Administrivia
Homework #2–#6 will be later.
Homework #2–#6 will be later.
Final answer counts.
Homework #2–#6 will be later.
Final answer counts.
Answers will be released 18 hours later.
Lab #2 tomorrow Sunday.
Lab #2 tomorrow Sunday.

Where can you get help in this class?
- Blackboard forum
- Instructors in labs and office hours
Lab #2 tomorrow Sunday.
Where can you get help in this class?
- Blackboard forum
- Instructors in labs and office hours
You don’t need to install Python—but if you do, use Python 3.
Lab #2 tomorrow Sunday.

Where can you get help in this class?
- Blackboard forum
- Instructors in labs and office hours

You don’t need to install Python—but if you do, use Python 3.

This is not a “weeder” class—you can succeed!
Quick Review & A Bit New
x = 10
How Assignment Works

\begin{align*}
x &= 10 \\
y &= x \times x
\end{align*}
How Assignment Works

\[
\begin{align*}
x &= 10 \\
y &= x \times x \\
x \times x &= y
\end{align*}
\]
How Assignment Works

\[
\begin{align*}
    x &= 10 \\
    y &= x \times x \\
    x \times x &= y \\
    x, y &= y, x \quad \# \text{a neat trick}
\end{align*}
\]
You will have graded quiz starting from the upcoming Monday lecture!
Our execution model

\[ x = 3 \times 5 \]

memory

\[ x = 15 \]

processor

15
Question #1

\[ x = 10 \]
\[ y = x + 1 \]
\[ y = x \times y \]

What is the value of \( y \)?

A 11
B 100
C 110
D None of the above
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
Question #5

What is the value of $y$?

A 10
B 5
Data Types
What is an encoding?

01001000 01000101 01001100 01001100

What does a binary data value like this represent?

What does binary data represent?
What is an encoding?

01001000 01000101 01001100 01001100

What does a binary data value like this represent?

- What does binary data represent?
- How does the processor know?
Data Types
What is an encoding?

01001000 01000101 01001100 01001100
What does a binary data value like this represent?

- What does binary data represent?
- How does the processor know?
- The encoding interprets the value.
What is a data type?

- A **data type** defines an encoding rule.
- All values have a type.
What is a **data type**?

- A **data type** defines an encoding rule.
- All values have a type.
- The type defines how data is represented in memory.
What is a **data type**?

- A **data type** defines an encoding rule.
- All values have a type.
- The type defines how data is represented in memory.
- The type defines allowed operations and how they work.
Example

01100111 can be the number 103, the letter $g$, hexadecimal 67, 3.5, etc.

- So what are these data types?
Numeric Data Types
How do binary numbers work?

- Numeric types can be represented in binary:
  - 000 0 100 4
  - 001 1 101 5
  - 010 2 110 6
  - 011 3 111 7
How do binary numbers work?

- Numeric types can be represented in binary:
  - 000  0  100  4
  - 001  1  101  5
  - 010  2  110  6
  - 011  3  111  7

- If we add more, the number **overflows**.
How do binary numbers work?

- Numeric types can be represented in binary:
  - 000 0 100 4
  - 001 1 101 5
  - 010 2 110 6
  - 011 3 111 7

- If we add more, the number **overflows**.
- Negative numbers? Add a **sign bit**.
Integers, \( \mathbb{Z} \)

- Integers have been our only type thus far.
  
  \( \ldots, -4, -3, -2, -1, 0, +1, +2, +3, \ldots \)

- What are limits?
• Evaluating an expression of integers will generally result in an integer answer
  • 3 + 5
Evaluating an expression of integers will generally result in an integer answer

- $3 + 5$
- EXCEPTION: DIVISION!
Evaluating an expression of integers will generally result in an integer answer:

- $3 + 5$
- **EXCEPTION: DIVISION!**
- $3 / 4 \rightarrow 0.75$
Evaluating an expression of integers will generally result in an integer answer

- $3 + 5$
- $3 / 4 \rightarrow 0.75$
- $3 \ // \ 4 \rightarrow 0$ (floor division)
Floating-point numbers include a fractional part.
Floating-point numbers include a fractional part. (Anything with a decimal point—2.4, 3.0.)
Floating-point numbers include a fractional part. (Anything with a decimal point—2.4, 3.0.)

