Chapter 12

Answers to Exercises

12.1 Chapter 1: Introduction to Python

Conceptual Questions
1. Python reads the code provided the same way that we read, left to right, top to bottom!

2. In Python, brackets are used to make lists. Parentheses are used to make tuples. However, when indexing for any of these collections, only brackets are used. Parentheses are also used when providing inputs for functions for example; when using NumPy's function for the mean, you would use parentheses in "np.mean(XXX). It should be noted that all shells of the parentheses should be accounted for as not doing so can result in painful syntax errors.

3. A tuple is defined as a sequence of items that cannot be changed, also characterized as *immutable*. A list is defined as a sequence of items that can be changed, also known as *mutable*. When writing tuples use a set of parentheses, and for lists use brackets.

4. In Python, indentations are necessary as they indicate where the statement belongs to in terms of blocks of code. In Python, indentations indicate specifically where in the block of code a statement belongs.
   Additionally, the amount of indentation is important because if there is extra space or missing space within a block of Python code, it likely will cause a syntax error or change your desired outcome.

5. In Python, the four most common types of data that Python supports with variables are "int", "float", "bool", and "str". Int can be either a positive or negative integer also known as a whole number. Floats, also known as floating point numbers, are positive or negative numbers that have a decimal point. These can be used in scientific notation. Bool is short for

Boolean, which is the variable with two possible values, True and False. When writing Booleans, the first letter of the variable must be capitalized (True/False) so that Python can recognize the key words. Python confirms this by displaying the syntax coloring. Finally, str means string, and is used when you want to write text into the Python code. When you want to use a string, you must contain the text wanted within quotation marks so that Python does not read the text as a command.

Python Exercises

Question 1:

```
varOne = 20
varTwo = 5

# Part A
addVar = varOne + varTwo
print(addVar)

# Part B
divVar = varOne / varTwo
print(divVar)

# Part C
multVar = addVar * divVar
print(multVar)
```

Question 2:

```
L1 = ['berries', 'cake', 'chips', 'candy', 'apple']
print(L1)
print(L1[2:4])

L2 = ['soccer', 'baseball', 'golf']
print(L2)
print(sorted(L2))
```

Question 3:

```
import numpy as np

vector1 = np.arange(5, 15, 2)
print(vector1 ** 2)
```

Question 4:

```
import numpy as np

matrix1 = np.zeros((3, 6))
print(matrix1)
```

```
matrix2 = np.array([['Chocolate','Strawberry','Vanilla'],
['RootBeer','Orange','Pepsi']])
print(matrix2)
```

Question 5:

```
for x in range(58, 71, 1):
  ifx%2 ==0:
    print('The number ' + str(x) + ' is an even number.')
  else:
    print('The number ' + str(x) + ' is an odd number.')
```

12.2 Chapter 2: Introduction to Computational Neuroscience

1. No suggested answer, just enjoy learning about the automata. It turns out that Conway's Game of Life is Turing-complete, which means that anything that can be computed can be computed with a GoL! (Might take a while, though!)

2. The methods differed (on an implementation level), but the mental math and process of addition is the same. You took the input (the two numbers), processed them through an algorithm (symbolized by the addition symbol), and your output, hopefully, was 10.

3. TBD

4. TBD

5. TBD

6. TBD

12.3 Chapter 3: Passive Membrane Models

12.3.1 Conceptual Questions

1. The neuron begins at resting potential. In this state, the membrane potential is about -70 mV, all voltage gated channels (VG) closed, and potassium may move through leak channels. Current will eventually enter through the ligand-gated channels raising the membrane potential to around -50 mV. At this voltage, a threshold is met and voltage-gated Na+ channels open. Na+ ions flood into the cell and further depolarize it in a positive feedback loop. Eventually voltage-gated Na+ channels will be inactivated,

marking the absolute refractory period. Voltage-gated K+ channels will open (usually around 40 mV), and K+ rushes out of the cell. Following this period, the relative refractory period occurs in which voltage-gated Na+ channels are no longer inactivated, but the cell is hyperpolarized, making it more difficult, but still possible, for another action potential to occur.

