CSE 111 Bio: Program Design I
Lecture 9: Exam I Review

Tanya Berger-Wolf (CS) & Boris Igić (Bio)
University of Illinois, Chicago
September 20, 2016

This Class

- Discuss midterm topics
- Go over practice examples
- Answer any questions

Learning programming…

1) Expect it to be different!

2) Don't feel you need to memorize it

3) Immersion == Experimentation

The Secret of Happiness is…
(in programming)

- Don't memorize!
- Look at examples of similar problems
- Experiment
- Syntax that looks weird now will become second nature soon

Midterm I: Topics Covered

- Variables
- Mathematical operators
- Statements
- Types
- Strings, lists, slicing
- Files
- Simple functions
- Map-reduce

Midterm I: Topics, continued

- DNA
- RNA
- Proteins
- Central Dogma

Any general questions?

What type of variable would you use to store the number of zebras in a population?

A. int

B. float

C. list

D. boolean

E. string

F. More than one of the above

What type would you use for a variable to store the fraction of the population with a specific gene?

A. int
B. float
C. list
D. boolean
E. string
F. More than one of the above

What type would you use for a variable to store whether or not there is a lion in the area?

A. int
B. float
C. list
D. boolean
E. string
F. More than one of the above

What type would you use for a variable for the ID/name of that lion?

A. int
B. float
C. list
D. boolean
E. string
F. More than one of the above

What type would you use for the set of the IDs of all the zebras in the population?

A. int
B. float
C. list
D. boolean
E. string
F. More than one of the above

DNAseq="gattaca"

- Write an expression that returns the last character in DNAseq. DNAseq[-1] or DNAseq[len(DNAseq)-1]
- Write an expression that returns every other position in this sequence, starting with the first DNAseq[::2]
- What is the difference between this DNA and mRNA string? T's changed to U's
- Write the sequence of the first mRNA `antisense' codon, (3'-to-5'). Reverse complement, T->U aca->tgt->**ugu**

Suppose you have the following function defined:

```
def square(x) :
    return x**2
```

Write a function that takes integers x and y and returns x squares of numbers in a row, starting with y^2. Don't forget the docstring!

```
def myFun(x,y) :
    """takes integers x and y and returns x squares of numbers in a row, starting with y^2"""
    return map(square, range(y,y+x))
```

Write a function that takes a file name (string) as an argument, reads the file by that name, appends the reverse of the file contents to the original contents of the file, and writes the modified string back to the file without overwriting the original.

```
def myFun(fileName) :
    f = open(fileName, "r+")
    a = f.read()
    a = a+ a[ : :-1]
    f.write(a)
    f.close()
```

You can also open with "r", then close(), then open again with "a"

```
def foo(x) :
    x = x + x[1]
    print("foo:" + x)

def bar(y) :
    y = y*2
    print("bar:", y)
    foo(y)
    return y

def mainFun( ) :
    z = 3
    w = bar("Madness"*z)
    print(z)
    print(w)
```

- What does bar("silly") return?

```
bar("silly")              y = "silly"
y = y*2                   y = "sillysilly"
print("bar:", y)    doesn't change y
 foo("sillysilly")        x = "sillysilly"
 x = x+x[1]               x = "sillysillyi"
 print("foo:", x)  doesn't change x or y
return y                  "sillysilly"
```

- What is the output of bar("silly")? Make sure to write not only the return statement but everything that happens when the function is called.

"bar:sillysilly"
"foo:sillysillyi"
"sillysilly"

- What is the output of mainFun( )? You may find it helpful to draw a picture of what's happening in memory as you trace through the program.

"bar:MadnessMadnessMadnessMadnessMadnessMadness"
"foo:MadnessMadnessMadnessMadnessMadnessMadnessa"
3
"MadnessMadnessMadnessMadnessMadnessMadness"

Try this…

Write a function called span that returns the difference between the maximum and minimum numbers in a list…

```
>>> span([3, 1, 42, 7])
41
>>> span([42, 42, 42, 42])
0
```

```
min(x, y)
max(x, y)
```

These are built-in to Python!

Now try this…

1. Write a python function called gauss that takes as input a positive integer N and returns the sum 1 + 2 + … + N

2. Write a python function called sumOfSquares that takes as input a positive integer N and returns the sum $1^2 + 2^2 + 3^2 + \ldots + N^2$

You can write extra "helper" functions too!