Announcements

- Course evaluation
  - Your opinion matters!
- Project 5 due Thu
  - Remember second part to be done individually with separate submission. Details on course home.
- Attendance grades
  - Will be posted prior to the final
- Final on Dec 11 in EE 129, 10:30 – 12:30
  - Also posted on course home

Fun things to do with Python

- Build video games
  - http://pygame.org/news.html
  - http://rene.f0o.com/mywiki/PythonGameProgramming

Lego Mindstorms

- Program your robots with Python
  - http://code.google.com/p/nxt-python/

Professional Python Use

- Bio Informatics
  - http://shop.oreilly.com/product/9780596154516.do
- Numpy / Scipy
  - http://numpy.scipy.org/

Final

- Around 40 Questions
- Multiple Choice
- Same format as midterms
- Material includes last week’ s

How to Prepare

- Material from text books: Chapters 3 – 6, 8 – 12. Chapter 7 material limited.
- Recursion, tree encodings
- Complexity (i.e., $O(n^2)$ etc.)
- Algorithms, including binary search, priority queue insertion/deletion, heap sort, merge sort, insertion sort, permutations, anagrams.

How to Prepare

- Past and current midterms
- Past finals
- Read through solutions to projects
  - Is there code you do not understand?
- Read through lab solutions
  - Is there code you do not understand?
- Review the slides

What is the complexity?

```
def myFun(myList):
        n = len(myList)
        i = 1
        while ( i<n):
                myList[i] = i
                i = i*2
        return myList
```

A: O(n)
B: $O(n^2)$
C: O(1)
D: O(log n)

What is the complexity?

```
def trickyReturns(list):
    k = 0
    for w in range(len(list)):
        if(list[w] == 1001):
            return w
        else:
            k=k+1
            return k
```

A: O(n)
B: $O(n^2)$
C: O(1)
D: O(log n)

What does this code do?

```
def  mystery(x):
   if x == 1:
      return 1
   else:
      return x *  mystery(x-1)
```

What does this do?

```
def mystery(x):
    return x + mystery(x-1)
```

Tracing the mystery function

- mystery(5)
- 5 + (mystery(4))
- 5 + (4 + (mystery(3)))
- 5 + (4 + (3 + (mystery(2))))
- ….
- Why are the parentheses important?

What if we had this function?

```
def  mystery(x):
    if x == 0:
        return 0
    else:
        return x  -  mystery(x-1)
```

Tracing the mystery function

- mystery(5)
- 5 - (mystery(4))
- 5 - (4 - (mystery(3)))
- 5 - (4 - (3 - (mystery(2))))
- ….

```
>>> mystery(3)
2
>>> mystery(4)
2
>>> mystery(5)
3
>>>
```

Identify the term that has the largest growth rate

Num of steps	growth term	complexity
$6n + 3$	$6n$	$O(n)$
$2n^2 + 6n + 3$	$2n^2$	$O(n^2)$
$2n^3 + 6n + 3$	$2n^3$	$O(n^3)$
$2n^{10} + 2^n + 3$	2^n	$O(2^n)$
$n! + 2n^{10} + 2^n + 3$	$n!$	$O(n!)$

Comparison of complexities: fastest to slowest

- O(1) – constant time
- O(log n) – logarithmic time
- O(n) – linear time
- O(n log n) – log linear time
- $O(n^2)$ – quadratic time
- $O(2^n)$ – exponential time
- O(n!) – factorial time

What is the terminating condition / base case?

```
def  mystery(x):
   if x == 1:
      return 1
   else:
      return x *  mystery(x-1)
```

What if we call mystery with a negative number?

Now what is the terminating condition / base case?

```
def  mystery(x):
   if x <=1:
      return 1
   else:
      return x *  mystery(x-1)
```

What if we call mystery with a negative number?

What is the output of the following code?

```
list = ['A',1,'B',2,'C',3,'D',4]
myDict = {}
for i in range(0,len(list),2):
        myDict[list[i]] = list[i+1]
```

Past CQ’s

CQ

There are X permutations of 4 objects, where X is:

A. About 12
B. 24
C. 36
D. 60

CQ:

Merge sort can be done using recursion

A. True
B. False
C. Depends

CQ: For large n, which is faster?

A. Running time for input size n is $10^{20}n$

B. Running time for input size n is $10^{-20}\ n^2$

CQ: For large n, which is faster?

A. $10^{20}n$ (seconds)

B. $10^{-20}\,n^2$ (seconds)

A is better when n > 10^{20}

Clicker Question

- What is the complexity of hiding the image in project 3, where the image is $n \times n$ pixels?