What are limits?
Floating-point numbers include a fractional part. (Anything with a decimal point—2.4, 3.0.)

What are limits?
- Overflow/underflow
- Arbitrary precision (\(\pi\), e)
Floating-point operations

- Evaluating an expression of floating-point values will result in a floating-point answer.

Engineers and scientists need to think carefully about the precision of answers.
Floating-point operations

- Evaluating an expression of floating-point values will result in a floating-point answer.
  - $3.0 + 5.5 \rightarrow 8.5$
Evaluating an expression of floating-point values will result in a floating-point answer.

- \(3.0 + 5.5 \rightarrow 8.5\)
- \(3.0 + 5.0 \rightarrow 8.0\)
Evaluating an expression of floating-point values will result in a floating-point answer.

- 3.0 + 5.5 → 8.5
- 3.0 + 5.0 → 8.0
- 3 + 5.5 → ? (what happens here?)
Floating-point operations

- Evaluating an expression of floating-point values will result in a floating-point answer.
  - $3.0 + 5.5 \rightarrow 8.5$
  - $3.0 + 5.0 \rightarrow 8.0$
  - $3 + 5.5 \rightarrow ?$ (what happens here?)

- Engineers and scientists need to think carefully about the precision of answers.
Complex numbers, \( \mathbb{C} \)

- Represent numbers with an imaginary component.

\( 1.0 + 1j \)
Complex numbers, $\mathbb{C}$

- Represent numbers with an imaginary component.
- Use $j$ for $i$:
  
  $1.0 + 1j$
Complex numbers, \( \mathbb{C} \)

- Represent numbers with an imaginary component.
- Use \( j \) for \( i \):
  
  \[ 1.0 + 1j \]

- Think of "jimaginary" numbers, I suppose.
x = 4
y = 3 + 1j
z = 33.3333
print(x + y + z)

What is printed to the screen?
Example

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

What is printed to the screen?

A 40  
B 40.3333  
C 40.3333 + 1j  
D None of the above
Reaches inside of a value to access part of its data (called an attribute).
Reaches inside of a value to access part of its data (called an attribute).

Extracts special variables stored “inside” of the type.

```python
print(x.real)
print(x.imag)
```
Reaches inside of a value to access part of its data (called an attribute).

Extracts special variables stored “inside” of the type.

print(x.real)
print(x.imag)

Both of these components are floats.
Example

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

What is the value of \( z.\text{imag} \)?
Example

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

What is the value of \( z\text{.imag} \)?

A 4.5 + 1j
B 4.5
C 1j
D 1.0
String Data Type
How does text work?

- Each symbol is stored individually, one byte long:
  - 01001000 72
  - 01000101 69
  - 01001100 76
  - 01001100 76
  - 01001111 79
## ASCII encoding table

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
<th>ASCII Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>nul</td>
<td>032</td>
</tr>
<tr>
<td>001</td>
<td>soh</td>
<td>033</td>
</tr>
<tr>
<td>002</td>
<td>stx</td>
<td>034</td>
</tr>
<tr>
<td>003</td>
<td>etx</td>
<td>035</td>
</tr>
<tr>
<td>004</td>
<td>eot</td>
<td>036</td>
</tr>
<tr>
<td>005</td>
<td>enq</td>
<td>037</td>
</tr>
<tr>
<td>006</td>
<td>ack</td>
<td>038</td>
</tr>
<tr>
<td>007</td>
<td>bel</td>
<td>039</td>
</tr>
<tr>
<td>008</td>
<td>bs</td>
<td>040</td>
</tr>
<tr>
<td>009</td>
<td>tab</td>
<td>041</td>
</tr>
<tr>
<td>010</td>
<td>lf</td>
<td>042</td>
</tr>
<tr>
<td>011</td>
<td>vt</td>
<td>043</td>
</tr>
<tr>
<td>012</td>
<td>np</td>
<td>044</td>
</tr>
<tr>
<td>013</td>
<td>cr</td>
<td>045</td>
</tr>
<tr>
<td>014</td>
<td>so</td>
<td>046</td>
</tr>
<tr>
<td>015</td>
<td>si</td>
<td>047</td>
</tr>
</tbody>
</table>

