

Intro to Programming

or,

Getting Homer to go to the Store

Lisa: "I *gave* you a list!"

Homer: "Oh, you were *way* off."

What's a Program?

- A program is a list of instructions that the computer should follow:

Go to the store;

Get bread;

Get milk;

Pay at checkout;

Return home;

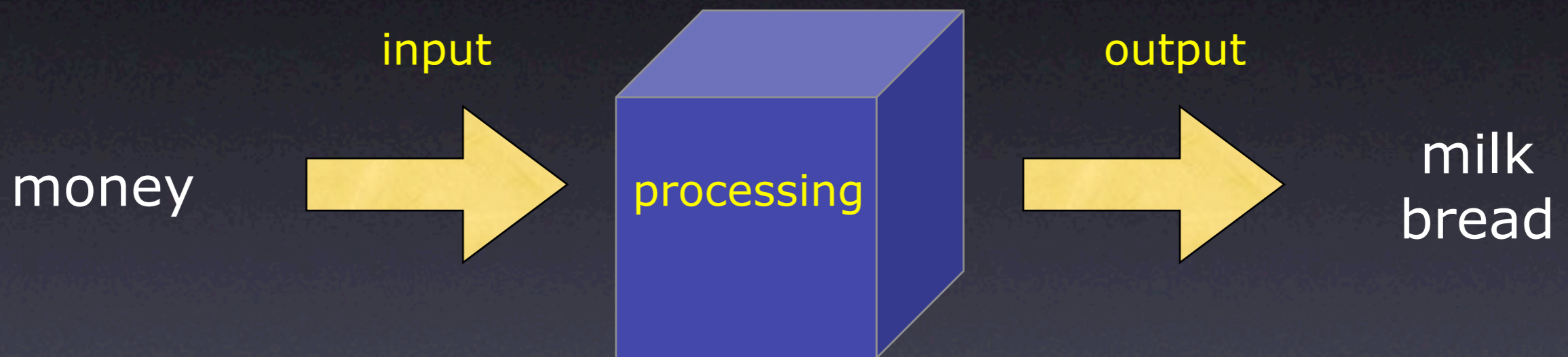
- Each instruction is unambiguous and acts like a complete sentence in English (or, more specifically, an independent clause) and so ends with a semi-colon.

The Medium

- The program itself is just a plain text document, not unlike a webpage. You can write programs in NotePad or any other barebones text editor.
- A program differs from a webpage, though, in a few significant ways:
 - Whereas a webpage is opened in a browser like Internet Explorer, a program has to be opened by an application that understands the language you're using. (More on this in the lab.)
 - The contents of the text document have to obey the grammatical rules (the "syntax") of the programming language, and have to be meaningful statements (i.e., have "semantic content"; they can't be gibberish).

Input / Processing / Output

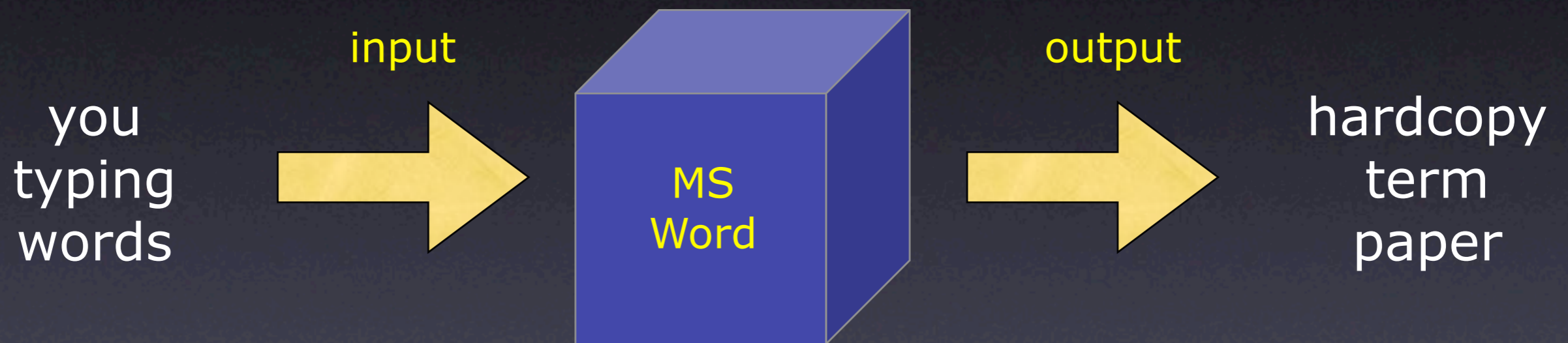
A computer program is pretty simple: it accepts some **input**, **processes** it according to those instructions, and then yields some **output**.



go to store; manipulate
items, baskets, bags,
money.

