

Computational Structures in Data Science



UC Berkeley EECS
Adj. Ass. Prof.
Dr. Gerald Friedland

Lecture #4: Higher Order Functions

Hackers steal medical data of US Olympic stars

<http://money.cnn.com/2016/09/13/news/wada-hacked-russian-spies/index.html?iid=surge-story-summary>

September 16, 2016

<http://inst.eecs.berkeley.edu/~cs88>



Administrative issues

- **Concurrent Enrollment: Assume you are in and work on the class!**
- **Data 8 is a requirement. You need to have taken c8 or do it concurrently.**
- **If you can't get into data8, try CS10.**



Computational Concepts today

- More on Recursion
- Runtime

- Higher Order Functions
- Functions as Values
- Functions with functions as argument
- Assignment of function values
- Higher order function patterns
 - Map, Filter, Reduce
- Function factories – create and return functions





More on Recursion

```
def sum_of_squares(n):  
    if n < 1:  
        return 0  
    else:  
        return n**2 + sum_of_squares(n-1)
```

- The sum of no numbers is zero
- The sum of 1^2 through n^2 is n^2 plus the sum of 1^2 through $(n-1)^2$



Recap: Tail Recursion

- All the work happens on the way down the recursion
- On the way back up, just return

```
def sum_up_squares(i, n, accum):  
    """Sum the squares from i to n in incr. order"""  
    if i > n:  
        Base Case  
    else:  
        Tail Recursive Case  
  
>>> sum_up_squares(1, 3, 0)  
14
```



How much ???

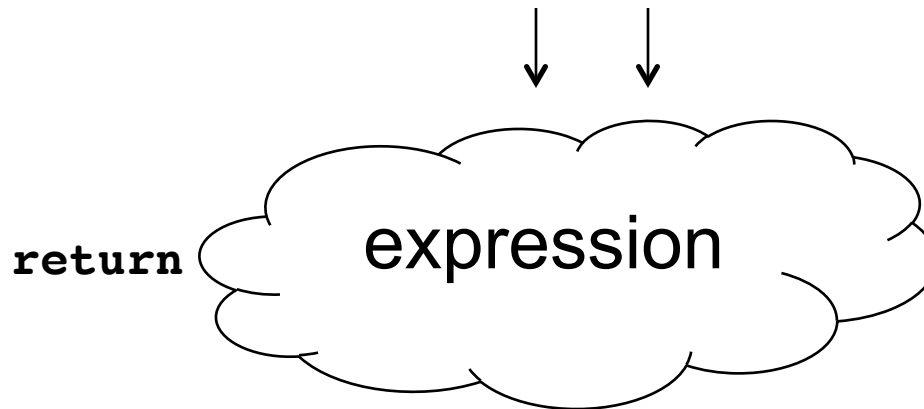
- **Time is required to compute `sum_of_squares(n)`?**
 - Recursively?
 - Iteratively ?
- **Space is required to compute `sum_of_squares(n)`?**
 - Recursively?
 - Iteratively ?
- **Count the frames...**
- **Recursive is linear, iterative is constant !**
- **And what about the order of evaluation?**