2. The resting potential of a neuron is also known as an ionic steady state because the ion movement is steadily maintaining a resting membrane potential of around -70 mV. Ions naturally want to even out a concentration gradient to achieve equilibrium. When these ions are forced to maintain an unequal gradient there is a resulting potential energy. On the other hand, ions are attracted to opposite charges (positive and negative attract). This potential is how much energy is required to oppose net diffusion for an ion and is known as the Nernst potential. Each of the major ions (Na+, K+, Cl-, Ca2+) have a Nernst potential, and a weighted average of these Nernst potentials. The weights are provided by the relative permeability of the various ions across the membrane. These permeabilities are related to the density of ion channels in the membrane, as well as the extent to which the channels will allow ions to flow.

3. The reversal potential for a specific ion can be calculated with the Nernst equation. However, a more realistic idea of the reversal potential for the whole membrane would incorporate more than one ion. The way to calculate this information is with the Goldman-Hodgkin-Katz (GHK) equation. The GHK equation can only be calculated when more than one of the transmembrane ion channels is open.

4. Different ways that ions can cross membranes include:

Voltage-gated ion channels: when a voltage threshold is achieved, the channel will open for a specific amount of time allowing a particular ion to flow down its concentration gradient.

Ligand-gated ion channels: Ligand-gated channels open when a particular hormone or neurotransmitter binds to the postsynaptic cell. After binding, specific ions can flow down their concentration gradient and into the cell.

Mechanically-Gated Channels: Are opened by mechanical movement such as pressure on someone's skin and allow ions to flow in.

Leak Channels: Leak channels are quite self-explanatory. In the cell, there are channels in which potassium and occasionally chloride can very slowly leak out and diffuse out of the cell.

Pumps: There are also pumps that can move particular ions against their concentration gradient. A great example of this is the sodium potassium pump which exchanges three Na+ ions out for 2 K+ ions in with energy from ATP.

5. The correct matches are:

Water:: Ions
Water naturally wants to fill space just like ions naturally wanting to balance concentration gradients. Water will build potential energy when blocked from an area that it wants to flow to.

Dam::Voltage-gated ion channels
A dam in a river is just like a closed voltage gated channel. The water is higher on one side than the other so potential energy is built up and then transferred into kinetic energy when the dam opens. This is just like sodium ions wanting to flow into the cell when a channel opens.

Water stream::Ion pump
Additionally, forcing water against gravity involves externally applied energy. In cells, ATP is used to power pumps that can move ions to places they don't want to go just like a generator powering a water jet.

6. The leaky integrate and fire model considers the membrane to be a resistor-capacitor (RC) circuit. It is a resistor because ions cannot flow freely across the membrane, but instead need to flow through selective channels. It is a capacitor because the small size of the membrane makes for effective separation of charge (negative ions on the inside and positive on the outside). If this circuit is provided with an external current, the change in potential across the membrane is a product of both resistive and capacitive currents that flow. However, the model does not fire on its own. As an experimenter, we assign a threshold and state that the neuron has fired once that threshold has been exceeded.

7. Some of the ways in which the leaky integrate and fire model is unrealistic include:

- It does not model individual ion conductances, but rather places them into one leak term.
- It does not fire on its own. We have to code the action potentials in by hand.
- It does not include any spatial terms for the action potential propagating down the axon.

8. TBD

9. TBD

12.3.2 Coding Questions

1a.

```
# Step 1: Import libraries
from numpy import log

# Step 2: Define general function
```

```
def Eion(z, o, i):
  return z * log(o/i)

# Step 3: Use function to print the Nernst potential of each ion
# Step 3.1: Na+ ion
print(Eion(1, 2, 1))
# Step 3.2: K+ ion
print(Eion(1, 1, 2))
# Step 3.3: Cl- ion
print(Eion(-1, 2, 1))

# Answer: Na+ is positive, K+ and Cl- are negative
```

1b.

```
# Step 1: Edit general function to use physiological values
def Eion(z, o, i):
  return 26.5/z * log(o/i)

# Step 2:  Use function to print the Nernst potential of each ion
# Step 2.1: Na+ ion
print(Eion(1, 145, 15))
# Step 2.2: K+ ion
print(Eion(1, 4, 150))
# Step 2.3: Cl- ion
print(Eion(-1, 110, 11))

# Answers:
# Na+: 60.1 mV
# K+: -96 mV
# Cl-: -61 mV
```

