break	**except**	**import**	**or**	**with**
class	**False**	**in**	**pass**	**yield**
continue	**finally**	**is**	**raise**	
def	**for**	**lambda**	**return**	

- If you give a variable an illegal name, you get a syntax error
- Note: Programmers generally choose names for their variables that are meaningful to the human readers of the program

Variable Names

- Mark the following identifiers as either valid or invalid.

a. theDog
b. all-In-One
c. const
d. recycling
e. DVD_ROM
f. elizabeth_the_2nd
g. 2morrow
h. Page#

Variable Names

- Which one is a valid variable name?
  A. 3com
  B. Three_com
  C. 3_com
  D. three-com
  E. dollar$

Statements

- A statement is a unit of code that is meaningful (computer can execute)

- When you type a statement in interactive mode, the interpreter executes it and displays the result, if there is one.
  - So far we have learned about:
  - print statement in Python which prints something in the output:

```
print x
print “Hello world!”
```

  - and assignment statement that assign a value to a variable:

```
message = “Python is really cool!”
x = 9
pi = 3.14159
```

Assignment statements

- An Assignment statement changes the value of a variable.

```
balance = 32
```

  - You are ordering the computer to “set the value of variable **`balance`** to 32”.
  - The single variable to be changed is always on the left of the assignment operator ‘=‘
  - On the right of the assignment operator can be
    - constant: `age = 21`
    - Variables: `my_saving = balance`
    - Expressions: `dist = (acceleration*time*time)/2`

(means compute `(acceleration*time*time)/2` and the assign it to variable `dist`)

Assignment statements

- What is the value of `big` and `small` after executing this code?

```
big = 10
small = 20
big = small
```

(A) big: 10 and small: 20

(B) big: 20 and small: 10

(C) big: 10 and small: 10

(D) big: 20 and small: 20

(E) big: 30 and small: 20

Assignment statements

- What is the value of `n` and `m` after executing this code?

```
m = 10
n = 20
m = n
n = m
```

(A) m: 10 and n: 20

(B) m: 20 and n: 10

(C) m: 10 and n: 10

(D) m: 20 and n: 20

(E) m: 30 and n: 20

Assignment statements

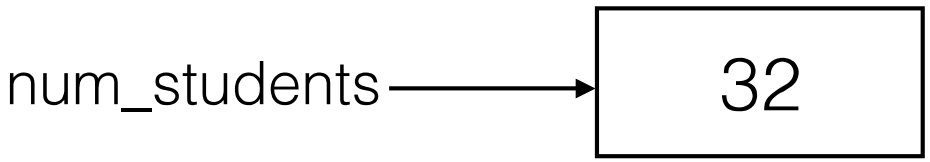
- The ‘=‘ operator in Python is not an equal sign.
  - The following statement cannot be true in algebra

```
num_students = num_students + 2
```

Assignment statements

- The ‘=‘ operator in Python is not an equal sign.
  - The following statement cannot be true in algebra

