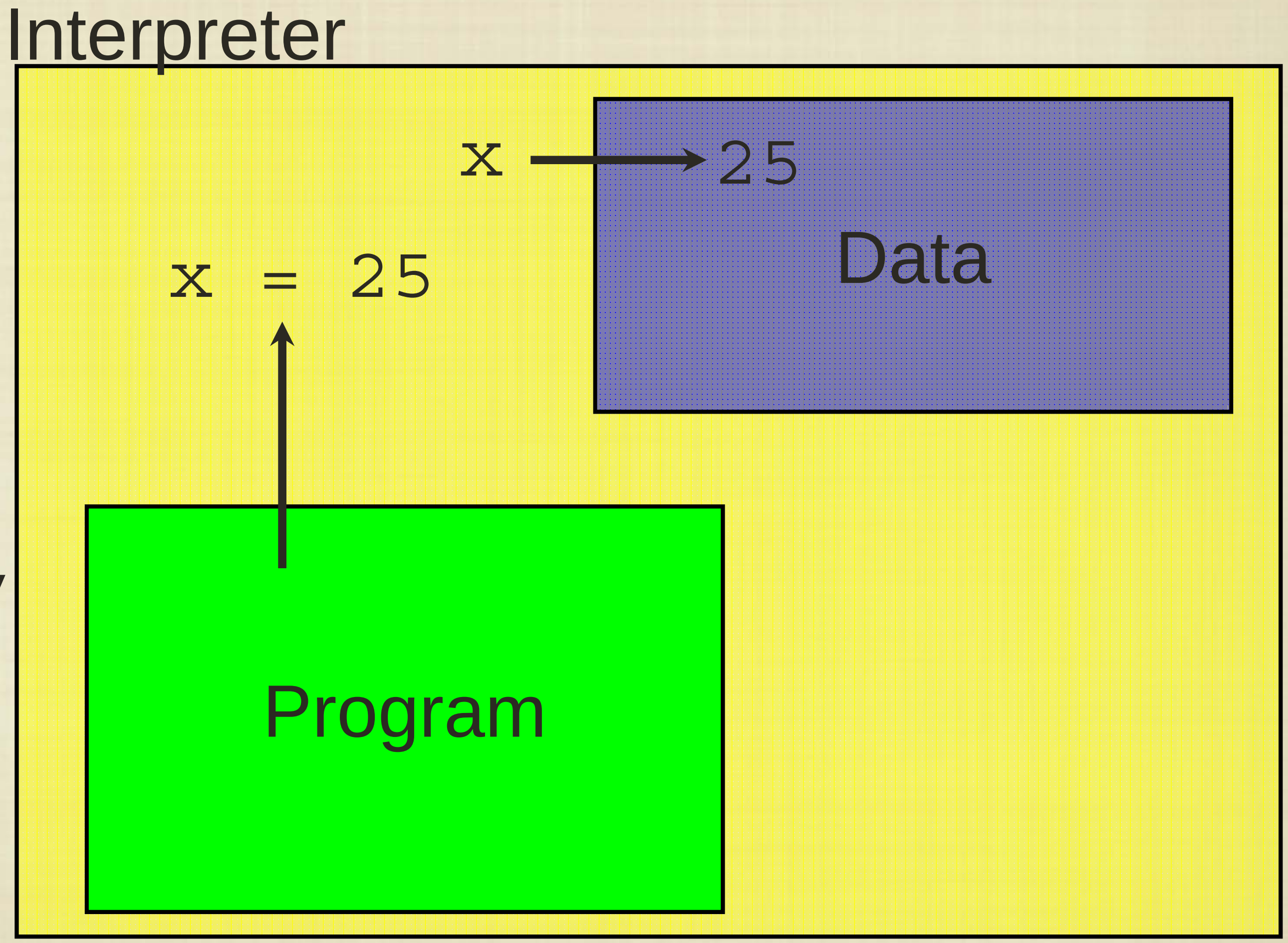
Languages and Data Types II

Spring 2014
Carola Wenk

Python Types

- Unlike most languages, we do not need to “declare" variables up front.

```
x = 25
x = [1,2,3]
x = “cmps1600”
```