**String Data Type**

```
33/44
72 69 76 76 79
"HELLO"
'HELLO'
```
### ASCII encoding table

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Character</th>
<th>ASCII Code</th>
<th>Hexadecimal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>null</td>
<td>016</td>
<td>032 sp</td>
<td>048 °</td>
</tr>
<tr>
<td>001</td>
<td>end of file</td>
<td>017</td>
<td>033 !</td>
<td>049 !</td>
</tr>
<tr>
<td>002</td>
<td>start of text</td>
<td>018</td>
<td>034 &quot;</td>
<td>050 &quot;</td>
</tr>
<tr>
<td>003</td>
<td>end of text</td>
<td>019</td>
<td>035 #</td>
<td>051 #</td>
</tr>
<tr>
<td>004</td>
<td>end of transmission</td>
<td>020</td>
<td>036 $</td>
<td>052 $</td>
</tr>
<tr>
<td>005</td>
<td>end of record</td>
<td>021</td>
<td>037 %</td>
<td>053 %</td>
</tr>
<tr>
<td>006</td>
<td>echo</td>
<td>022</td>
<td>038 &amp;</td>
<td>054 &amp;</td>
</tr>
<tr>
<td>007</td>
<td>stop and wait</td>
<td>023</td>
<td>039 '</td>
<td>055 '</td>
</tr>
<tr>
<td>008</td>
<td>bell</td>
<td>024</td>
<td>040 (</td>
<td>056 (</td>
</tr>
<tr>
<td>009</td>
<td>horizontal tabulation</td>
<td>025</td>
<td>041 )</td>
<td>057 )</td>
</tr>
<tr>
<td>010</td>
<td>line feed</td>
<td>026</td>
<td>042 *</td>
<td>058 *</td>
</tr>
<tr>
<td>011</td>
<td>vertical tabulation</td>
<td>027</td>
<td>043 +</td>
<td>059 +</td>
</tr>
<tr>
<td>012</td>
<td>escape</td>
<td>028</td>
<td>044 ,</td>
<td>060 &lt;</td>
</tr>
<tr>
<td>013</td>
<td>return</td>
<td>029</td>
<td>045 -</td>
<td>061 =</td>
</tr>
<tr>
<td>014</td>
<td>start of heading</td>
<td>030</td>
<td>046 .</td>
<td>062 &gt;</td>
</tr>
<tr>
<td>015</td>
<td>start of text</td>
<td>031</td>
<td>047 /</td>
<td>063 ?</td>
</tr>
</tbody>
</table>

72 69 76 76 79 = HELLO
<table>
<thead>
<tr>
<th>ASCII encoding table</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 (nul) 016 ▲ (dle) 032 sp 048 0 064 @ 080 P 096 0 112 p</td>
</tr>
<tr>
<td>001 ⊙ (soh) 017 ▼ (dc1) 033 ! 049 1 065 A 081 Q 097 a 113 q</td>
</tr>
<tr>
<td>002 ® (stx) 018 ‹ (dc2) 034 &quot; 050 2 066 B 082 R 098 b 114 r</td>
</tr>
<tr>
<td>003 ★ (etx) 019 !! (dc3) 035 # 051 3 067 C 083 S 099 c 115 s</td>
</tr>
<tr>
<td>004 ‡ (eot) 020 ☐ (dc4) 036 $ 052 4 068 D 084 T 100 d 116 t</td>
</tr>
<tr>
<td>005 ⊘ (enq) 021 ¡ (nak) 037 % 053 5 069 E 085 U 101 e 117 u</td>
</tr>
<tr>
<td>006 ♠ (ack) 022 — (syn) 038 &amp; 054 6 070 F 086 V 102 f 118 v</td>
</tr>
<tr>
<td>007 ♦ (bel) 023 † (etb) 039 ' 055 7 071 G 087 W 103 g 119 w</td>
</tr>
<tr>
<td>008 ▲ (bs) 024 ↑ (can) 040 ( 056 8 072 H 088 X 104 h 120 x</td>
</tr>
<tr>
<td>009 ▼ (tab) 025 ↓ (em) 041 ) 057 9 073 I 089 Y 105 i 121 y</td>
</tr>
<tr>
<td>010 ◄ (lf) 026 ☐ (eof) 042 * 058 : 074 J 090 Z 106 j 122 z</td>
</tr>
<tr>
<td>011 ® (vt) 027 ← (esc) 043 + 059 ; 075 K 091 ] 107 k 123 {</td>
</tr>
<tr>
<td>012 ⊘ (np) 028 L (fs) 044 , 060 &lt; 076 L 092 \ 108 l 124</td>
</tr>
<tr>
<td>013 ☑ (cr) 029 ☐ (gs) 045 - 061 = 077 M 093 ] 109 m 125 }</td>
</tr>
<tr>
<td>014 ♦ (so) 030 ▲ (rs) 046 . 062 &gt; 078 N 094 ^ 110 n 126 ~</td>
</tr>
<tr>
<td>015 ★ (si) 031 ▼ (us) 047 / 063 ? 079 O 095 _ 111 o 127 a</td>
</tr>
</tbody>
</table>

