Announcements

• Midterm 1 is on Monday 9/22 from 7pm to 9pm
  ▪ 2 review sessions on Saturday 9/20 3pm–4:30pm and 4:30pm–6pm in 1 Pimentel
  ▪ HKN review session on Sunday 9/21 from 12pm to 3pm in 2060 Valley LSB
• No lecture on Monday
• No lab or office hours next week: Tuesday 9/23, or Wednesday 9/24
• Optional Hog strategy contest ends Wednesday 10/1 @ 11:59pm
Abstraction
Functional Abstractions

```python
def square(x):
    return mul(x, x)
```

```python
def sum_squares(x, y):
    return square(x) + square(y)
```

What does `sum_squares` need to know about `square`?

- Square takes one argument.  
  - Yes

- Square has the intrinsic name `square`.  
  - No

- Square computes the square of a number.  
  - Yes

- Square computes the square by calling `mul`.  
  - No

```python
def square(x):
    return pow(x, 2)
```

```python
def square(x):
    return mul(x, x-1) + x
```

If the name “square” were bound to a built-in function, `sum_squares` would still work identically.
Choosing Names

Names typically don’t matter for correctness

**but**

they matter a lot for composition

<table>
<thead>
<tr>
<th>From:</th>
<th>To:</th>
<th>Name Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>rolled_a_one</td>
<td>Names should convey the meaning or purpose of the values to which they are bound.</td>
</tr>
<tr>
<td>d</td>
<td>dice</td>
<td>The type of value bound to the name is best documented in a function's docstring.</td>
</tr>
<tr>
<td>play_helper</td>
<td>take_turn</td>
<td></td>
</tr>
<tr>
<td>my_int</td>
<td>num_rolls</td>
<td></td>
</tr>
<tr>
<td>l, i, 0</td>
<td>k, i, m</td>
<td>Function names typically convey their effect (print), their behavior (triple), or the value returned (abs).</td>
</tr>
</tbody>
</table>
Which Values Deserve a Name

Reasons to add a new name

Repeated compound expressions:

```
if sqrt(square(a) + square(b)) > 1:
    x = x + sqrt(square(a) + square(b))
```

```
hypotenuse = sqrt(square(a) + square(b))
if hypotenuse > 1:
    x = x + hypotenuse
```

Meaningful parts of complex expressions:

```
x = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```
discriminant = sqrt(square(b) - 4 * a * c)
x = (-b + discriminant) / (2 * a)
```

More Naming Tips

- Names can be long if they help document your code:
  ```
  average_age = average(age, students)
  is preferable to
  # Compute average age of students
  aa = avg(a, st)
  ```

- Names can be short if they represent generic quantities: counts, arbitrary functions, arguments to mathematical operations, etc.

  ```
n, k, i - Usually integers
x, y, z - Usually real numbers
f, g, h - Usually functions
```
Test-Driven Development

Write the test of a function before you write the function.

A test will clarify the domain, range, & behavior of a function.

Tests can help identify tricky edge cases.

Develop incrementally and test each piece before moving on.

You can't depend upon code that hasn't been tested.

Run your old tests again after you make new changes.

Bonus idea: Run your code interactively.

Don't be afraid to experiment with a function after you write it.

Interactive sessions can become doctests. Just copy and paste.
Decorators
Function Decorators

(Demo)

@trace1
def triple(x):
    return 3 * x

Decorated function

is identical to

Why not just use this?

def triple(x):
    return 3 * x
triple = trace1(triple)
Review
What Would Python Print?

The print function returns None. It also displays its arguments (separated by spaces) when it is called.

```
from operator import add, mul

def square(x):
    return mul(x, x)

# A function that takes any argument and returns a function that returns that arg

def delay(arg):
    print('delayed')
    def g():
        return arg
    return g

def g():
    return arg
```

```
This expression | Evaluates to | Interactive Output
---|---|---
5 | 5 | 5
print(5) | None | 5
print(print(5)) | None | 5 None
(delay(delay)()()6)() | 6 | delayed 6
delay(print)()()4) | None | delayed 4 None
```
What Would Python Print?

The print function returns None. It also displays its arguments (separated by spaces) when it is called.

```python
from operator import add, mul

def square(x):
    return mul(x, x)

def pirate(arggg):
    print('matey')

def plunder(arggg):
    return arggg
    return plunder

A function that always returns the identity function

def pirate(arggg):
    print('matey')
    def plunder(arggg):
        return arggg
    return plunder

This expression | Evaluates to | Interactive Output
-----------------|-------------|-------------------
add(pirate(3)(square)(4), 1) | 17 | Matey
                      |     | 17
func square(x) | 16 |
                      |     |       
pirate(pirate(pirate))(5)(7) | Error | Matey
                      |     | Matey
                      |     | Error

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.
```python
def horse(mask):
    horse = mask
    def mask(horse):
        return horse
    return horse(mask)

mask = lambda horse: horse(2)
horse(mask)
```