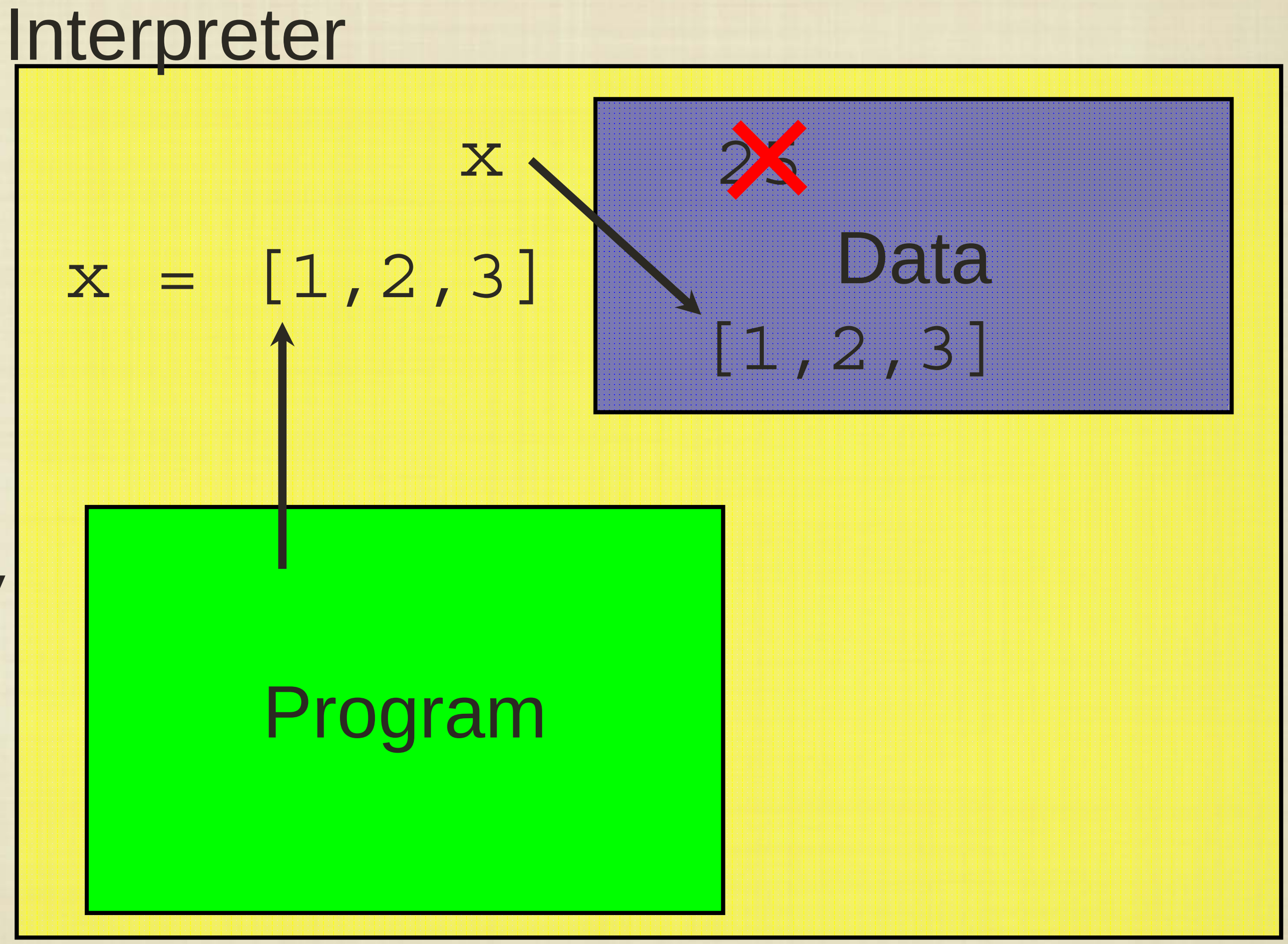


Python allows types of variables to be unspecified and allocates storage as necessary.

Python Types

Unlike most languages, we do not need to “declare" variables up front.

```
x = 25
x = [1,2,3]
x = “cmps1600”
```

