Announcements

- Midterm 1 is on Monday 2/9 from 7pm to 9pm
  - Go to a room based on the first letter of your @berkeley.edu email:
    145 Dwinelle (A-B, #), 155 Dwinelle (C-K), & 1 Pimentel (L-Z)
  - HKN review session on Saturday 2/7 (2050 VLSB 1-4)
  - Review sessions on Sunday 2/8 (1 Pimentel 1-2:30 & 2:30-4)
  - Info: cs61a.org/exams/midterm1.html
- No lecture on Monday 2/9
- No lab or office hours on Tuesday 2/10 or Wednesday 2/11
- Optional Hog strategy contest concludes Wednesday 2/18 @ 11:59pm

Abstraction

Functional Abstractions

```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>Name</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>true_false</td>
<td>Names should convey the meaning or purpose of the values to which they are bound.</td>
</tr>
<tr>
<td>d</td>
<td>The type of value bound to the name is best documented in a function’s docstring.</td>
</tr>
<tr>
<td>helper</td>
<td>Function names typically convey their effect (print), their behavior (triple), or the value returned (lab).</td>
</tr>
<tr>
<td>my_int</td>
<td></td>
</tr>
<tr>
<td>l, i, o</td>
<td></td>
</tr>
<tr>
<td>k, i, m</td>
<td></td>
</tr>
</tbody>
</table>

Which Values Deserve a Name

Reasons to add a new name:

- Repeated compound expressions:
  ```python
  if sqrt(square(a) + square(b)) > 1:
      x = x + sqrt(square(a) + square(b))
  ```

- Meaningful parts of complex expressions:
  ```python
  x = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
  ```

More Naming Tips:

- Names should be short if they represent generic quantities: counts, arbitrary functions, arguments to mathematical operations, etc.
- Names can be long if they help document your code:
  ```python
  average_age = average(age, students)
  ```

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)

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

is identical to

```python
def triple(x):
    return 3 * x
triple = trace1(triple)
```

Decoration

Why not just use this?

---

Currying

**Function Currying**

```python
def make_adder(n):
    return lambda k: n + k
```

```text
>>> make_adder(2)(3)
5
```

```text
>>> add(2, 3)
5
```

There's a general relationship between these functions.

Curry: Transform a multi-argument function into a single-argument, higher-order function.

---

Review

**Review**

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

What Would Python Print?

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

```text
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 argument.

```python
def delayed(a):
    print('delayed')
    def g():
        return a
    return g
```

Names in nested definitions can refer to their enclosing scope.

```python
def print(a):
    print(a)
```

This expression

```text
print(print(5))
delay(delay)()(6)
delay(print)(4)
```

Evaluates to

Interactive Output

```
5
None
5
None
5
None
5
None
5
None
5
None
5
None
```

---

Global Frame

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

Return Value

```
f1:
    mask

f2:
    lambda

f3:
    mask
```

```
husk (parent=Global)

mask (parent=Global)

func
```