Three-eyed troubles: 
GradeScope, Python, & VSCode...

Aliens Attack! Picobot programmer Z. Dodds was subject of a bizarre encounter yesterday with three-eyed aliens. The trinocular tourists, it seems, were conducting experiments that would help them understand "how humans think."

It seems the aliens used a shrinking ray which let them enter the programmer’s head in order to see what was happening. A witness reports deeply disappointed voices emanating from within.

To escape the attack, Dodds had to turn the ray on himself – as he shrank, the aliens quickly flew off, departing so fast that he was unable to use the reverse ray before they left. "No worries," Dodds mused – in fact, this might help me tomorrow...!

Automatic translation: if it’s possible for human languages... perhaps for CS, too?

This week only: Schedule changes

Thursday: Prof. Dodds will guest lecture.
No office hours in the LAC.

Saturday: Bonus office hours in the LAC, 12—2pm.
The **challenge** of programming...

- **syntax**
  - How it looks
- **semantics**
  - What it does
- **intent**
  - What it should do

**human-typed input** → **machine-produced output** → **human-desired output**

learning a language ~ **syntax**
unavoidable, but not the point

... but learning CS ~ **semantics**
learning how machines *think!*

Inside the machine...

What's behind the scenes (processing + memory):

Computation

Data Storage

Memory!

Random Access Memory

- 512 MB of memory
- a big list of boxes, each with a name, type, location, and value
- **variable** ~ **boxes**
- **on or off**
- **bit** = smallest amt. of info.: 0 or 1
- **byte** = 8 bits
- **word** = 64 bits
All languages use **datatypes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>What is it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>3.14 or 3.0</td>
<td>numeric values with a fractional part, even if the fractional part is .0</td>
</tr>
<tr>
<td>int</td>
<td>42 or 10**100</td>
<td>integers – Python has infinite precision ints!</td>
</tr>
<tr>
<td>bool</td>
<td>True or False</td>
<td>the T/F results from a test or comparison: ==, !=, &lt;, &gt;, &lt;=, &gt;=</td>
</tr>
</tbody>
</table>

Hey! Someone can’t spell! *Boolean values* 

George Boole

Python operators

- **parens** ( )
- **power** **
- **negate** –
- **times, mod, divide** * / % //
- **add, subtract** + –
- **compare** > == <
- **assign** =

It’s not worth remembering all these *+* things! I’d recommend parentheses over precedence.

% the **mod** operator

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7 % 3</td>
<td>9 % 3</td>
</tr>
<tr>
<td>8 % 3</td>
<td>30 % 7</td>
</tr>
</tbody>
</table>

x%y is the **remainder** when x is divided by y

For what values of x are these True?

- x%2 == 0
- x%2 == 1
- x%4 == 0
- x%4 == 3

If x is a year, what happens on these years!? If x is a year, what happens on these years!? What happens on these years, football-wise!? 

// **integer division**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7 // 3</td>
<td></td>
</tr>
<tr>
<td>8 // 3</td>
<td></td>
</tr>
<tr>
<td>9 // 3</td>
<td></td>
</tr>
<tr>
<td>30 // 7</td>
<td></td>
</tr>
</tbody>
</table>

x//y is x/y, rounded-down to an integer
the "equals" operators

This is true – but what is it saying!?

**Quiz**

```
how = works

Run these lines:

x = 41
y = x + 1
z = x + y

What are x, y, and z at this time?

Then run this line:

x = x + y

What are x, y, and z at this time?
```

```
Extra!

a = 11//2
b = a%3
c = b** a+b *a
```

What are the values of a, b, and c after the 3 lines, at left, run?

```
Are numbers enough for everything?

Yes and no...

You need lists of numbers, as well! and strings - lists of characters - too.

Both of these are Python sequences..."
strings: textual data

\[
\begin{align*}
\text{s} &= \text{'scripps'} \\
\text{c} &= \text{'college'}
\end{align*}
\]

type...
\[
type(s)
\]
len
\[
\text{len}(s)
\]
add!
\[
s + c
\]
multiply!!
\[
2*s + 3*c
\]

What did you say?!

Lists ~ collections of any data

\[
M = [4, 7, 100, 42, 5, 47]
\]

strings: textual data

\[
\begin{align*}
\text{s1} &= \text{'ha'} \\
\text{s2} &= \text{'t'}
\end{align*}
\]

Given
\[
\begin{align*}
\text{s1} &= \text{'ha'} \\
\text{s2} &= \text{'t'}
\end{align*}
\]

What are
\[
s1 + s2
\]
\[
2*s1 + s2 + 2*(s1+s2)
\]

Indexing uses [ ]

\[
s = \text{'harvey mudd college'}
\]

Strings
Negative indices...

```
s = 'harvey mudd college'
s[-1] is 'e'
s[-18] is
s[-7] is
s[-0] is
```

Negative indices count *backwards* from the end!

Slicing

```
s = 'harvey mudd college'
s[ : ] slices the string, returning a substring
```

```
s[0:6] is 'harvey'
s[12:18] is 'colleg'
s[17:] is 'ge'
s[: ] is 'harvey mudd college'
```

Try it!

```
pi = [3,1,4,1,5,9]
L = [ 'pi', "isn't", [4,2] ]
M = 'You need parentheses for chemistry !'
```

```
Part 1

What is len(pi) 6
What is len(L)
What is len(L[1])
What is pi[2:4]
What slice of pi is [3,1,4] pi[0:3]
What slice of pi is [3,4,5]
```

```
Part 2

What is L[0] 'pi'
What is L[0][1]
What is L[1:]
What slice of M is 'try'?
What slice of M is 'shoe'?
What is M[9:15]
What is M[:5]
```

Extra Mind Muddler

```
What are pi[0]*(pi[1]+pi[2]) and pi[0]*(pi[1:2]+pi[2:3]) ?
```
Python slices - it dices...

... but wait, there's more!

```
Python functions
```

Functioning in Python

```python
# my own function!
def dbl(x):
    """ returns double its input, x ""
    return 2*x
```

This doesn't look quite right...

Function Fun!

```python
def undo(s):
    """ this "undoes" its input, s ""
    return 'de' + s
```

```
>>> undo('caf')
'decaf'

>>> undo(undo('caf'))
```

Python's keywords

comment for other coders

documentation string for all users

Some of Python's baggage...
Have a dedecaf-ternoon!

morning + evening, too

Just undo it!

This week's lab ~

first two hw problems
how = works

"Quiz"

Run these lines:

\[ x = 41 \]
\[ y = x + 1 \]
\[ z = x + y \]

What are \( x \), \( y \), and \( z \) at this time?

Then run this line:

\[ x = x + y \]

What are \( x \), \( y \), and \( z \) at this time?

Extra!

\[ a = 11 \text{ // } 2 \]
\[ b = a \% 3 \]
\[ c = b ** (a+b) \times a \]

What are the values of \( a \), \( b \), and \( c \) after the 3 lines, at left, run?