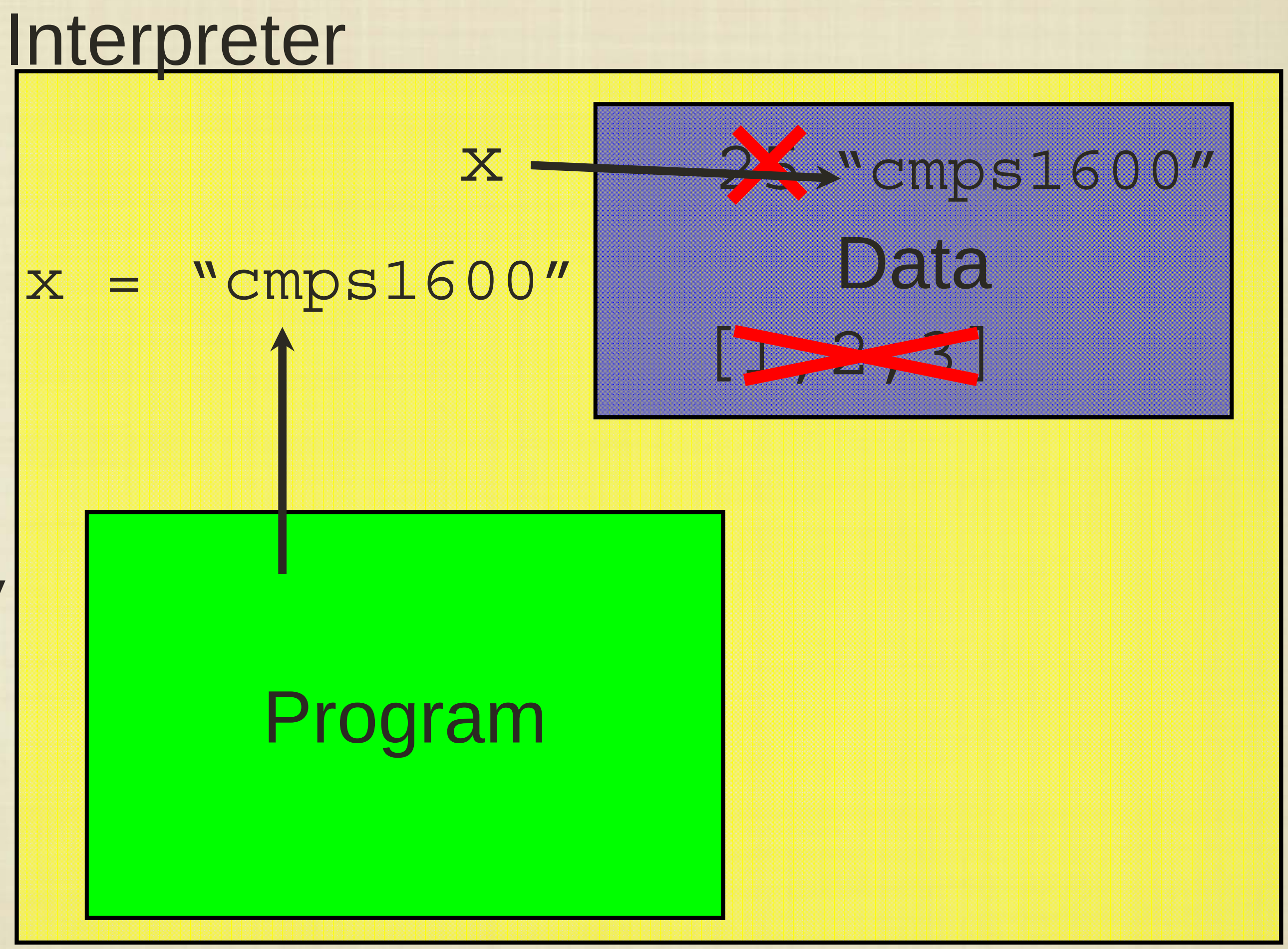


Python allows types of variables to be unspecified and allocates storage as necessary.

Python Types

Unlike most languages, we do not need to “declare" variables up front.

```
x = 25
x = [1,2,3]
x = “cmps1600”
```



Python allows types of variables to be unspecified and allocates storage as necessary.

Python Types

```
def f(arg1, arg2):
  if (arg1 + arg2 < 10):
     result = “small”
  elif (arg1 + arg2 < 15):
     result = “medium”
  else:
     result = 1000
  return result;


print f(10, 1)
print f(15, 25)
print f(f(9,6), f(1, 1))
```