Input / Processing / Output

Even complex programs obey the same dynamic. Word processors and programs that drive the space shuttle all take input, process it, and produce output.



manipulate lines; maintain margins; check spelling; hyphenate words; place footnotes; add bibliography.

An Algorithm

- Technically, our list of instructions isn't a "program" but rather an **algorithm**:

Go to the store;

Get bread;

Get milk;

Pay at checkout;

Return home;

- An algorithm is "a finite sequence of unambiguous, executable steps that ultimately terminate if followed." Even Homer Simpson could follow the instructions. Algorithms often are written in a kind of pidgen English so humans can read them.

Algorithms & Languages

- So whereas an algorithm is in pseudo-English, a program is that same algorithm but encoded in a particular computer programming language.
- There are hundreds, probably thousands, of programming languages. Like human languages, each has its salient properties, strengths, weaknesses and even, yes, literary styles. Every language has its devotees.
- We're using a language called **Perl** (note: not all caps!) that was invented by a linguist named Larry Wall in the early 1990s.

What's a Variable?

- Variables are containers into which the computer can put its goodies:

Go to the store;

Put human into CAR

Get bread;

Put bread into BASKET

Get milk;

Put milk into BASKET

Pay at checkout;

Put bread and milk into BAG

Return home;

Put human and BAG into CAR

What's a Variable?

- Variables are containers into which the computer can put its goodies:

Go to the store;

Put human into CAR

Get bread;

Put bread into BASKET

Get milk;

Put milk into BASKET

Pay at checkout;

Put bread and milk into BAG

Return home;

Put human and BAG into CAR

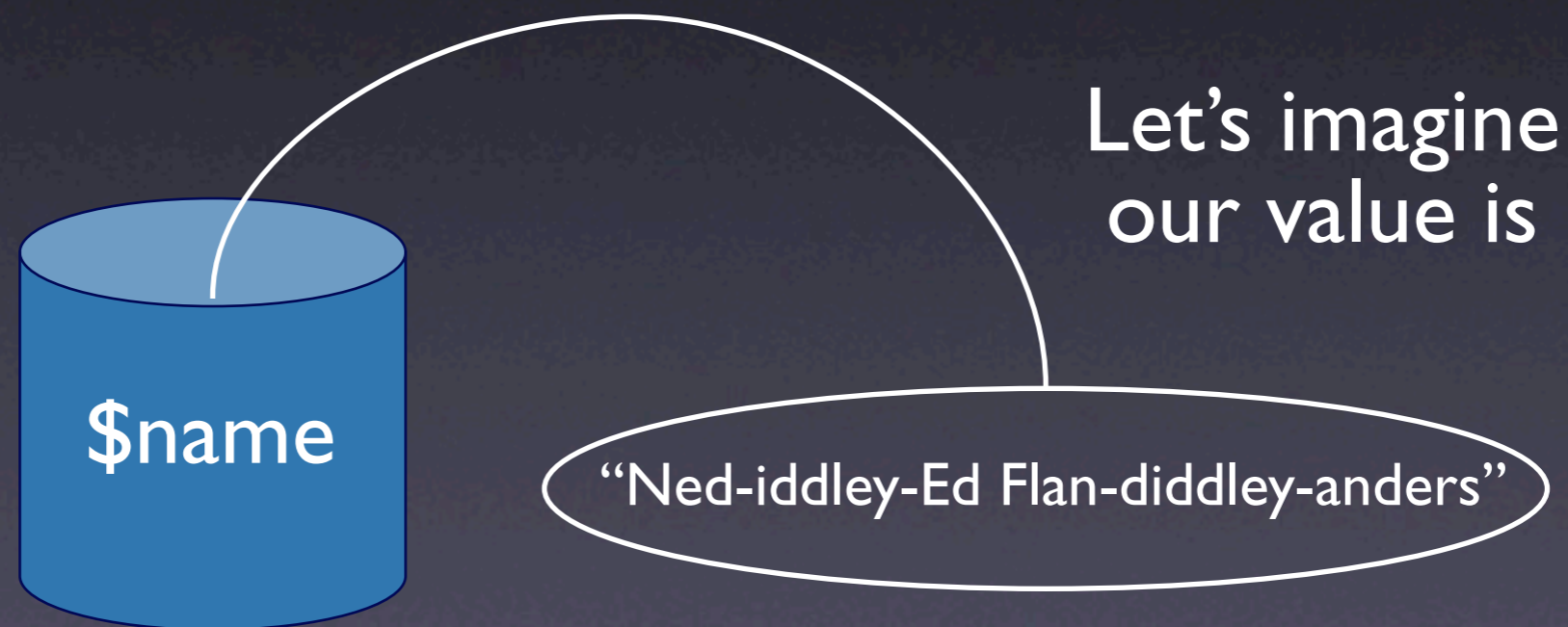
- So CAR, BASKET and BAG are variables that can store data. Sometimes variables (like CAR) can contain other variables (like BAG).