2a

```
# Step 1: Import the necessary Python libraries
from numpy import log

# Step 2: Set up the general function
def Vrest(Nao, Nai, Ko, Ki, pCl, Cli, Clo):
  return 26.5 * log((0.05*Nao + 1*Ko + pCl*Cli) / (0.05*Nai + 1*Ki + pCl*Clo))

# Step 3: Use the function along with the provided values
print(Vrest(145, 15, 4, 150, 0.45, 10, 110))

#Answer: -67 mV
```

2b

```
# Change the values for pCl and [out] and [in] for Cl-
print(Vrest(145, 15, 4, 150, 0.1, 110, 10))

# Answer: Vrest=-50.9
# This resting potential is less negative, making it more likely to fire an action potential.
```

12.4 Chapter 4: Hodgkin and Huxley

12.4.1 Conceptual Questions:

1. TBD
2. An example of *negative* feedback is:

- In the rising phase of the action potential, there is less driving force on Na+ as Vm approaches the Nernst potential for Na+.

An example of positive feedback is:

- An initial depolarization of the membrane causing voltage-gated Na+ channels to open. This allows for more Na+ to enter, which further depolarizes the cell.

3. The main difference between the leaky integrate and fire model, and the Hodgkin and Huxley model is the inclusion of the Na+ and K+ channels instead of just using the leak channels. Including the Na+ and K+ voltage gated channels allows the H-H model to spike on its own. In the Integrate and Fire model, one must include artificial spikes to see the relationship between the applied current, the membrane threshold and the conductance of the leak channels.
4. TBD
5. TBD

12.4.2 Coding Questions:

Problem 1, part 1:

```
tStimStart =  100
tStimEnd  =  400
tFinal  =  500
dt  =  0.1
Ie   =  100
RestingPotential = -65
HHmodel(Ie, dt, tFinal, tStimStart, tStimEnd, RestingPotential)
```

Problem 1, part 2: Set I_e to 250. The firing rate increases.

Problem 2, part 1:

```
# Note: as long as the range includes 13, this will work
Ie = np.arange(20)
HHmodel_threshold(Ie)
```

Problem 2, part 2:

```
Ie = [12,13]
RestingPotential = -65
HHmodel_compare(Ie,RestingPotential)
```

Problem 2, part 3:
restingPotential = -60

Problem 3, part 1:

```
if timeVec[v] >= 300:
  beta_h = 0
```

Problem 3, part 2:

```
if timeVec[v] > 600:
  alpha_m = 0
```

Problem 3, part 3:
If TTX was indeed a cure, then there should be action potentials occurring in that range between 600 and 800 ms.

12.5 Chapter 5: Firing Rates

12.5.1 Conceptual Questions

1. The spike count rate is defined as the spike count during an interval of time, divided by the length of time, average over time. While this is an easy method to apply mathematically, it is not a continuous method. This means that this method does not reflect that the rates change over time, instead it assumes that rate stays the same.

2. The spike density rate is defined as the number of spikes observed during a time interval divided by the number of trials that were run, then multiplied by one divided by the length of the time interval, average over several runs. However, this requires multiple trial runs for usable data. As a result, it is not a good model for real world situations, as events do not happen repeatedly.

3. The spike density rate is defined as the number of spikes observed during a time interval divided by the number of neurons on which trials were run, then multiplied by one divided by the length of the time interval, average over several neurons. This method is continuous, and it is all done in one

trial, instead of several trial runs. However, depending on the electrode to work simultaneously makes this method hard to execute. Additionally, the neurons need to all have the same basic computation.

4. TBD

12.5.2 Coding Questions

Problem 1, parts a and b.

```
# Import the necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# Initialize data structures
# Hint: your simulation runs for 2000 ms
timeVec = np.arange(2000)

# Hint: you may express your desired rate as a nSpikes / 1000 ms probability
probability = 50/1000

# Hint: you want the default value to be zero
spikes = np.zeros(len(timeVec))

# Loop through each time point - fill in missing code
for i in range(len(timeVec)):
  if np.random.rand() < probability:
    # fill in this key line
    spikes[i] = 1

# Compute the spike count rate here
spikeCountRate = np.sum(spikes) / 2

# Print the spike count rate
print('The firing rate was: {} Hz'.format(spikeCountRate))

# Create a figure of the spike train
plt.figure()
plt.plot(timeVec, spikes)
plt.title('Spikes versus Time')
plt.xlabel('Time (ms)')
plt.ylabel('Spikes')
```