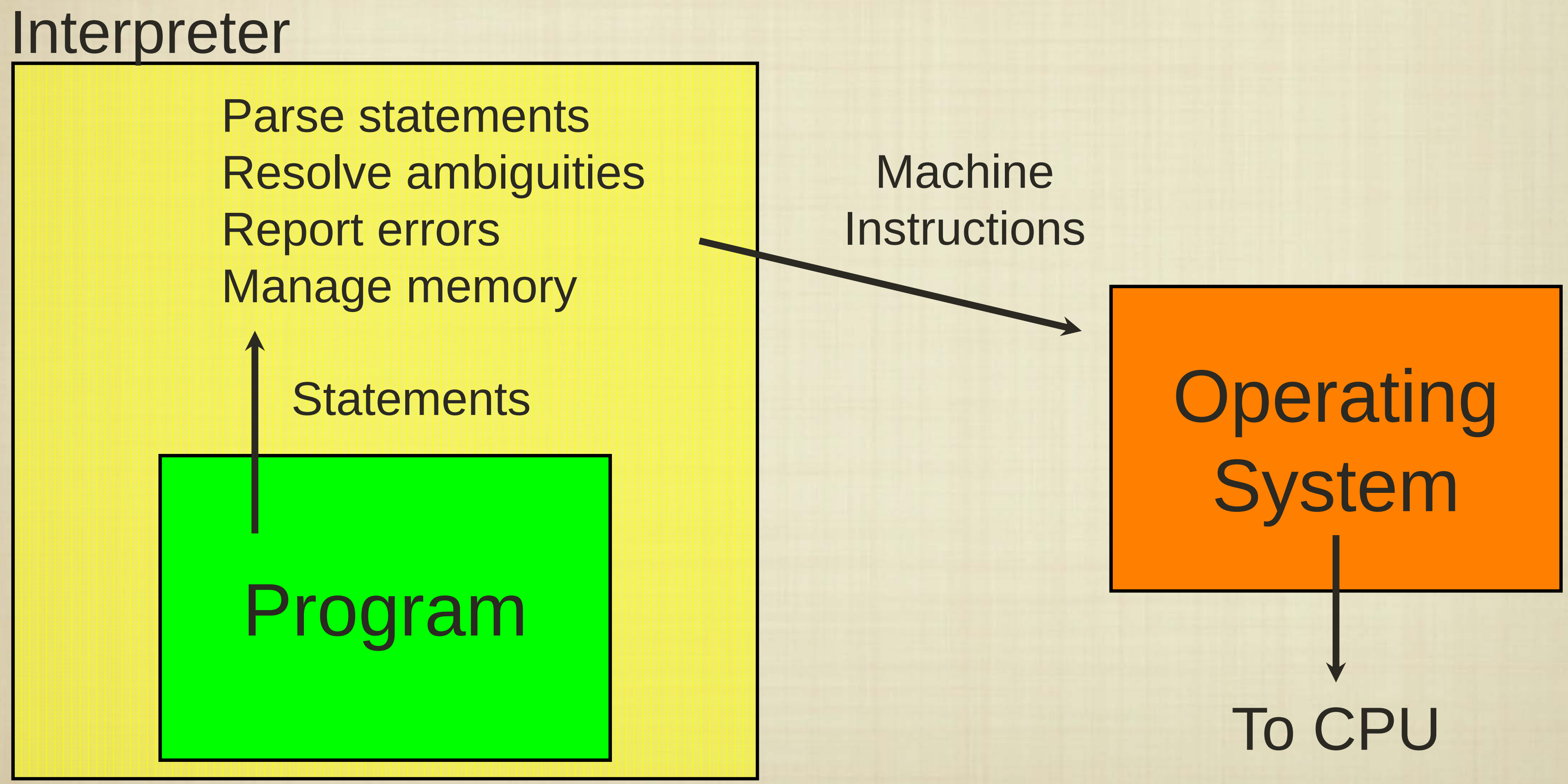
What does this program do?

Causes a “runtime” error

The interpreter decides whether operations “make sense” and types are sometimes “inferred” automatically; it is also a safety net since errors can be detected prior to machine code generation.