72 69 76 76 79 = 'HELLO'
Strings

- As a literal: text surrounded by quotes.
  - "DEEP"
Strings

- As a literal: text surrounded by quotes.
  - "DEEP"
- Each symbol is a character.
Strings

- As a literal: text surrounded by quotes. 
  
  "DEEP"

- Each symbol is a character.

- Unlike numeric types, strings vary in length.
String operations

- **Concatenation**: combine two strings
  - Uses the + symbol
  - `'RACE' + 'CAR'`
String operations

- **Concatenation**: combine two strings
  - Uses the `+` symbol
  - `'RACE' + 'CAR'`

- **Repetition**: repeat a string
  - Uses the `*` symbol
  - `'HELLO'*10`
String operations

- Concatenation: combine two strings
  - Uses the + symbol
  - 'RACE' + 'CAR'

- Repetition: repeat a string
  - Uses the *
  - 'HELLO' *10

- Formatting: used to encode other data as string
  - Uses % symbol
Formatting operator

- Creates string with value inserted

```python
x = 100 * 54
s = "String is: %i" % x
print(s)
```
Formatting operator

- Creates string with value inserted
  - Formats nicely
  - Requires indicator of type inside of string

```python
x = 100 * 54
s = "String is: %i" % x
print(s)
```
Formatting operator

- Creates string with value inserted
  - Formats nicely
  - Requires indicator of type inside of string

```python
x = 100 * 54
s = "String is: %i" % x
print(s)
```
Example

name = "Tao"
grade = 2 / 3
m1 = "Hello, %s!" % name
m2 = "Your grade is:  %f." % grade
print(m1)
print(m2)
Example

name = "Tao"
grade = 2 / 3
m1 = "Hello, %s!" % name
m2 = "Your grade is:  %f." % grade
print(m1)
print(m2)

Hello, Tao!
Your grade is 0.66667.
Example

```python
x = 3
s = ('%i' % (x+1)) * x**(5%x)
print(s)
```

What does this program print?

A 333333333333
B 4444444444
C 9999
D %i%i%i%i%i
Indexing operator

- Extracts single character

a = "FIRE"
a[0]  
The integer is the index.
We count from zero!
If negative, counts down from end.
Extracts single character

```python
a = "FIRE"
a[0]
```
Indexing operator

- Extracts single character
  a = "FIRE"
  a[0]
- The integer is the index.
Indexing operator

- Extracts single character
  
a = "FIRE"
da[0]
- The integer is the index.
- We count from zero!
Indexing operator

- Extracts single character
  a = "FIRE"
  a[0]
- The integer is the index.
- We count from zero!
- If negative, counts down from end.
Question

\[s = "ABCDE"\]
\[i = 3\]
\[x = s[i]\]

What is the value of \(x\)?

- A 'A'
- B 'B'
- C 'C'
- D 'D'
- E 'E'
Question

s = "ABCDE"
i = 25 % 3
y = s[i]

What is the value of y?

A 'A'
B 'B'
C 'C'
D 'D'
E 'E'
s = "ABCDE"
i = (11 % 3) - 7
z = s[i]

What is the value of z?
A 'A'
B 'B'
C 'C'
D 'D'
E 'E'
Lab #2 tomorrow Sunday.