Scalar

- “Scalar” means simply “singular” — as opposed to “plural.”
-
- So an object that can be treated as a single, non-plural entity will be considered a scalar, no matter if it’s composed of numbers or letters.
- So all of the following are scalars:
 - 4
 - 3.14159
 - “hello, my name-iddley name is Nediddly”
 - 4.34
 - 0x4FC3
 - “Howdy\n\n\t\tWorld!\n\n\n”

Scalar Variables

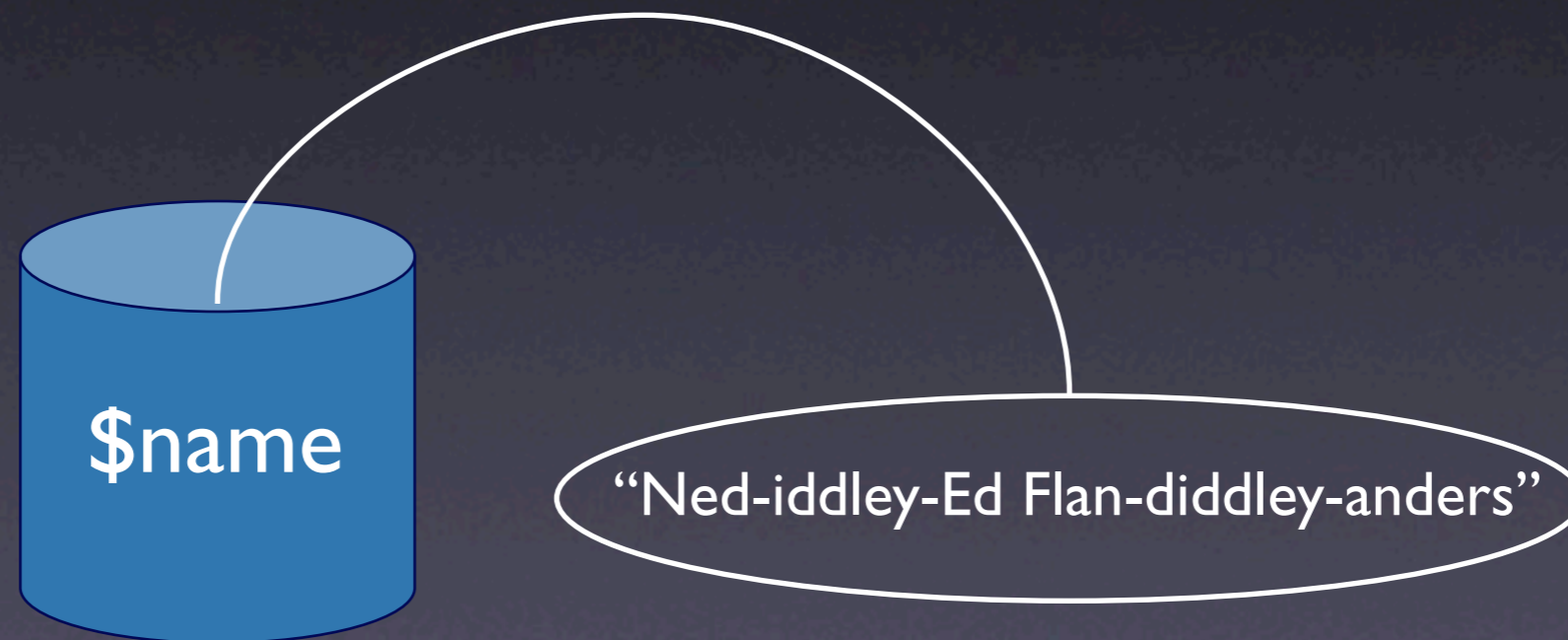
Think of a **variable** as a bucket into which we can put a value. Unlike you at the grocery store, computer programs can't carry anything; they need a bucket for everything. Buckets need to be named. Buckets that contain scalar values have a **\$** prefix—**\$** looks like an **S** which stands for Scalar.



Scalar Variables

In Perl:

```
$name = "Ned-iddley-Ed Flan-diddley-anders";
```



Scalar Variables

- In Perl, a bucket that will hold a scalar is called a “scalar variable.”
- According to the Rules of Perl, a scalar variable is a name prefixed with a \$
- By convention (and sometimes by rule), variable names are typically lowercase.
- Choose variable names that clearly establish what you’re storing in the variable.