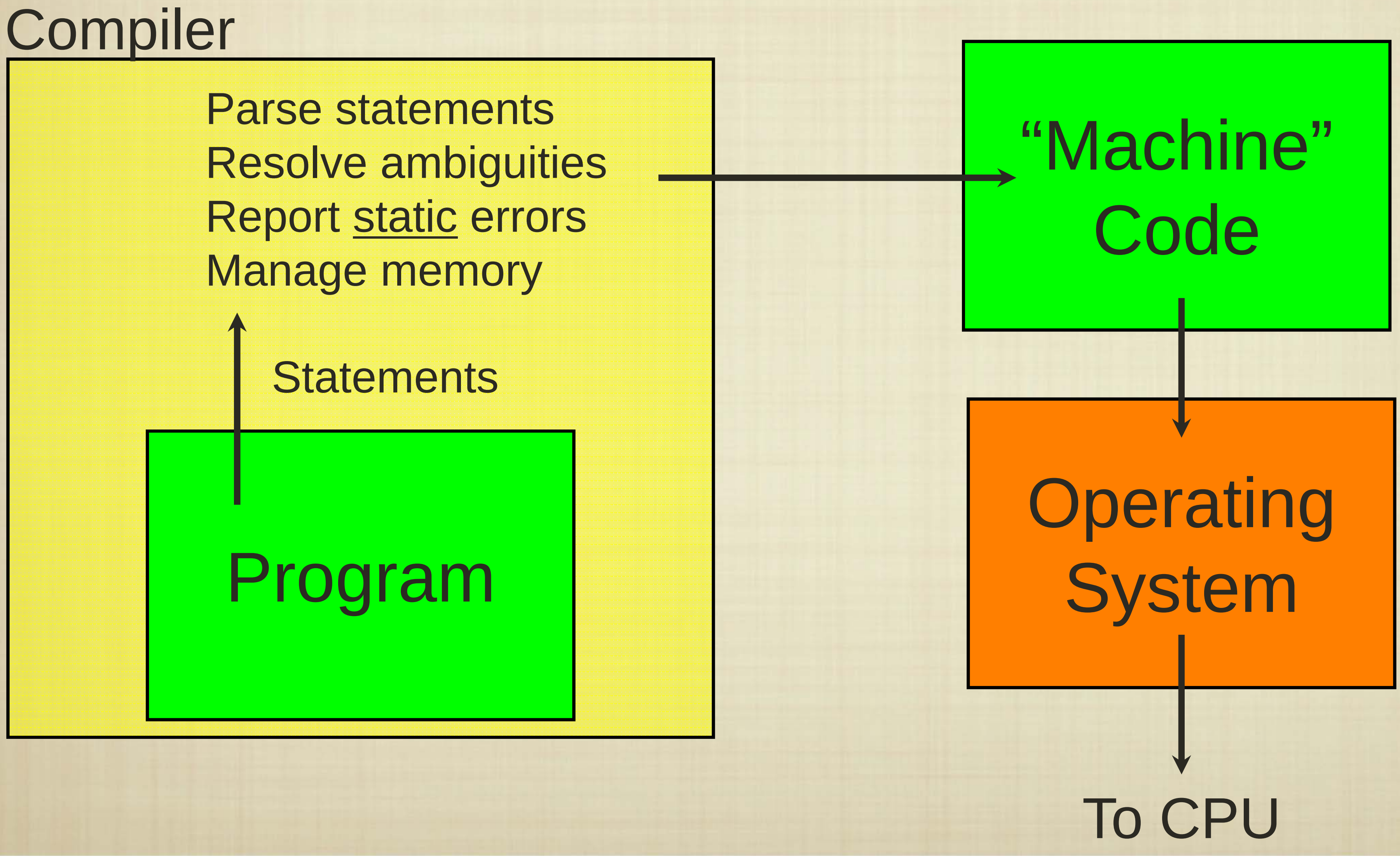
Language Platforms

Interpreted languages operate in an environment that provides some language features “under the hood”.



Language Platforms

Compiled languages operate in a self-contained environment, and generally do not have a "safety net."



Variables and Types

- Python doesn’t care about types, because the work of checking that program statements make sense is left to the interpreter.

- In other languages (Java, C/C++), these checks are not performed to improve performance.

- Thus <u>strongly typed</u> languages must have a way to <u>declare</u> the types/size of data variables can hold.

<u>Python</u>

```
x = 0;
x = 1.0;
x = [1, 2, 3];
x = “abcdefgh”;
x = [‘a’,‘b’,‘c’,’d’]
```

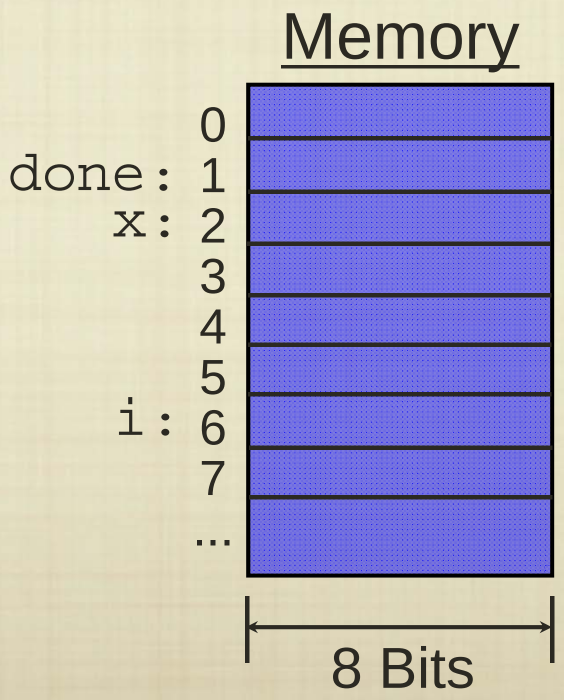
<u>Java</u>

```
int i = 0;
double f = 1.0;
int L[] = {1, 2, 3};
String s = “abcdefgh”;
char M[] = {‘a’,’b’,’c’,’d’}
```