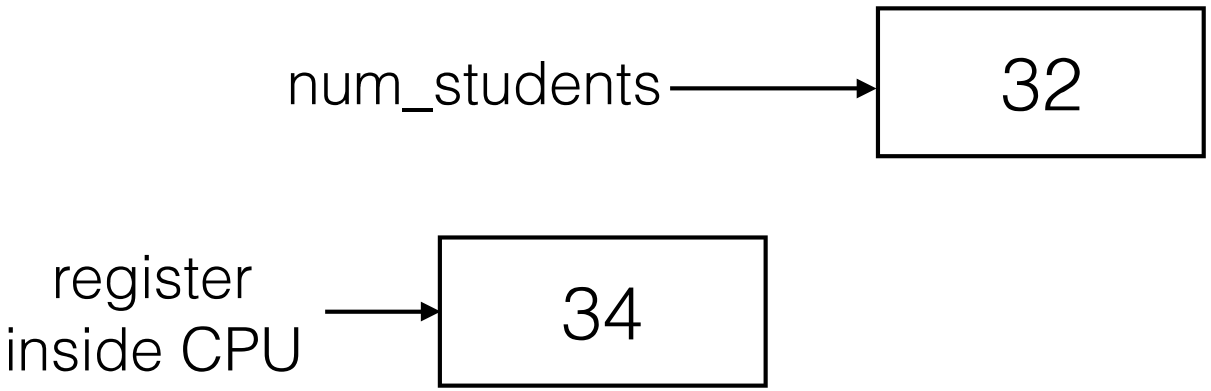
```
num_students = num_students + 2
```



Assignment statements

- The ‘=‘ operator in Python is <u>not</u> an equal sign.
  - The following statement cannot be true in algebra

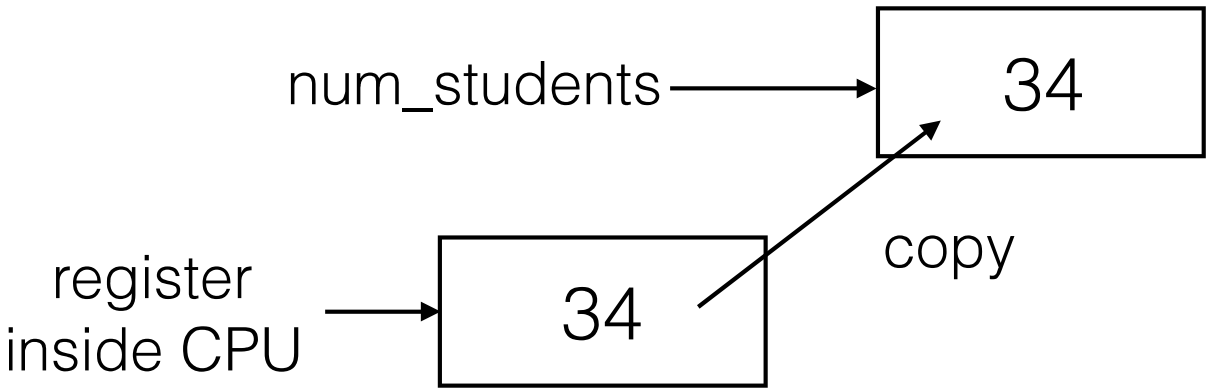
```
num_students = num_students + 2
```



Assignment statements

- The ‘=‘ operator in Python is <u>not</u> an equal sign.
  - The following statement cannot be true in algebra

```
num_students = num_students + 2
```

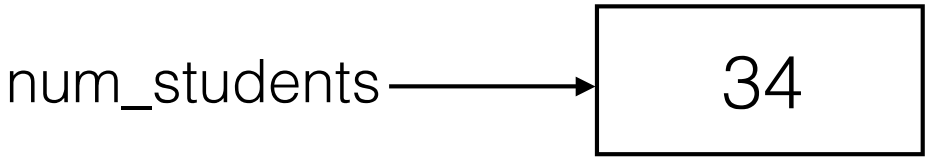


Assignment statements

- The ‘=‘ operator in Python is <u>not</u> an equal sign.
  - The following statement cannot be true in algebra

```
num_students = num_students + 2
```

  - It means, the new value of `num_students` is the previous value of `num_students` plus `2`.



Assignment statements

- What is the value of `c` after executing this code?

  (A) 3

  (B) 5

  (C) 15

  (D) 23

```
a = 15
b = 3
c = 5
c = a + 5
c = c + 3
```

Assignment statements

- What is the value of `x` after executing this code?

```
x = x + 3
```

(A) 3 because x is not initialized.

(B) unknown, because x is not initialized

(C) The program crashes because x is not defined.

Assignment statements

- What is the value of `a`, `b` and `c` (respectively) after executing this code?