Linear
proportional to n
 cn for some c



Recap: Defining Functions

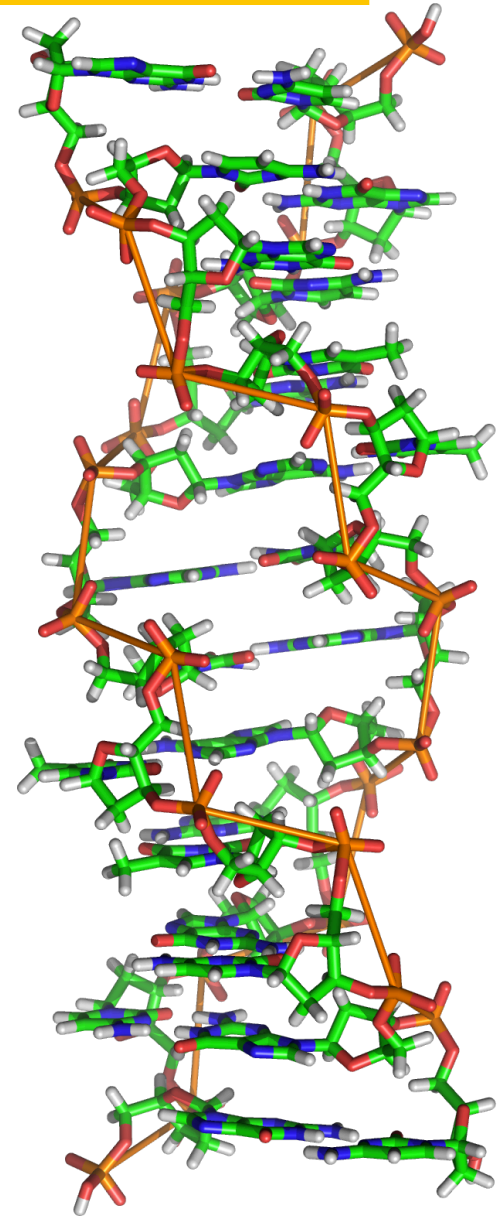
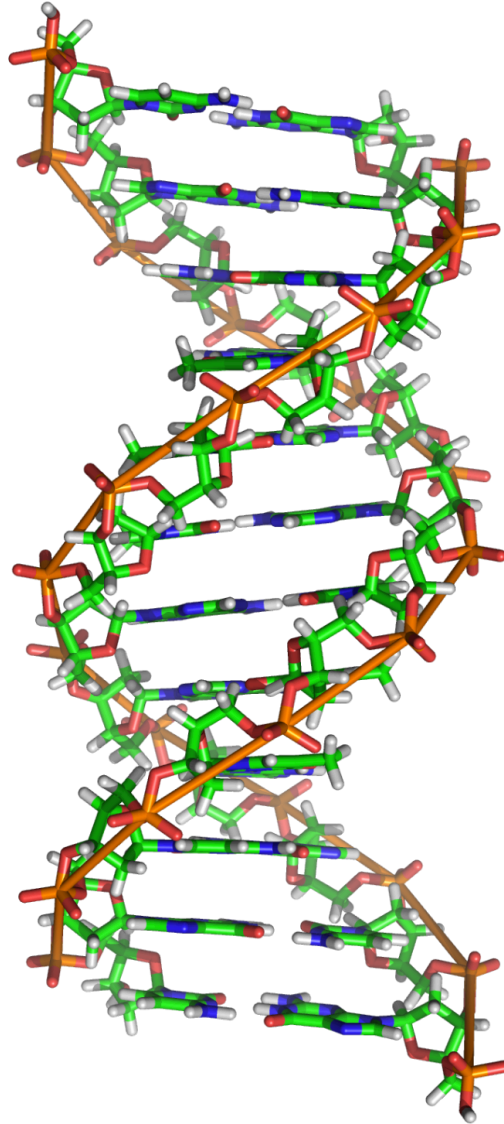
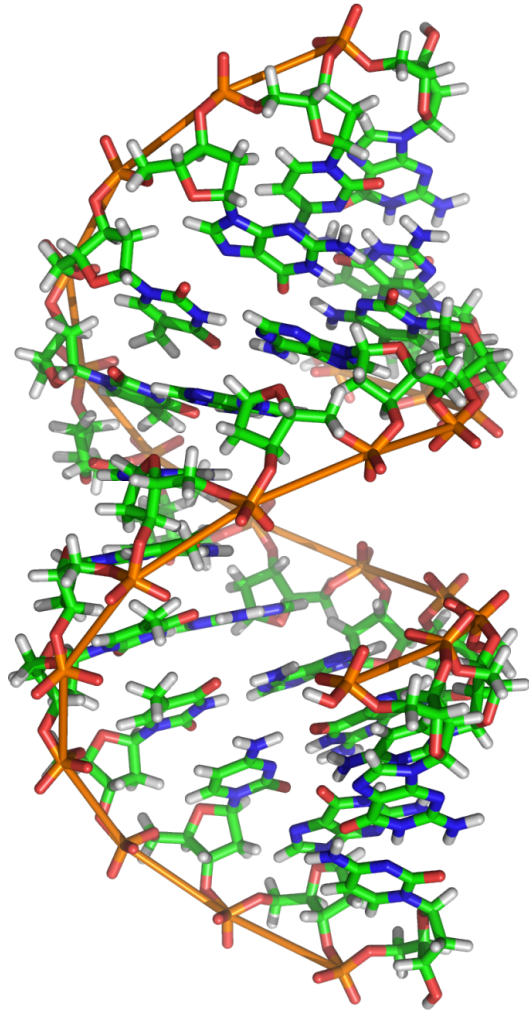
`def` <function name> (<argument list>) :



