The PCAP framework for controlling complexity

Some simple Python procedures

```python
def square(x):
    return x*x

def average(a,b):
    return (a + b) / 2.0

def meanSquare(a,b):
    return average(square(a), square(b))
```

Hero of Alexandria’s algorithm for computing square roots:

To compute an approximation to the square root of \( x \):

1. Let \( g \) be a guess for the answer
2. Compute an improved guess by taking the average of \( g \) and \( x/g \)
3. Keep improving the guess until it’s good enough.

A procedure for computing square roots:

```python
def goodEnough(guess, x):
    return abs(x-square(guess)) < .00001

def improve(guess,x):
    return average(guess, x/guess)

def sqrtIter(guess,x):
    while not(goodEnough(guess,x)):
        guess=improve(guess,x)
    return guess

def sqrt(x):
    return sqrtIter(1.0,x)
```
Another version of the square root procedure, which uses block structure

```python
def sqrt(x):
    def goodEnough(guess):
        return abs(x - square(guess)) < .00001
    def improve(guess):
        return average(guess, x/guess)
    def iter(guess):
        while not(goodEnough(guess)):
            guess = improve(guess)
        return guess
    return iter(1.0)
```

Computing powers, $b^e$

```python
def expt(b,e):
    if e==0:
        return 1
    else:
        return b*expt(b,e-1)
```

This results in a **linear time process**

Fast exponentiation:

```python
def fastexp(b,e):
    if e == 0:
        return 1
    elif e % 2 == 1:
        return b * fastexp(b,e-1)
    else:
        return square(fastexp(b,e/2))
```

This results in a **logarithmic time process**
A procedure for evaluating polynomials. (Uses list comprehension.)

```python
def evalPoly(p,x):
    m=len(p)
    d=m-1
    return sum([p[i] * x**(d-i) for i in range(m)])
```

Evaluating polynomials with Horner's rule

```python
def horner(p,x):
    result = 0
    for coeff in p:
        result = coeff + x*result
    return result
```

Recap of the PCAP framework (to continue next week)

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitives</td>
<td>+,*,/,==</td>
</tr>
<tr>
<td>Means of combination</td>
<td>if, while, 3*(4+7), list comprehension</td>
</tr>
<tr>
<td>Means of abstraction</td>
<td>def</td>
</tr>
<tr>
<td>Capturing common patterns</td>
<td>??</td>
</tr>
</tbody>
</table>