This week's classes

Three-eyed? This week, we're 3d'ed!

Homework #11, due 4/22

Python features, motivated by VPython...

Tuples are similar to lists, but they're parenthesized:

- Example: `T = (4, 2)` and `x = (1, 0, 0)`
- Tuple `T` and list `x`

```python
def f(x=3, y=17):
    return 10*x + y
```

Examples of default and named inputs in a function definition:

```
T = ('a', 2, 'z')
```

Connect 4, Part 2

hw11pr2.py

What methods will help?

```
b.colsToWin('O')
b.colsToWin('X')
```

```
b.aiMove('O')
b.aiMove('X')
```

```
hostGame(self)
```

Tuples!

Lists that use parentheses are called **tuples**:

```
T = (4, 2)
```

Tuples are **immutable** lists: you can't change their elements...

...but you can always redefine the whole variable, if you want!

```
T[0] = 42
```

Error!

```
T[0] = 'a'
```

+ Tuples are more memory + time efficient
+ Tuples can be dictionary keys: *lists can’t be*
- But you can’t change tuples’ elements...

```
T = ('a', 'b', 'c')
```
Default – and named – inputs!

Functions can have default input values and can take named inputs.

```python
def f(x=3, y=17):
    return 10*x + y
```

Example of default input values for x and y.

```python
f(x=4, y=2)
```

Example of named input values for x and y.

Function inputs look like tuples, but they're not quite the same...

Named inputs

```python
def f(x=2, y=11):
    return x + 3*y
```

Input your name(s) = ____________________

What will these function calls to f return?

- f(3,1) →
- f(3) →
- f(y=4, x=2) →

What call to f returns the string 'Lalalalala'? (These are tuples – they work like lists!)

What is the shortest call to f returning 42?

Extra... what does this return?

```python
y = 60; x = -6; f(y=x, x=y)
```

```
def f(x=2, y=11):
    return x + 3*y
```

Extra... what does this return?

Documentation

This is for later on. (The documentation links for the browser-based GlowScript are not currently supported.)

- http://www.glowscript.org/ - home page, where you login and access your programs
- http://www.glowscript.org/docs/GlowScriptDocs/index.html - docs for each object, the
- http://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/ - examples (you
A demo of vPython's API:

```python
# the simplest possible vpython program:
box( color = vector(1, 0.5, 0) )
# try changing the color: the components are
# red, green, blue each from 0.0 to 1.0
# then, add a second parameter: size=vector(2.0,1.0,0.1)
# the order of those three #s: Length, Height, Width
# then, a third parameter: axis=vector(2,5,1)
# the order of those three #s: x, y, z
```

What's `box`? What's `color`? What's `vector`? Getting used to everything!

vPython example API call(s)

**API**

... stands for **Application Programming Interface**

A programming description of how to use a software library

**vPython** example API call(s)

**API**

... stands for **Application Programming Interface**

- **constructor** + default arguments; data!
- **shapes + docs**!
- **constructors + methods**!
- **cool stuff**...

**vectors**

- `b.pos, b.vel,...` are vectors
- `b.pos = vector(0,0,0)`
- `b.pos = b.pos + b.vel*0.2`
- `b.vel = vector(1,0,0)`

named components

scalar multiplication

component-by-component addition

let's compare with tuples...
Vectors act like "arrows"

The vector Object

The vector object is not a displayable object but is a powerful aid to 3D computations.

```
vector(x, y, z)
```

Returns a vector object with the given components, which are made to be floating-point (that is, 3 is converted to 3.0).

Vectors can be added or subtracted from each other, or multiplied by an ordinary number. For example,

```
v1 = vector(1, 2, 3)
v2 = vector(10, 20, 30)
print(v1+v2)  # displays <12 22 33>
print(2*v1)  # displays <2 4 6>
```

You can refer to individual components of a vector:

```
v2.x is 10, v2.y is 20, v2.z is 30
```

It is okay to make a vector from a vector: `vector(v2)` is still `vector(10,20,30).

The form `vector(10)` is shorthand for `vector(10, 0, 0)`.

A vector is a Python sequence, so `v2.x` is the same as `v2[0]`, `v2.y` is the same as `v2[1]`, and `v2.z` is the same as `v2[2].` 

Vector functions

The following functions are available for working with vectors:

```
mag(A) = A.mag = |A|, the magnitude of a vector
mag2(A) = A.mag2 = |A|^2, the vector's magnitude squared
norm(A) = A.norm() = A/|A|, a unit vector in the direction of the vector
hat(A) = A.hat = A/|A|, an alternative to A.norm(), based on the fact that unit vectors are customarily written in the form \( \hat{\mathbf{e}} \) with a "\( \hat{\cdot} \)" over the vector
dot(A, B) = A.dot(B) = A dot B, the scalar dot product between two vectors
cross(A, B) = A.cross(B), the vector cross product between two vectors
diff_angle(A, B) = A.diff_angle(B), the angle between two vectors, in radians
proj(A, B) = A.proj(B) = dot(A, norm(B))*norm(B), the vector projection of A along B
comp(A, B) = A.comp(B) = dot(A, norm(B)), the scalar projection of A along B
A.equals(B) is True if A and B have the same components (which means that they have the same magnitude and the same direction)
vec.random() produces a vector each of whose components are random numbers in the range -1 to +1
```

# if the ball ventures too far, restart with random velocity
if mag(ball.pos - origin) > 10.0:
    # mag finds magnitude of a vector
    ball.pos = vector(0,0,0)  # reset the ball.pos (position)
    ball.vel = 4.2*vector.random()  # set a random velocity
    ball.vel.y = 0.0  # with no y component (no vertical)
    print("velocity is now:", ball.vel)

But arrows are arrows!

www.glowscript.org/docs/GlowScriptDocs/index.html

 Documentation!
JavaScript ex.cr. option this week...