- **Generalizes an expression or set of statements to apply to lots of instances of the problem**
- **A function should *do one thing well***



Recap: Data or Code?





Higher Order Functions

- Functions that operate on functions
- A function

```
def odd(x):  
    return (x%2==1)  
  
>>> odd(3)  
True
```

Why is this not 'odd' ?

- A function that takes a function arg

```
def filter(fun, s):  
    return [x for x in s if fun(x)]  
  
>>> filter(odd, [0,1,2,3,4,5,6,7])  
[1, 3, 5, 7]
```



Higher Order Functions (cont)

- A function that returns (makes) a function

```
def leq_maker(c):  
    def leq(val):  
        return val <= c  
    return leq
```

```
>>> leq_maker(3)  
<function leq_maker.<locals>.leq at 0x1019d8c80>
```

```
>>> leq_maker(3)(4)  
False
```

```
>>> filter(leq_maker(3), [0,1,2,3,4,5,6,7])  
[0, 1, 2, 3]  
>>>
```



One more example

- **What does this function do?**

```
def split_fun(p, s):  
    """ Returns <you fill this in>."""  
    return [i for i in s if p(i)], [i for i in s if not p(i)]
```

```
>>> split_fun(leq_maker(3), [0,1,2,3,4,5,6])  
([0, 1, 2, 3], [4, 5, 6])
```



Three super important HOFs

```
map(function_to_apply, list_of_inputs)
```

Applies function to each element of the list

```
filter(condition, list_of_inputs)
```

Returns a list of elements for which the condition is true

```
reduce(function, list_of_inputs)
```

Reduces the list to a result, given the function



Recursion with Higher Order Fun

```
def map(f, s):
```

```
  if
```

```
    Base Case
```

```
  else:
```

```
    Recursive Case
```

```
def square(x):
```

```
  return x**2
```

```
>>> map(square, [2,4,6])  
[4, 16, 36]
```

- Divide and conquer



Using HOF to preserve interface

```
def sum_of_squares(n):  
    def sum_upper(i, accum):  
        if i > n:  
            return accum  
        else:  
            return sum_upper(i+1, accum + i*i)  
  
    return sum_upper(1,0)
```

- What are the globals and locals in a call to `sum_upper`?
 - Try [python tutor](#)
- Lexical (static) nesting of function def within def - vs
- Dynamic nesting of function call within call



Recap: Quicksort

- **Break the problem into multiple smaller sub-problems, and Solve them recursively**

```
def split(x, s):
    return [i for i in s if i <= x], [i for i in s if i > x]

def qsort(s):
    """Sort a sequence - split it by the first element,
    sort both parts and put them back together."""
    if not s:
        return []
    else:
        pivot = first(s)
        lessor, more = split(pivot, rest(s))
        return qsort(lessor) + [pivot] + qsort(more)

>>> qsort([3,3,1,4,5,4,3,2,1,17])
[1, 1, 2, 3, 3, 3, 4, 4, 5, 17]
```



Quicksort with HOF

```
def qsort(s):
    """Sort a sequence - split it by the first element,
    sort both parts and put them back together."""

    if not s:
        return []
    else:
        pivot = first(s)
        lessor, more = split_fun(leq_maker(pivot), rest(s))
        return qsort(lessor) + [pivot] + qsort(more)

>>> qsort([3,3,1,4,5,4,3,2,1,17])
[1, 1, 2, 3, 3, 3, 4, 4, 5, 17]
```




How much ???