  (A) 5, 3, 7

  (B) 3, 5, 7

  (C) 5, 5, 5

  (D) 5, 7, 5

  (E) 3, 5, 3

```
a = 5
b = 3
c = 7
c = b
b = a
a = c
```

Operators and Operands

- Operators are special symbols that represent computations like addition and multiplication.
  - The values the operator is applied to are called operands.
- Arithmetic operators
  - **+** for addition
  - **-** for subtraction
  - ***** for multiplication
  - **/** for devision
  - ****** for exponentiation

```
20+32
hour-1
hour*60+minute
minute/60
5**2
(5+9)*(15-7)
```

Arithmetic Operations

- The division operator in Python 2.x would divide two integers and truncate the result to an integer:
  - Try these statements in shell (Python 2.x):

    ```
    minute = 59
    minute/60
    ```

  - the result would be 0
- In Python 3.x, the result of this division is a floating point result:
  - `0.983333333333333`
- To obtain the same answer in Python 3.0 use floored (// integer) division.
  - `minute//60`

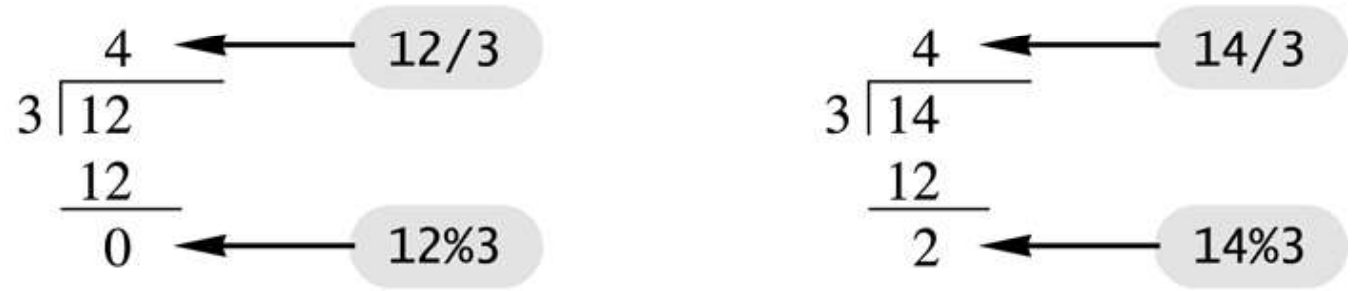
Integer reminder

- % operator (called *modulus operator*) gives the reminder from integer division
  - try this in shell:

```
print 17%5
```

  - the result would be 2

Integer Division



Integer reminder

- % operator (called *modulus operator*) turns out to be very useful
- How can I get the rightmost digit of an integer?
- How can I check whether one number is divisible by another?

Assignment statements

- What is the value of `t` after executing this code?

  (A) 3

  (B) 5

  (C) 15

  (D) 1

```
a = 15
b = 3
c = a
a = b
b = c
t = a%10
```

Expressions

- Expression: a combination of values, variables, and operators.
- A value all by itself is considered an expression, and so is a variable (assuming that the variable x has been assigned a value):
  - Example (assuming that the variable x has been assigned a value)
  - 17
  - x
  - x+17
- If you type an expression in interactive mode, the interpreter evaluates it and displays the result
- But in a script, an expression all by itself doesn't do anything! **This is a common source of confusion for beginners**

Asking the user for input

- Sometimes we would like to take the value for a variable from the user via their keyboard.
- Python provides a built-in function:
  - in Python 3.x it is called `input()`
  - in Python 2.x it is called `raw_input()`
  - try this in shell (Python 2.x)
  - `x = raw_input()`

Asking the user for input

- Before getting input from the user, it is a good idea to print a prompt telling the user what to input.
- You can pass a string to input to be displayed to the user before pausing for input:
  - try this in shell (Python 2.x)
  - `name = raw_input("what is your name?\n")`
- \n at the end of the prompt represents a newline, which is a special character that causes a line break. That's why the user's input appears below the prompt.

Asking the user for input

- `raw_input()` always return the input as string.
- If you expect the user to type an integer, you can try to convert the return value to int using the int() function:
  - try this in shell (Python 2.x)

```
>>> prompt = 'What...is the airspeed velocity of an unladen swallow?\n'
>>> speed = input(prompt)
```

  - then type:

```
>>> int(speed)
```

Suggested Reading

- Chapter 1 and 2