A. $O(1)$

B. $O(n)$

C. $O(n^2)$

D. $O(n^3)$

CQ:

What is the last character of the string returned by read()

A. ‘\n’
B. The last character in the last line of the file
C. Depends

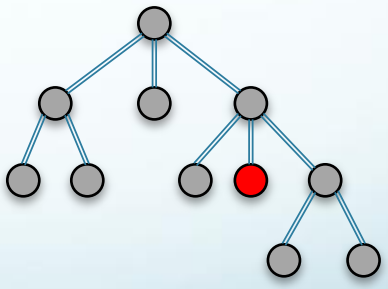
CQ: How do we select ‘Leaf4’ from the Tree?

```
Tree = [‘Root’, [‘Node1’, ‘Leaf0’, ‘Leaf1’],
                ‘Leaf2’,
                [‘Node2’, ‘Leaf3’, ‘Leaf4’, [‘Node3’, ‘Leaf5’, ‘Leaf6’]]]
```

A: Tree[4][3]

B: Tree[3][2]

C: Tree[8]



CQ: How many?

What does the following program print?

```
S = "a,b,,d,e"
print(len(S.split(",")))
```

A. 8

B. 5

C. 4

CQ: which mapping?

- $A = \begin{pmatrix} 0 & 1 & 2 \\ 5 & 4 & 3 \end{pmatrix}$ stored as list A = [0,1,2,5,4,3],
  indexed zero-up: A[1][1] = 4

```
def get_Elt_1(i, k, A):
    p = i*3 + k
    return A[p]
```

```
def get_Elt_2(i, k, A):
    p = k*3 + i
    return A[p]
```

```
def get_Elt_3(i, k, A):
    p = i*3 + k - 1
    return A[p]
```

A) get_Elt_1

B) get_Elt_2

C) get_Elt_3

Announcements

- CoS survey on team experience needed. Link on course home:
  - Go to the section “Science Gains survey”
- Course evaluation
  - Your opinion matters!
- Project 5 due Thu
  - Remember second part to be done individually with separate submission. Details on course home.
- Attendance grades
  - Will be posted prior to the final
- Final on Dec 11 in EE 129, 10:30 – 12:30
  - Also posted on course home

CQ: What is S[:] ?

A. S

B. S[0:0]

C. S[0:len(S)]

CQ:Are these programs equivalent?

1

```
b =
[ 'h' ,' e' ,' l' ,' l
' ,' o' ]
def myFun(l):
    l.append(6)
    return l
print(myFun(b))
```

2

```
b =
[ 'h' ,' e' ,' l' ,' l' ,'
o' ]
def myFun(l):
    l +[6]
    return l
print(myFun(b))
```

A: yes

B: no

CQ:Are these programs equivalent?

1

```
b =
[ 'h' ,' e' ,' l' ,' l
' ,' o' ]
def myFun(l):
    l.append([6])
    return l
print(myFun(b))
```

2

```
b =
[ 'h' ,' e' ,' l' ,' l' ,'
o' ]
def myFun(l):
    l +[6]
    return l
print(myFun(b))
```

A: yes

B: no

CQ: Are these programs equivalent?

1

```
b =
[ 'h' ,' e' ,' l' ,' l
' ,' o' ]
b.insert(len(b), "w" )
print(b)
```

2

```
b =
[ 'h' ,' e' ,' l' ,' l' ,'
o' ]
b.append( "w" )
print(b)
```

A: yes

B: no

Clicker Question: Are these two functions equivalent?

```
def printByCharacter(str)
    i = 0
    while i < len(str):
        print (str[i])
        i = i + 1
```

```
def printByCharacter(str)
    i = 0
    while i < 16:
        print (str[i])
        i = i + 1
```

A: yes

B: no

CQ: Are these programs equivalent?

```
i = 0
x = "This is a string"
while i < len(x):
    print (x[i])
    i = i + 1
```

```
x = "This is a string"
for y in x:
    print (y)
```

A: yes

B: no

CQ: Are these programs equivalent?

```
i = 0
x = "This is a string"
while i < len(x):
    print (x[i])
    i = i + 1
```

```
x = "This is a string"
i = 0 – len(x)
while i < 0:
    print (x[i])
    i = i + 1
```

A: yes

B: no

CQ: Are these programs equivalent?

1

```
1.capitalize()
```

2

```
“1” .capitalize()
```

A: yes

B: no

CQ:Are these programs equivalent?

1

```
for a in range(0, 10, 1):
    print(a)
```

2

```
for a in range(10):
    print(a)
```

A: yes

B: no

CQ: Do these programs print the same text?