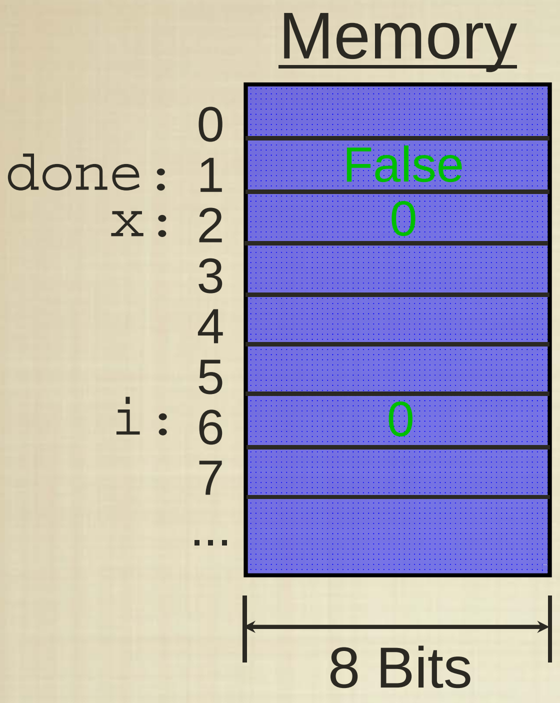
Variables and Types

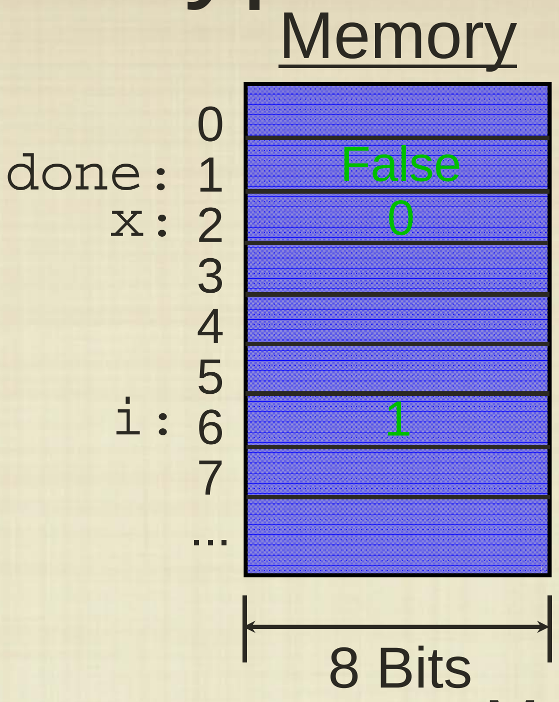
- Although Python doesn't care about types, they exist: numbers, strings, and lists.
- Java has the same types: `int/short/long, float/double, boolean, char.`
- A variable name is simply a placeholder for a memory address.

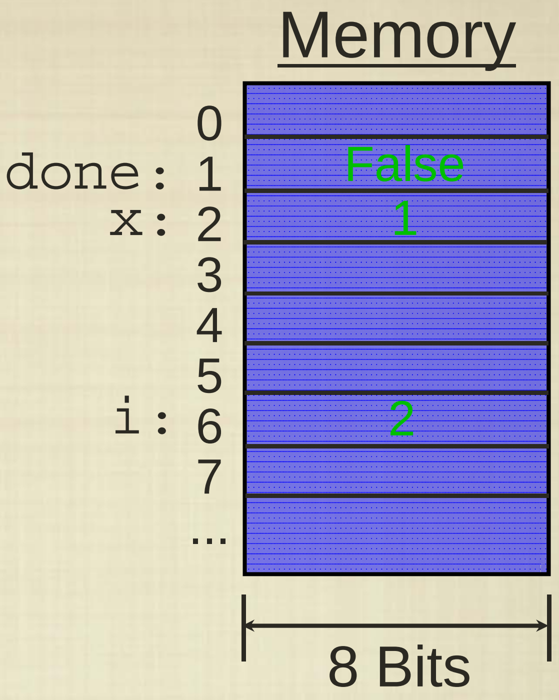
```
def f():
    x = 0
    i = 0
    done = False
    #snapshot
    while (!done):
        x = x + i
        done = (i > 2)
        i = i + 1
        # snapshot
    return x
```



