# Digging into memory

To understand what pointers are, you'll need to dig into the memory of the computer.

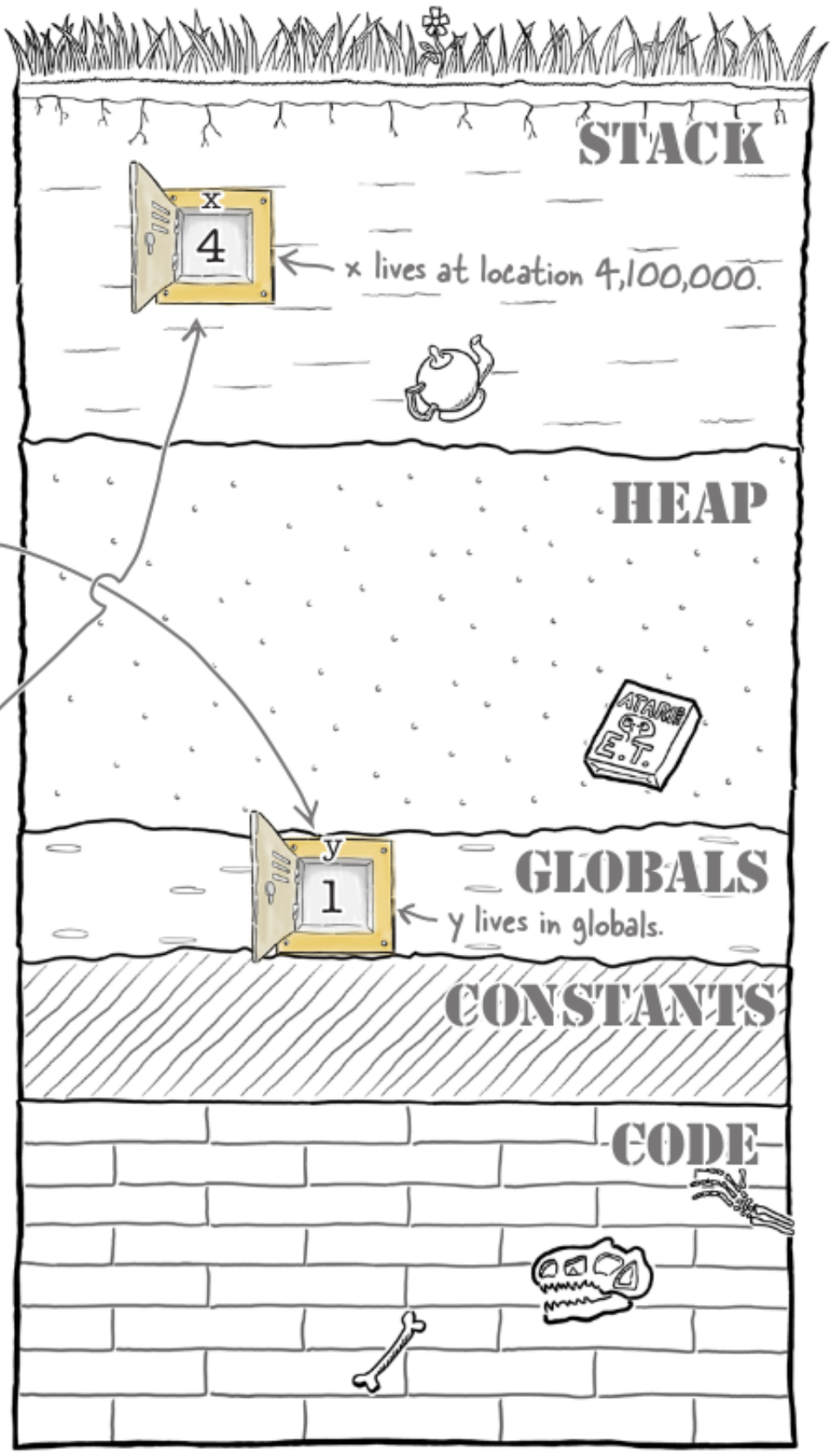
Every time you declare a variable, the computer creates space for it somewhere in memory. If you declare a variable *inside* a function like `main()`, the computer will store it in a section of memory called the **stack**. If a variable is declared *outside any function*, it will be stored in the **globals** section of memory.

```
int y = 1;

int main()
{
   int x = 4;
   return 0;
}
```

Variable y will live in the globals section. Memory address 1,000,000. Value 1.

Variable x will live in the stack. Memory address 4,100,000. Value 4.




The computer might allocate, say, memory location 4,100,000 in the stack for the x variable. If you assign the number 4 to the variable, the computer will store 4 at location 4,100,000.

If you want to find out the memory address of the variable, you can use the **&** operator:

```
printf("x is stored at %p\n", &x);
```

&x is the address of x.

%p is used to format addresses.

This is what the code will print.

```
x is stored at 0x3E8FA0
```

This is 4,100,000 in hex (base 16) format.

You'll probably get a different address on your machine.

The address of the variable tells you where to find the variable in memory. That's why an address is also called a ***pointer***, because it ***points*** to the variable in memory.

**A variable declared inside a function is usually stored in the stack.**
**A variable declared outside a function is stored in globals.**