Scalar Variable Names

- So the following are valid scalar variable names:
 - \$total
 - \$how_many
 - \$H2SO4
 - \$minimumNumberOfElves
 - \$number_of_guns_owned_by_Chuck_Heston
 - \$x
 - \$is_the_list_sorted_yet

Scalar Variables

- Perl variable names are case sensitive, so the following variable names are NOT equivalent (each refers to a different bucket):
 - `$how_many` \neq `$HOW_MANY` \neq `$How_many`
 - `$H2SO4` \neq `$h2SO4` \neq `$h2sO4`
 - `$ID_number` \neq `$id_number` \neq `$id_Number`
 - `$name` \neq `$NAME` \neq `$nAmE` \neq `$NaMe`

Assignment in Perl

- Notice that the equals sign (=) is not simply the statement of a fact. It's a transitive verb and is therefore directional. Think of it like this:

```
$name ← “Ned-iddley-Ed Flan-diddley-anders”;
```

- It means “take whatever is on the right-hand side and put it into the variable listed on the left-hand side.”
- You'll notice that the test for equality (i.e., to decide if something is a fact) in a Perl **if** statement is two equals signs:

```
if ($count == 10) {print “Basket is full!”};
```

Strings

- A collection of letters (or, more accurately, alphanumerics) is called a “string” and a string gets treated as a scalar, no matter how long it is. Each of these is a scalar string and we put it in quotation marks to define its boundaries:
 - “hello, my name-iddley name is Nediddly”
 - “Welcome to the Desert of the Real.”
 - “Yo! My friend has 36 cats!”
 - “4 is larger than 3, but pi is 3.14159\n”
 - “\t\t\t\tHowdy,\n\n\n\n\n\nWorld!\n”

Perl Assignment

- Here are some Perl assignment statements to show you how scalars get put into buckets:
 - `$name = "Homer";`
 - `$how_many = 4;`
 - `$total = 3.97 + 6.49;`
 - `$full_name = "Marge Simpson";`
 - `$output = "Homer\n+\nMarge";`

Statements

- Just like the = sign is really a transitive verb, each line in a Perl program is like an independent clause in English. It can have nouns, verbs, direct objects and even indirect objects.
- And just as we separate independent clauses with semi-colons in English, so too in Perl:

```
$name = "Ned-iddley-Ed Flan-diddley-anders";
```

Statements

- In effect, Perl doesn't care how long a sentence is or how oddly it's spaced. Perl will keep reading until it finds the semi-colon:

`$name`

`=`

`“Ned-iddley-Ed Flan-diddley-anders”;`

Numeric Scalar

- Statements allow you to manipulate data by simply referring to the variable names. Scalar variables that hold numbers can be combined using these operations:
 - + signifies addition
 - signifies subtraction
 - * signifies multiplication
 - / signifies division
 - ** signifies exponentiation (a number raised to a power)

Statements

- Examples of statements using scalars that contain numbers:

`$busted = $ghosts + $apparitions;`

`$circumference = 2 * $pi;`

`$average = $total / $how_many;`

`$exposed_area = $length ** 2 - 2 * $pi;`

`$sum = $sum + $widgets_just_purchased;`

String Scalar Operations

- Scalar variables that hold strings can be combined using these operations:
 - signifies addition (“concatenation”)
 - x signifies repetition

String Scalar

Examples of string scalar statements:

```
$full_name = "Ned" . " " . "Flanders";
```

```
$full_name = $first_name . " " . $last_name;
```

```
$song = "I am evil Homer!\n" x 3;
```

```
$speech = ($rant . "Yeah!") x $how_many . "Done!";
```

Getting Output

- Output in Perl is sent to the screen with the **print** statement. Contrary to popular belief, **print** doesn't send output to the printer, but to the screen.
- You can print variables, literal strings, expressions, or a combination of all:

```
print "I am evil Homer!";
```

```
print $total;
```

```
print "Ned-diddley" x 3;
```

```
print 4 * 3;
```

Review of Terms

- **Algorithm**: a finite sequence of unambiguous, executable steps that ultimately terminate if followed.
- **Program**: an algorithm written in a particular programming language like Perl or Java or C++.
- **Variable**: a bucket that holds data
- **Scalar**: a singular piece of data (a datum). There are plural forms of data, but we'll learn those later. Scalar variable prefix: **\$** (looks like a funky "S" for "scalar").
- **Statement**: one "sentence" in a program; like sentences in English, they end with a semi-colon.

Top Secret Preview

- Singular data are called **scalars**. The variable prefix is **\$**, which looks like a funky “S”.
- Plural data—otherwise known as lists—are handily placed into a variable type called an **array**. An array’s variable prefix is **@**, which looks like a funky “a”. Example: a customer waiting list at a busy restaurant.
- Data that naturally come in pairs are conveniently stored in a variable type called a **hash**. Its prefix is **%**, which looks like a funky “h”. (Supposedly.) Example: an inventory.

Chardonnay: 7; Pinot Noir: 3; Merlot: 8.

Intro to Programming

or,

Getting Homer to go to the Store

Lisa: "I *gave* you a list!"

Homer: "Oh, you were *way* off."