Topics for Today

- **Exam1**
  - Thursday, February 19, 6:30-7:30pm
  - Location: LWSN 1106 (class room)
  - Closed book; no calculators
  - Can bring Appendix A of Zelle
  - CS exams tend to be graded on a curve
- **Office hour before exam**
  - Hambrusch: Wed 1-2, Thur 2-3
  - Tang: Thur 3-4
- **No class on Monday, February 23**

1

Prepare for the exam

- Slides from class (at class website)
  - Work through the examples and the material covered
  - Slides have little redundancy
- Zelle chapters contain lots of detailed explanation
- Review solutions posted for first three problem sets
- Review solutions to lab problems
- "Learning Python" contains the essentials (plus more)

2

Reviewing for the Exam (1)

- Python language features to know…
  - assignment statements: simultaneous assignments
  - expressions: operators on numbers, strings, lists
  - if statement: matching else clauses
  - for loops: iterables
  - range function: boundary conditions, 3-parameter version
  - strings: Python string operations, string library
  - lists: subscripting, slicing, concatenation, etc., but not "dot methods"
  - arrays: create, zeros, ones, append

3

Reviewing for the Exam (2)

- More Python language features to know…
  - type conversions: float, int, str
  - relational operators: a < b < c, (a < b) and (b < c)
  - logical operators: and, or, not
- Functions
  - defining, calling, parameter passing, returning
  - Understand scope of variables
- Simple programs: defs, loops, conditionals
  - Be able to read them
  - Find simple logic errors

4

Reviewing for the Exam (3)

Not on the exam:

- I/O operations
  - Read from a file, write to a file, formatting output
- Vpython
- Matplotlib
- No syntax details
  - Expect no syntax errors in program segments given

5

About the sample questions posted

- Understand what expressions return
- Understand in what context operations are performed
- Understanding short program segments
  - Trace the execution
  - Be organized and show the value of each parameter
- Understand how functions change or don't change parameters
  - Python scope rules
- Understand how strings and lists can be changed (and are changed in function calls)

6

```
x = 2
y = 10
for j in range(0, y ,x):
    print j
    print x+y
print "done"
```

- Questions often ask “what is printed” or “what is the value”
- Figure out what range(0, y ,x) generates
- This determines the values of j
- Realize that the value of x+y does not change

7

Change a, p = 30, 20 to a, p = 20, 30

```
a, p = 20, 30
while p > 10:
        a = a - 10
        p = p - a
print a, p
```

- While loops don’t necessarily terminate
- Always check whether there is progress towards termination
- the loop given does not terminate

8

Functions (1)

```
x = 22
y = 10

def my_func (a, b, c):
    x = 15      # x is local to my_func
    a = b+c     # a is local to my func
    return x+a

a = ["blue", "green"]
p = my_func(a, 5, y)
```

9

Function (2)

```
x = 22
y = 10
z = 50
def my_func2 (a, b, c):
    x = 15      # x is local to my_func
    a = b+z     # a is local to my func
            # z does not exist in my_func; looks "outside"
    return x+a

a = ["blue", "green"]
p = my_func2(a, 5, y)
```

10

Functions 3

```
x = [33, 34, 38]
y = "test me"

def my_func (a, b):
    a[0] = "white"      # a is a list
    y = "I tried"       # x, y are local to my func
    x = 0
    return

a = ["blue", "green"]
my_func(a, x)
```

11

```
import string

def DNA_complement(str):
    dna_list = list(str) # Convert the string to a list

    # Loop through all of the list indices and convert each letter at that
    # index in the list to its complement.
    for i in range(len(dna_list)):
        if dna_list[i] == 'a':
            dna_list[i] = 't'
        elif dna_list[i] == 't':
            dna_list[i] = 'a'
        elif dna_list[i] == 'c':
            dna_list[i] = 'g'
        elif dna_list[i] == 'g':
            dna_list[i] = 'c'

    # Convert the list back into a string, using a blank string as the
    # separator between each item in the list.
    return ''.join(dna_list)

print DNA_complement("attgcgc")
```

See Learning Python, page 146 for an explanation of the operations

12

```
def DNA_complement2(str):
    new_str = ""
    for i in range(len(str)):
        if str[i] == 'a':
            new_str = new_str + 't'
        elif str[i] == 't':
            new_str = new_str + 'a'
        elif str[i] == 'g':
            new_str = new_str + 'c'
        elif str[i] == 'c':
            new_str = new_str + 'g'

    return new_str
```

A solution without using lists:

- **need to build up the new string;**
- **uses concatenation**

Understand the if-construction

13