1

```
x = 0
y = 0

for k in range(5):
        x = x + k
        y = x + k
print (y)
```

2

```
x = 0
y = 0

for k in range(5):
        x = x + k
y = x + k
print (y)
```

A: Yes

B: No

CQ: Do these functions have the same output?

```
def nested1(a,b):
    for x in range(0, a):
            for y in range (0, b):
                print(x*y)
```

A: yes

B: no

```
def nested2(a,b):
    for y in range(0,b):
            for x in range (0, a):
                print(x*y)
```

CQ:Are these programs equivalent?

1

```
a = 0
while(a < 10):
    print(a)
    a = a+1
```

2

```
for a in range(10):
    print(a)
```

A: yes

B: no

CQ

- Is this list empty?

  [[ ]]

  A: This list is empty

  B: This list is not empty

Clicker Question

1

```
if "False" :
    print( "hi" )
```

2

```
if False:
    print( "hi" )
```

A: 1 and 2 both print

B: only 1 prints

C: only 2 prints

D: neither 1 nor 2 print

Clicker Question

3

```
if eval( "False" ):
    print( "hi" )
```

2

```
if False:
    print( "hi" )
```

A: 3 and 2 both print

B: only 3 prints

C: only 2 prints

D: neither 3 nor 2 print

CQ: Do these programs print the same thing?

1

```
x = 12
if x > 10:
    print (x)
x = x + 1
print (x)
```

2

```
x = 12
if x > 10:
    print(x)
else:
    x = x + 1
print(x)
```

A: yes

B: no

Clicker Question

- Now we can start building useful conditions

```
if x and y > 0:
        print( x , y )
```

- *Does this print if x > 0?*

A: yes

B: no

Clicker Question:
Are these programs equivalent?

1

```
if (x+y) < 10:
  print(x)
if (x+y)>=10:
  print(y)
```

2

```
if(x+y) < 10:
  print(x)
else:
  print(y)
```

A: yes

B: no

CQ: Do these programs print the same thing?

1

```
x = 7
if x > 10:
    print (x)
x = x + 1
print (x)
```

2

```
x = 7
if x > 10:
    print (x)
else:
    x = x + 1
print (x)
```

A: yes

B: no

CQ: Are these programs equivalent?

```
def printCountNTimes(n):
    count = 0
    while (count < n):
        print ('The count is: ', count )
        count = count + 1
```

1

```
printCountNTimes(8)
```

2

```
printCountNTimes(4)
printCountNTimes(4)
```

A: yes

B: no

CQ: Are these programs equivalent?

1

```
x = 7
if x > 10:
    print x
x = x + 1
print x
```

2

```
x = 7
if x > 10:
    print x
else:
    x = x + 1
print x
```

A: yes

B: no

CQ: Precedence

- Consider the expression $a * b + c * d$; which of the three is it equal to?

  A. $(a * b) + (c * d)$

  B. $a * (b + c) * d$

  C. $((a * b) + c) * d$

CQ: Is x global or local?

```
x = 3
def myFun():
    y = 4
    z = x + y
myFun()
```

A: global

B: local

CQ: does this program print 3 or 4?

```
x = 3
def myFun():
    print (x)
x = 4
myFun()
```

A: 3

B: 4

CQ: Do these programs print the same text?

1

```
a = 3
def myFun(a):
    print (a)
myFun(4)
```

2

```
a = 3
print (a)
```

A: yes

B: no

CQ: Do these programs print the same text?

1

```
a = 3
def myFun(b):
    print(b)
print(a)
myFun(3)
```

2

```
a = 3
def myFun(b):
    print(b)
    print(b)
myFun(3)
```

A: yes B: no

CQ: Do these programs print the same text?

1

```
a = 3
def myFun(a):
    print(a)
print(a)
```

2

```
a = 3
def myFun(a):
    print(a)
    print(a)
```

A: yes

B: no

CQ: Do these programs print the same text?

1

```
def myFun(a):
    print(a)
    return a
print(myFun(4))
```

2

```
def myFun(a):
    print(a)
print (myFun(4))
```

A: yes

B: no

Clicker Question

- Which variable name is not valid?

A. a

B. seven

C. 4a

D. _4

CQ: Do these programs print the same text?

1

```
print(Hello)
```

2

```
print( “Hello” )
```

A: yes

B: no

C: maybe

Note on Heaps

- Heap = priority queue
- data structure a full binary tree except possibly the last level which must be filled in left-to-right
- Mapping functions allow encoding heap as a list
- If you take out the first list element, the mapping functions get messed up. Therefor, "plug the hole"
- Insertion works from tree bottom up
- Making a heap by n insertions into an empty hea[is $O(n \log n)$
- Single insertion or deletion is $O(\log n), n$ the heap size
- See week 13…