Example Solutions to Practice Lab Exams

Jane Shevtsov

11/22/2017

Note: this set of example solutions shows a different style of programming and way to follow the instructions, as well as the bare minimum of commenting.

Solutions to Practice Lab Exams
Follow the instructions below:

- Write your solution to each question in the cell just below the question.
- If you need any space to try code that will not be graded, use the space at the bottom below the word “Scratchwork”.
- There are multiple ways to solve many of these problems, so we will give credit (including partial credit) based on how well your code solves the problem given.
- You should also use good programming style, such as descriptive variable/function names and axes labels.
- Comments are only required if specifically requested, but feel free to use them whenever you want.
- If you’re not sure how to code something, use comments to describe what the code should do and code as much as you can to earn partial credit.

By writing your name below, you acknowledge that you read the instructions above and have not looked at anyone else’s code or any other resources outside this worksheet.

# Name: Jane Shevtsov
# Section: none (I'm one of the developers of LS30 and authors of the textbook!)
# Date: 11/22/2017

Practice Exam # 1

1. (10 points) Write a function that prints "It’s hot" if the temperature is greater than 85, "It’s cold" if temperature is less than 65, and "Not bad" if the temperature is between 65 and 85, inclusive. Test the function with three different temperatures.

# 1

def tempopt(temp):
    if temp > 85:

print "It's hot"
elif temp < 65:
    print "It's cold"
else:
    print "Not bad"

tempopt(90)
tempopt(80)
tempopt(50)

It's hot
Not bad
It's cold

2. (10 points) You are studying populations of penguins and marine iguanas on a beach in the Galapagos. Over five years, the penguin population at your study site has been 62, 93, 75, 56 and 76. In the same years, the marine iguana population has been 34, 21, 15, 25 and 34. Plot the system’s states in penguin-marine iguana space, making the points green and large.

```python
penguin = [62, 93, 75, 56, 76]
iguana = [34, 21, 15, 25, 34]
list_plot(zip(penguin, iguana), color="green", size=40, axes_labels=['Penguins', 'Marine Iguanas'])
```

3. (10 points) The simulation script below has five errors. Correct the errors and explain what each line does in a comment.

```python
var("N,P")
t = srange(0,100,0.1)
sol=desolve_odeint([0.5*N - 0.01*N*P, 0.5*0.01*N*P - 0.2*P], ics=[50,75], dvars=[N,P], times=t)
list_plot((t, sol[:,0])) + plot(zip(t,sol[:,1]), color="red")
```

# 3.
4. (10 points) Write a script that calculates the factorial of an integer $n$ (written $n!$). To calculate a factorial, we take the number, $n$, and multiply it by all of the integers between 1 and $n$, inclusive. For example, $2!$ is $2 \times 1 = 2$ and $3! = 3 \times 2 \times 1 = 6$. Test the script with two different values of $n$.

```python
# 4
n=4
dec = 1
for i in srange(1,n+1):
    fac = fac*i
fac

n=5
dec = 1
for i in srange(1,n+1):
    fac = fac*i
fac

n=10
dec = 1
for i in srange(1,n+1):
    fac = fac*i
fac
```
Practice Exam # 2

1. (10 points) Plot the function $f(x) = x^2$ in red for values of $x$ ranging between -10 and 10. Plot the function $g(x) = x^3$ in green for values of $x$ ranging between -5 and 5. Overlay these plots.
(Note: you are not required to define the functions.)

```python
# 1
plot(x^2, (x, -5 , 5) , color="red") + plot(x^3, (x, -5 , 5) , color="green", axes_labels=["x", "f(x)"])
```

2. (10 points) The script below should iterate the function $f(x) = 3x$ five times with an initial value of 1, but has five errors. Correct the errors and explain what each line does in a comment. If the script works correctly, it should generate the output at the bottom of the script. In a new cell, convert the script into a function that takes the number of iterations as input and returns the list of values. Test the function for three different numbers of iterations.

```python
# 2
def mult3func(iter):
    mult3 = [1] # Creates list to hold values
    nums = srange(0,iter) # Creates list for counting how many times to run loop
```
for n in nums:  # Creates for loop
test = 3*mult3[n]  # Multiplies last value of mult3 by 3
mult3.append(test)  # Appends test to mult3
return mult3  # Shows result

# Note: make sure you don't use the same name for both a variable \nand a function! That's what Jane did when she took the practice \nexam!

[1, 3, 9, 27, 81, 243]
[1, 3, 9, 27, 81, 243, 729, 2187]
[1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049]

3. (10 points) Modify the example code below to simulate the model given for time steps (step size) of 0.2 and initial conditions of N=100 and P=20. Plot the trajectory as a smooth curve in pink. Plot the vector field for the model given with N ranging between 0 and 150 and P ranging between 0 and 120.

var("N,P")
t = srange(0,100,0.2)
sol=desolve_odeint([0.5*N - 0.01*N*P, 0.5*0.01*N*P - 0.2*P], ics=[50,75], dvars=[N,P], times=t)
list_plot(zip(sol[:,0], sol[:,1]), plotjoined=True, color="red")

# 3.

var("N,P")
t = srange(0,100,0.2)
sol=desolve_odeint([0.5*N - 0.01*N*P, 0.5*0.01*N*P - 0.2*P], ics=[100,20], dvars=[N,P], times=t)
list_plot(zip(sol[:,0], sol[:,1]), plotjoined=True, color="pink", title="Trajectory", axes_labels=["N", "P"])

# Titles aren't required, but always useful
# You can also just use sol - this code basically separates sol \ninto its two columns and then zips them back together! It's how \nwe used to teach plotting trajectories.

plot_vector_field([0.5*N - 0.01*N*P, 0.5*0.01*N*P - 0.2*P],(N,0,150),(P,0,120), axes_labels=["N", "P"])
4. (10 points) Write a script that generates the complementary strand (as a list) to a DNA sequence with the variable name dna_seq with any number of nucleotides (A,T,C,G) (A and T are complementary, C and G are complementary). For example, if the script is given dna_seq = "ACTGACTGAC", it should print "TGACTGACTG". The variable dna_seq can be a string or list of characters such as ["A","C","T"] with only A,C,T,G. You don’t need to consider the possibility of other characters.

```python
# 4
dna_seq = "ACTGACTGAC"
newseq=[]
for base in dna_seq:
    if base == "A":
        comp = "T"
    elif base == "T":
        comp = "A"
    else:
        comp = """# 4
```
```python
comp = "A"
elif base == "C":
    comp = "G"
elif base == "G":
    comp = "C"
newseq.append(comp)
newseq
```