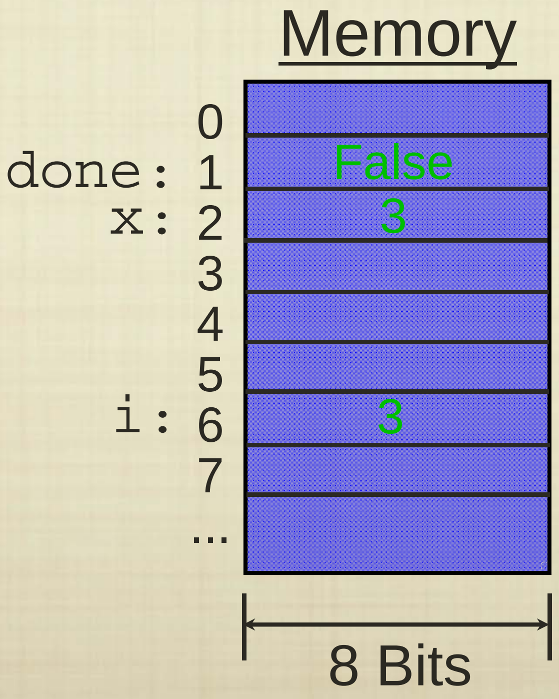
Variables and Types

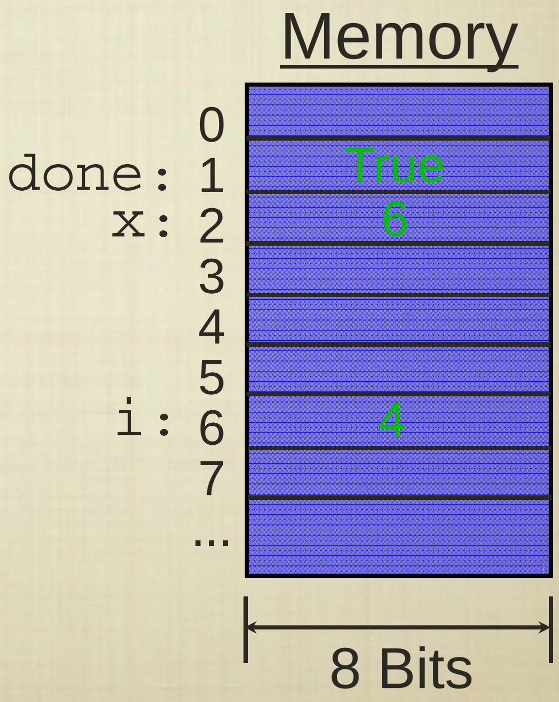






```
def f():
    x = 0
    i = 0
    done = False
    #snapshot
    while (!done):
        x = x + i
        done = (i > 2)
        i = i + 1
        # snapshot
    return x
```





Conditionals

Python

```
if <condition>:
    <block of statements>
elif <condition>:
    <block of statements>
else:
    <block of statements>
```

Java/C/C++

```
if (<condition>) {
<block of statements>
}
else if (<condition>){
<block of statements>
}
else {
<block of statements>
}
```

For conditional statements, the only real difference in syntax between Python and Java/C/C++ has to do with scope declaration.

Java/C/C++ use braces to delimit blocks of statements, instead of indentation.

Also, in Java/C/C++ the condition has to be enclosed in parentheses.

Looping

Python

```
for i in <list>:
    <block of statements>


while (<condition>):
    <block of statements>
```

Java/C/C++

```
for (<init>; <condition>; <increment>) {
<block of statements>
}

while (<condition>) {
<block of statments>
}

do {
<block of statements>
} while (<condition>);
```

Again, looping constructs are fairly similar, except for how scope is defined.

Java/C/C++ also have a “do-while” construct that can be convenient at times.

~~Functions~~ Methods

```
public int increment(int i) {
    return i+1;
}

public void printHello() {
    System.out.println(“Hello”);
}

}
```

We must also declare the types of not only variables, but also of functions (called methods in Java).

Everything is a Class…

```
public class Hello {

public static void main(String[] args){
  System.out.println(“Hello World”);
}
}
```

A key difference between Python and Java is that, while Python allows “optional” class declarations, in Java everything is a class.

That is, we cannot just execute a series of statements as in Python. Instead, all program execution occurs through the invocation of a class “instance”.