- Time is required to compute `quicksort(s)`?
- Space is required?
- Name of this recursion scheme?
 - Tree recursion

Logarithmic to $\text{len}(s)$
 $c \cdot \log(\text{len}(s))$ for some c



Questions?

```

/*****\
| Toledo Nanochess (c) Copyright 2009 Oscar Toledo G. All rights reserved |
| 1257 non-blank characters. Evolution from my winning IOCCC 2005 entry. |
| o Use D2D4 algebraic style for movements. biyubi@gmail.com Nov/20/2009 |
| o On promotion add a number for final piece (3=N, 4=B, 5=R, 6=Q) |
| o Press Enter alone for computer to play. |
| o Full legal chess moves. http://www.nanochess.org |
| o Remove these comments to get 1326 bytes source code (*NIX end-of-line) |
\*****/
char*l="ustvrtsuqqqqqqqqyyyyyyyyyy}{|~z|}"
" 76Lsabcdcdba .pknbrq PKNBRQ ?A6J57IKJT576,+--48HLSU";
#define F getchar()&z
#define v X(0,0,0,21,
#define Z while(
#define _ ;if(
#define P return--G,y^=8,
B,i,y,u,b,I[411],*G=I,x=10,z=15,M=1e4;X(w,c,h,e,S,s){int t,o,L,E,d,0=e,N=-M*M,K
=78-h<<x,p,*g,n,*m,A,q,r,C,J,a=y?-x:x;y^=8;G++;d=w|s&&s>=h&&v 0,0)>M;do{ _ o=I[
p=0]}{q=o&z^y _ q<7}{A=q--&2?8:4;C=o-9&z?q["& . $ "]:42;do{r=I[p+=C[l]-64]_!w|p
==w){g=q|p+a-S?0:I+S _!r&(q|A<3|lg)|| (r+1&z^y)>9&&q|A>2){ _ m=!(r-2&7))P G[1]=0,
K;J=n-o&z;E=I[p-a]&z;t=q|E-7?n:(n+=2,6^y);Z n<=t){L=r?l[r&7]*9-189-h-q:0 _ s)L
+=(1-q?l[p/x+5]-l[0/x+5]+l[p%x+6]*--!q-l[0%x+6]+o/16*8:!!m*9)+(q?0:(I[p-1]^n)+
!(I[p+1]^n)+l[n&7]*9-386+!!g*99+(A<2))+!(E^y^9)_ s>h||1<s&s==h&&L>z|d){p[I]=n,0
[I]=m?*g=*m,*m=0:g?*g=0:0;L-=X(s>h|d?0:p,L-N,h+1,G[1],J=q|A>1?0:p,s)_!(h||s-1|B
-0|i-n|p-b|L<-M))P y^=8,u=J;J=q-1|A<7||m||!s|d|r|o<z||v 0,0)>M;0[I]=o;p[I]=r;m?
*m=*g,*g=0:g?*g=9^y:0;}_ L>N){*G=0 _ s>1){ _ h&&c-L<0)P L _!h)i=n,B=0,b=p;N=L;}
n+=J|| (g=I+p,m=p<0?g-3:g+2,*m<z|m[0-p]||I[p+=p-0]);}}Z!r&q>2|| (p=0,q|A>2|o>z&
!r&&+C*-A));}}Z++0>98?0=20:e-0;P N+M*M&&N>-K+1924|d?N:0;}main(){Z++B<121)*G
++=B/x%x<2|B%x<2?7:B/x&4?0:*l++&31;Z B=19){Z B++<99)putchar(B%x?l[B[I]|16]:x)_
x-(B=F)){i=I[B+=(x-F)*x]&z;b=F;b+=(x-F)*x;Z x-(G=F)i=*G^8^y;}else v u,5);v u,
1);}}

```