Announcements

- Homework
  - I'm so sorry HW3 scores are not back to you yet. My Intro Physics class had a test today, which had priority. I'm hoping to get caught up over the weekend.
  - I've relaxed the deadline for HW4 till Sunday night owing to the delayed presentation of material this week.
- Much of what we are talking about today is fairly abstract, and it seemed trivial to make a super simple Arcade animation lab if you are just doing that for your homework. So lab today will just be open for you to work on your HW if you need to finish it up.
- Polling: `rembold-class.ddns.net`

Review Question

In which below situation would using the Newton-Raphson approach to find a solution be most useful?

A) Finding an integer solution a and b for $a^2 + b^2 = 450$.

B) Finding two prime numbers that sum to 100

C) Finding a value a such that $a^2 + \sin(a) = \frac{1}{a}$

D) Finding the number of ways in which 5 chips can be drawn from a bag containing chips of 7 different colors

Arcade Functions

- Functions can be used to group common tasks together in arcade
- Can use

  ```
  arcade.schedule(<function name>, <interval>)
  ```

  to schedule a function to run at some point in the future
- Forms the basis for animation, but some challenges we need to still overcome

Opinion Poll

```python
def vegas(x):
    y = 2
    for i in range(5):
        x = x + y
    return x

x = 3
z = vegas(x)
print('z =', z)
print('x =', x)
```

Consider the code to the right. When the final value of `x` is printed, what do you think will be the value of `x`?

A) 3

B) 5

C) 13

D) None

A Question of Scope

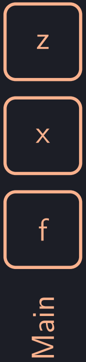
- Functions really do work as self-contained little boxes or environments
  - What happens in `vegas` stays in `vegas`.
- Whenever the python interpreter enters a new function, it gets out a new whiteboard to keep track of that functions variables
  - Changes in the variables on that whiteboard do *not* propagate back to other white boards
- The whiteboard that a function has available to it is called the scope of a function.
- When the Python interpreter finishes evaluating a function (and returns), that scope (or whiteboard) is *thrown away*.

Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```

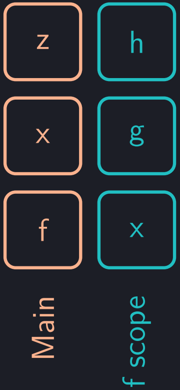
Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



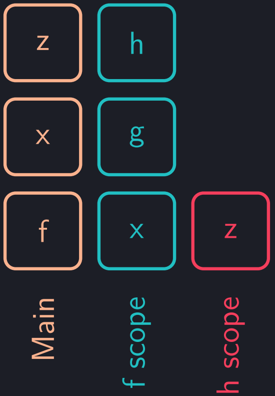
Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



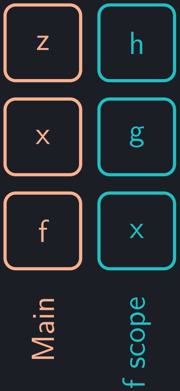
Scoping example

```
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



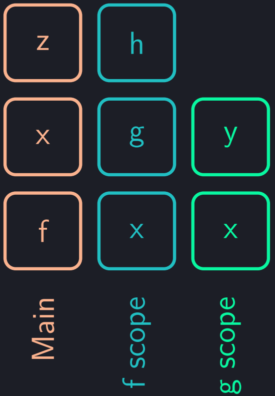
Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



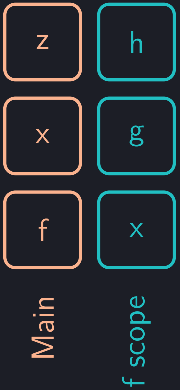
Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



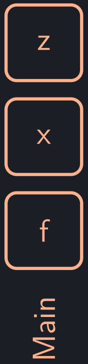
Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



Scoping example

```python
def f(x):
    def g():
        x = 'cat'
        y = 2
        print(x)
    def h():
        z = x
        print(z)
    x = x + 1
    h()
    g()
    return g
x = 5
z = f(x)
```



Scoping Take-aways

- Changes to variables only affect the local scope of that function.
  - The returned value is the only output or change you get from a function.
- A variable is added to the local scope only if provided as a formal parameter or defined as a local variable.
- A reference to a variable not in the current scope will cause the interpreter to look “one scope back” to see if that variable is defined there.
  - If it is, it will use that value
  - Otherwise, you’ll get an error

Scoping Take-aways

- Changes to variables only affect the local scope of that function.
    - The returned value is the only output or change you get from a function.
- A variable is added to the local scope only if provided as a formal parameter or defined as a local variable.
- A reference to a variable not in the current scope will cause the interpreter to look "one scope back" to see if that variable is defined there.
    - If it is, it will use that value
    - Otherwise, you'll get an error

Scoping can be confusing! When in doubt, just think of the function as its own self-contained box. The only thing it knows about the outside world is the inputs, and the only thing the outside world knows about it is what it returns.

Going Global

- Thus far, functions are localized
    - Some inputs in the form of arguments
    - Some output in the from of what is returned
    - This is one of the major benefits of functions!

Going Global

- Thus far, functions are localized
  - Some inputs in the form of arguments
  - Some output in the from of what is returned
  - This is one of the major benefits of functions!
- Sometimes though, we need some more flexibility
  - Take our `on_draw` and `main` functions. Nothing in one can effect the other!

Going Global

- Thus far, functions are localized
  - Some inputs in the form of arguments
  - Some output in the from of what is returned
  - This is one of the major benefits of functions!
- Sometimes though, we need some more flexibility
  - Take our `on_draw` and `main` functions. Nothing in one can effect the other!
- In these situations, one can use what are called global variables

Global Details

- A global variable is defined at the outermost scope of the program
  - This can make it accessible to all functions in the program
- To change a global variable, a function must declare explicitly within its code block that it wants to treat that variable name as a global variable
- Syntax is:

```python
def func(x):
    global y
    y += x

y = 2
func(5)
print(y)
```

I like to move it move it

- Can use globals to update values in `on_draw`
    - Since `global` they will not vanish each time `on_draw` finishes
- Need to draw them at a new position each time `on_draw` is scheduled.
- Don't forget to have `arcade.start_render()` in your `on_draw` function to clear the screen each time!
    - Or don't, if you want a more smeared effect

With Great Power...

- The whole point of functions is to improve readability by keeping everything in a function local
- Global variables destroy that localization
- Wanton use of globals can wreck havoc on the clarity and readability of your code!
    - Use with caution and only where they are really needed
    - Can you achieve something without them? Then it is almost always better to do without