Variables and Memory

What’s a variable, really?
What’s a type?

Not like variables in math

In math, \( x = x + 1 \) is just wrong

In Python, \( x = x + 1; \) has a meaning ...

Take the value of \( x \),
add 1 to it,
store the result in \( x \).

What’s a variable?

First try:
A variable is a name for a location in memory
(a memory “cell”).

(not quite right ... but it’s a start)

When I write “\( x = 32 \)” I mean “put the value
32 in the memory location named ‘\( x \)’.

What’s memory, really?

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0101010110</td>
</tr>
<tr>
<td>1</td>
<td>0101010110</td>
</tr>
<tr>
<td>2</td>
<td>0101010000</td>
</tr>
<tr>
<td>3</td>
<td>1101101010</td>
</tr>
<tr>
<td>4</td>
<td>0101101110</td>
</tr>
<tr>
<td>5</td>
<td>0101010110</td>
</tr>
<tr>
<td>6</td>
<td>0111101011</td>
</tr>
<tr>
<td>7</td>
<td>0101111011</td>
</tr>
</tbody>
</table>

... 

Memory cells are one big list,
numbered from zero.
The computer access them by
“address” (number).

Variable ‘\( x \)’ might mean cell #6.
\( x = 33 \) might mean:
put 100001 in cell #6
Memory

01100010_2

Means 62_{16} (interpreted as an integer)
Or 98_{10}
Or ‘b’ (interpreted as an ASCII character)
Or BOUND (interpreted as an x86 instruction)

Or ... it doesn’t “mean” anything, but we can interpret it several ways, as data or as a program instruction.

Memory Representation

It’s all binary (1’s and 0’s)

What does 01100010_2 mean?

(Trick question ... why?)

Where’s the type?

Dynamic Types
(Python, …)

x = 5

Static Types
(Java, …)

int x;

x = 5;

x

int

00000101

00000101
Dynamic types allow ...

\[
x = 5 
\]
\[
x = 5.0 
\]
\[
x = "Text" 
\]

Some basic Python types

<table>
<thead>
<tr>
<th>name</th>
<th>meaning</th>
<th>example (literal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>whole number, represented as 32 bits</td>
<td>42</td>
</tr>
<tr>
<td>float</td>
<td>floating point (approximation of real number), typically 64 bits</td>
<td>42.0</td>
</tr>
<tr>
<td>string</td>
<td>sequence of characters (text)</td>
<td>&quot;Hello World&quot;</td>
</tr>
<tr>
<td>boolean</td>
<td>true or false (represented as integers 1 and 0)</td>
<td>True</td>
</tr>
</tbody>
</table>

(There are more ... these will be enough for a while)
Operations depend on types!

15 + 32 is 47

“15” + “32” is “1532”

What is it?

x = 15
y = 2
z = x / y;

What value is in z?

What is it?

x = 15.0;
y = 2.0;
z = x / y;

What value is in z?

What is it?

x = 15
y = 2
z = x % y

What value is in z?

% is the “remainder” or “modulo” operation when applied to a pair of integers
Coercion

“Coercions” are implicit conversions from one type to another

x = 42
print x     ## What does it print?
x = x + 0.0
print x     ## Now what does it print?

Use cautiously ... can be confusing

Casts (explicit conversions)

Like coercions, but explicit ...

x = 42.84
y = int(x)   ## Convert the value of x to integer
              ## and then store in y
print y

What does it print?

Types in Assignment 1

The skeleton code for gears.py contains this line:

    ratio = float(ring) / float(cog)

Why use float()?
What happens if we omit float()?

Types in Assignment 1 (cont)

The skeleton code for change.py contains:

    kbd = raw_input("Enter ... cents: ")
    cash = int(kbd)

Why int(kbd)? Why not

    cash = raw_input("Enter ... cents:"
Variable names matter

The computer doesn’t care ... but other programmers (and graders!) do

“dollars” and “cents” are better than “x” and “y”

but this is not really a very good name longer ≠ better

Summary: What’s a variable?

It names a location in memory

Either the variable has a type (e.g., in Java), or the value in the variable has a type (in Python); the type determines how it is interpreted and the meaning of operations (e.g., +) on it.

Variable names should be descriptive (enough) and follow naming conventions

Now Python naming conventions, later Java naming conventions

Python Naming Conventions

Note: Java naming conventions differ!

### Constants (set once, never change)
MILE_PER_KM = 0.621371192

### Local variables
life_universe_etc = 42
author = “Douglas Adams”

### function
def double( n ) :
    “““Always document your functions”””
    return n + n

Office Hours Reminder

<table>
<thead>
<tr>
<th>Wednesday (today)</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:30-12:30</td>
<td>10-11</td>
<td>11:30-12:30</td>
</tr>
<tr>
<td>2-3</td>
<td>11-12</td>
<td>2-3</td>
</tr>
<tr>
<td></td>
<td>1-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2-3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>