CS101: Intro to Computing
Fall 2015

Lecture 3

Administrivia

- i>clicker attendance starts ***today***
  - Make sure you've registered on Compass!
- Homework 1 is due ***tonight***
  - I removed some questions
- Homework 2 is assigned ***now***
  - Due on Wednesday
- No lab ***next*** week

REVIEW

```
x=10
y=x+1
y=x*y
```

What is the value of y?

a) 11
b) 100
c) 110
d) None of the above.

```
x=10
y=x+1
y=x*y
```

What do we call x?

a) a literal
b) a variable
c) an expression
d) a statement

```
x=10
y=x+1
y=x*y
```

What do we call 10?

a) a literal
b) a variable
c) an expression
d) a statement

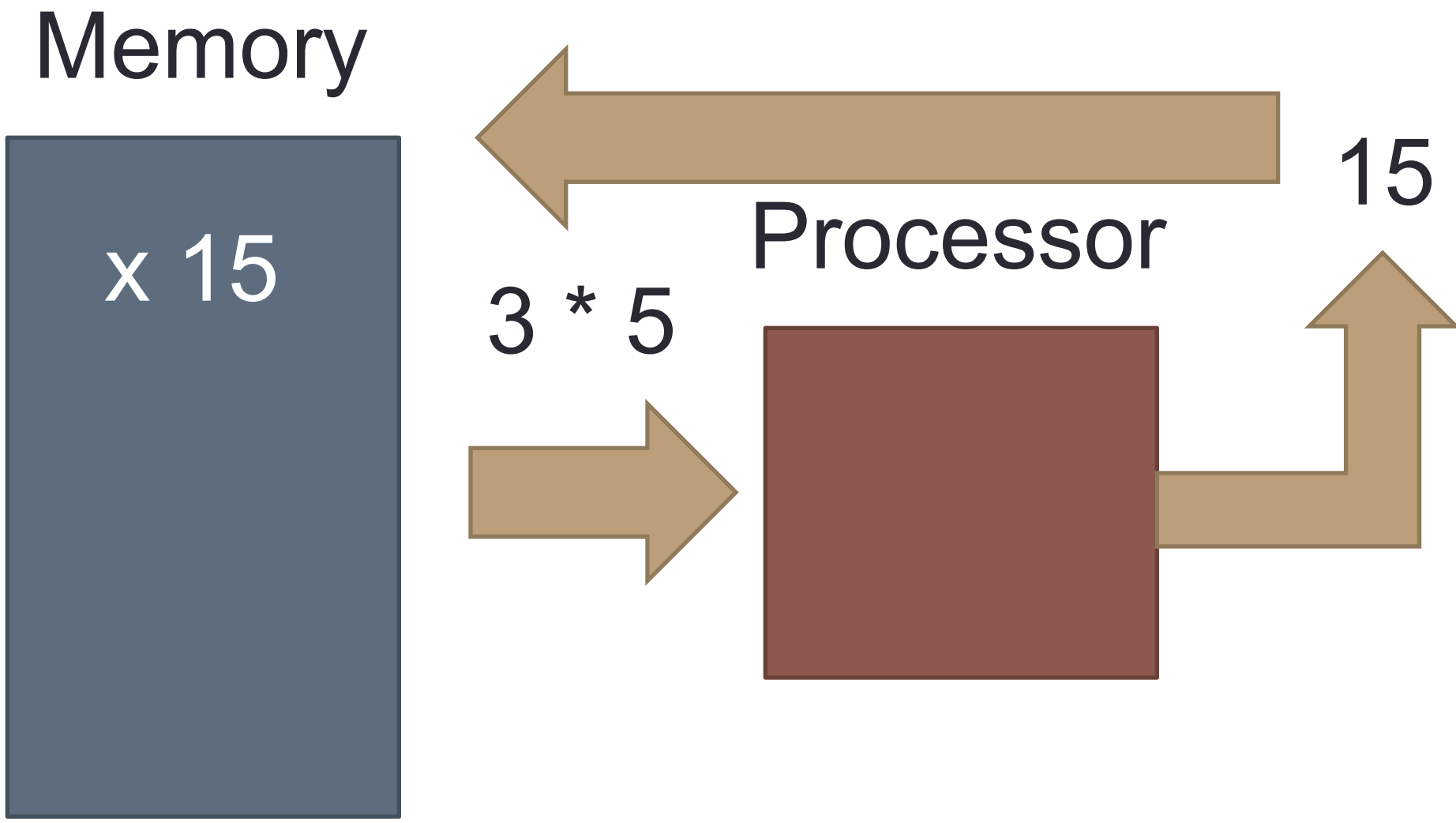
```
x=10
y=x+1
y=x*y
```

What do we call `y=x*y`?

a) a literal
b) a variable
c) an expression
d) a statement

DATA TYPES

x = 3 * 5



Encoding

```
01001000010001010100110001001100010001111
```

- What does this binary data represent?
- How does the processor know?
- Unless we know the *encoding* we cannot interpret the data.