Problem 1, part c.

```
# Import necessary libraries
import matplotlib.pyplot as plt
```

```
import numpy as np

# Find locations of spikes in vector
spikeLocs, = np.where(spikes == 1)

# Pre-allocate space for ISIs
ISI = np.zeros(len(spikeLocs)-1)

# Loop through all spikes
for i in range(len(ISI)):
  ISI[i] = spikeLocs[i+1]-spikeLocs[i]

# Create a plot of the results
plt.figure()
plt.hist(ISI)
plt.ylabel('Counts')
plt.xlabel('ISI in ms')
```

Problem 2, part a.

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#fill in the time vector and the spike frequency value of 50 Hz
timeVec = np.arange(2000)
p1 = 50/1000

#Allocate space for the 40 trials x 2000 ms array
allTrials = np.zeros((40, 2000))

#Create a Poisson spike generator, that loops 40 times for all 40 trials
for i in range(40):

    # For each trial, use an inner loop to represent all 2000 time points
    for v in range(len(timeVec)):
        if np.random.rand() < p1:
            allTrials[i, v] = 1

# Create the raster plot
for j in range(40): # How many trials are we running?
    spikes, =  np.nonzero(allTrials[j,:])#Fill in Spikes to represent every spike foun

    spikeTimes = timeVec[spikes]
    theseSpikes = np.ones(len(spikes))*j+1
```

```
    plt.scatter(spikeTimes, theseSpikes, s=2, c='k')

#Fill in the labels of the x, y axis
plt.title('Poisson Model Firing Rates for 40 Trials')
plt.xlabel('Time (ms)')
plt.ylabel('Trial number')
```

Problem 2, part b

```
fano = np.var(allTrials) / np.mean(allTrials)
print(fano)
```

12.6 Chapter 6: Reverse Correlation and Receptive Field Mapping

12.6.1 Conceptual Questions

1. A white noise stimulus is first generated to record the neuron's response to it. This random selection of stimuli is important in obtaining an independent assessment of the neuron's preference. As different white noise stimuli are presented, the action potentials that the neuron fires are being recorded. A time window is then chosen to observe each time there is a spike. By creating a window of time, we can examine the response of the neuron to the stimulus over a particular time and detect the preferred time window for the neuron in which it will respond to the stimulus. The stimulus presented each time before a spike at their respective time windows is finally stored in an array and averaged over all the time steps to obtain the spike-triggered average.

2. Point A has a higher level of resulting convolution because the stimulus and the STA (acting as the signal and kernel respectively), are aligned. This means that the stimulus has a lot of similarity with the preferred feature of the neuron at the given point of time, allowing the neuron to have a high firing rate. Point B, on the other hand, has very minimal neuronal firing due to the dissimilarity between the stimulus and the preferred feature.

3. Possible examples of a white noise stimulus in different sensory systems:

- Auditory system: playing audio white noise that includes random amplitudes of a wide range of audible frequencies to determine the preferred frequency of the neuron.

- Visual system: (a) An array of pixels in which the gray level of each pixel is randomly and independently chosen. (b) Choosing a random brightness using a flashlight with a broad spectrum of degrees of brightness in random sequences to determine the neuron's preferred light intensity.

4. In reverse correlation, using Gaussian white noise as the stimulus gives a distribution of zero mean and constant variance of stimulus intensity over the time step given. With the different strengths of stimuli overlapping with each other, the adaptation effects are minimized as it is modeled to be constant during the white noise stimulation.

5. A linear process only provides information on the degree of similarity between the stimulus and the STA. Firstly, the firing rates can take negative values from the linear approximation. Secondly, the firing rates can increase indefinitely as the input of stimulus intensity increases due to the continuity feature of the linear model. Thus, the non-linearity function is needed to eliminate these negative values and categorizing firing rates into super-threshold or sub-threshold through rectification because biologically, neurons will saturate as they have a maximum firing rate.

12.6.2 Coding Questions

12.7 Chapter 7: Decoding

12.8 Chapter 8: Neural Networks