... back to vPython:

Libraries ~ Languages

vPython!

Look over this VPython program to determine:

1. How many vPython classes/constructors are here? _____
2. How many named inputs are here? _________________
3. Tricky! How many vPython objects are here? ________
4. What line of code handles collisions? 
5. How does physics work? Where is it?

Let's run it first...

vPython: Linear + Spherical collisions...

At least some of the game needs to be about detecting collisions and changing velocities

Line ~ wall at x=10

How to bounce?
What else to do?
Spherical collisions

0 Zeroth approximation:
Stop \( q \). Undo any overlap.
Make \( r.\text{vel} = q.\text{vel} \).

1 First approximation:
Stop \( q \). Undo any overlap.
Compute \( d = r.\text{pos} - q.\text{pos} \)
Make \( r.\text{vel} = d \)

2 Second approximation:
Same as first, but
Make \( q.\text{vel} = d^\perp \), at 90° from \( d \)

Reality is just three eyes away!

Spherical collisions

0 Zeroth approximation:
Stop \( q \). Undo any overlap.
Make \( r.\text{vel} = q.\text{vel} \).

1 First approximation:
Stop \( q \). Undo any overlap.
Compute \( d = r.\text{pos} - q.\text{pos} \)
Make \( r.\text{vel} = d \)

2 Second approximation:
Same as first, but
Make \( q.\text{vel} = d^\perp \), at 90° from \( d \)

Reality is just two eyes away!

Spherical collisions

0 Zeroth approximation:
Stop \( q \). Undo any overlap.
Make \( r.\text{vel} = q.\text{vel} \).

1 First approximation:
Stop \( q \). Undo any overlap.
Compute \( d = r.\text{pos} - q.\text{pos} \)
Make \( r.\text{vel} = d \)

2 Second approximation:
Same as first, but
Make \( q.\text{vel} = d^\perp \), at 90° from \( d \)

Reality is just one eye away!

vPool – physics?


Equations below...
Lab goals

(0) Try out VPython: Get your bearings (axes!)
(1) Make guided changes to the starter code...
(2) Expand your walls and wall-collisions...
(3) Improve your interaction/game!
(4) Optional: add scoring, enemies, or a moving target, hoops, traps, holes, etc. ~ final project...

Collisions...

# if the ball hits wallA
if ball.pos.z < wallA.pos.z:  # hit - check for z
    ball.pos.z = wallA.pos.z  # bring back into bounds
    ball.vel.z *= -1.0        # reverse the z velocity

# if the ball hits wallB
if ball.pos.x < wallB.pos.x:  # hit - check for x
    ball.pos.x = wallB.pos.x  # bring back into bounds
    ball.vel.x *= -1.0        # reverse the x velocity

# if the ball collides with the alien, give a vertical velocity
if mag(ball.pos - alien.pos) < 1.0:
    print("To infinity and beyond!")
    alien.vel = vector(0,1,0)
# +++ start of EVENT_HANDLING section - separate functions for
# keypresses and mouse clicks...

def keydown_fun(event):
    """ function called with each key pressed """
    ball.color = randcolor()
    key = chr(event.which)
    ri = randint( 0, 10 )
    print("key: ", key, ri)  # prints the key pressed - caps only...

    amt = 0.42  # "strength" of the keypress's velocity changes
    if key in 'W1&':  # all capitals!
        ball.vel = ball.vel + vector(0,0,-amt)
    if key in 'A%J':
        ball.vel = ball.vel + vector(-amt,0,0)
    if key in 'S(K':
        ball.vel = ball.vel + vector(0,0,amt)
    if key in "D'L":
        ball.vel = ball.vel + vector(amt,0,0)
    if key in " ":
        ball.vel = vector(0,0,0)  # reset! via the spacebar
        ball.pos = vector(0,0,0)

    # random change of the sphere's color
    print("printing is great for debugging!")

    # variables make it easy to change
    # behavior across many lines of code
    # (here, all four motion directions)
    # have shortcuts to make your
    # game easier -- or reset it!
GlowScript / vPython examples...

Try out vPython in lab this week!

~ if you enjoy it, consider it for a final project
def f(x=2, y=11):
    return x + 3*y

What will these function calls to f return?

None of the above are 42! but they all share a factor with it! - Eli B. '17

What call to f returns the string 'Lalalalala'?

These are tuples – they work like lists!

What is f( (), (1,0) )?

What is the shortest call to f returning 42?

it's only four characters, too!

Extra... what does this return? y = 60; x = -6; f(y=x, x=y)