Types

- ***Types*** define the encoding in Python.
- All values in Python have a ***type***.
- Defines how data is represented in memory.
- Defines the operations that are allowed and how they work.

NUMERIC TYPES

Encoding Numbers

- Numeric types are represented in binary
  - Fixed-length (only a certain number of bytes)

1: 0001 2: 0010 3: 0011 4: 0100

5: 0101 6: 0110 7: 0111 8: 1000

… 15: 1111

- If we add more, this causes an ***overflow***
  - We've run out of bits!
- Negative numbers? Just add a ***sign bit.***

Integers

- So far, this has been our only type.
- Represent integers
  - Positive and negative whole numbers
- Literals are just integers (e.g. -128)
- Cannot represent all integers
  - Python scales integer sizes
  - Only integers that fit in memory
  - Bigger integers = SLOWER program

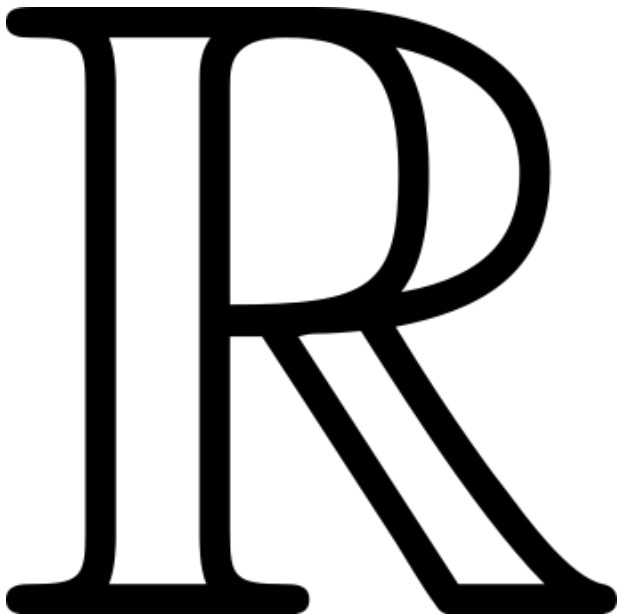
Integer operations

- Evaluating an expression of integers will always result in an integer answer.

```
3/4 =
a) 0
b) 1
c)3
d)0.75
```

Floats

- Referenced in the homework.
- Represent real numbers
  - Anything with a decimal point
- Literals have a decimal point (e.g. 3.0)
- Cannot represent all reals
  - Some are too large/small
  - Can not represent ***arbitrary precision*** (e.g. π)



Floating point operations

- Evaluating an expression of floats will result in a floating point answer.
- We will need to be careful about precision of operations.

Floats with Integers

- We can use floats and integers in the same expressions.
- The resulting value is a ***floating point***.
- Operations always default to most general numeric type.

Complexes

- Represent numbers on complex plane
  - Numbers with an imaginary component
- Imaginary component referred to with j
  - e.g. 2+1.3j
- They're "jmaginary" numbers!

```
x=4
y=3+1j
z=33.3333
print x+y+z
```

What is printed?

a) 40

b) 40.3333

c) 40.3333+1j

d) None of the above

Attribute operator

- “Reaches in” to a value to access part of its data (called an ***attribute***)
- Extracts special variables stored “inside” the type.

```
print x.real
print x.imag
```

- Both of these components are ***floats***.

```
x=(3.5+1j)
y=1
z=x+y
```

What is the value of z.imag?

a) 4.5+1j

b) 4.5

c) 1.0j

d) 1.0

STRING TYPE

Encoding Text

- Each ***symbol*** is stored individually.
  - Each symbol is one byte long
  - Represented by ASCII code

```
01001000 01000101 01001100
01001100 01001111
72       69       76
76       79
```

ASCII Table

0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y

Strings

- Literals: text surrounded by quotes
  - e.g. “TACO”
- Each symbol is called a ***character***
- Unlike numeric types, strings can vary in length!

String operations

- ***Concatenation***: combine two strings
  - Uses the + symbol
  - Example: `“CS”+”101”`
- ***Repetition***: repeat a string
  - Uses the * symbol
  - Example: `“HELLO! ”*10`
- ***Formatting***: used to encode other data as a string
  - Uses % symbol

Formatting operator

- Creates a string with a value stuck inside
  - Formatting them nicely
  - Have to indicate the ***type*** of the value INSIDE the string with a special code

```
x=100 * 54
s="String is: %i" % x
print s
```

Example

```
name=“Ryan”
grade=2/3
m1=“Hello, %s!” % name
m2=“Your grade is: %f” % grade
print m1
print m2
```