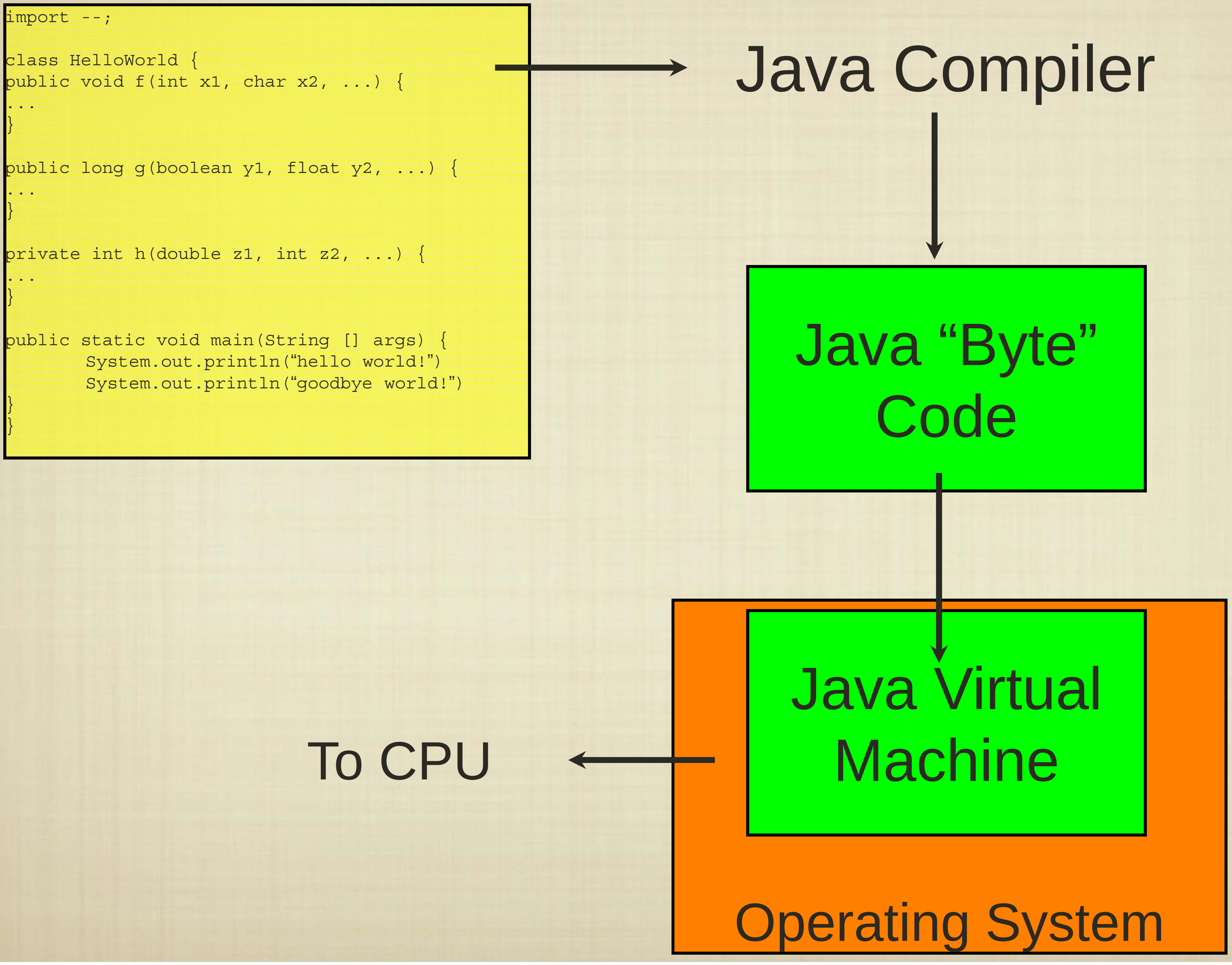
Program Structure

```
import A, B, C

def f(x1, x2, ...):
...

def g(y1, y2, ...):
   ...


print "hello world!"

def h(z1, z2, ...):
...

print "goodbye world!"
```

```
import A, B, C;

class HelloWorld {
public void f(int x1, char x2, ...) {
...
}

public long g(boolean y1, float y2, ...) {
...
}

private int h(double z1, int z2, ...) {
...
}

public static void main(String[] args) {
     System.out.println("hello world!")
     System.out.println("goodbye world!")
}
}
```

In Java, "everything is a class" so programs are initiated in the `main` method of a class, and class files are "executed."

Java Runtime System



```
import --;

class HelloWorld {
public void f(int x1, char x2, ...) {
...
}

public long g(boolean y1, float y2, ...) {
...
}

private int h(double z1, int z2, ...) {
...
}

public static void main(String [] args) {
       System.out.println("hello world!")
       System.out.println("goodbye world!")
}
}
```