Day07_CheckIn_Answers

September 6, 2022

1 Answers to the coding check ins

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[ ]:
```
#this is the specific directory where the data we want to use is stored
loaddir = '../data/'

#this is the directory where we want to store the data we finish analyzing
savedir = '../output/'
```

1.0.1 2.1 Coding check in answer

[ ]:
```
def get_pop_growth(K, r, MaxTime):
    x = np.linspace(0,MaxTime,20) # array from 0 to 10 with 20 equally spaced␣
 ↪values
    y = K/(1 + r*np.exp(-x)) # array of y values

    return(y)

K = 20
r = 10
MaxTime = 20

# using our log_growth function to generate a time series
time = np.linspace(0,MaxTime,20) # array from 0 to 10 with 20 equally spaced␣
 ↪values
log_growth = get_pop_growth(K = K, r = r, MaxTime = MaxTime) # array of y values

print(time)
print(log_growth)
```

[ ]:
```
# set up our basic plot
fig, ax = plt.subplots()
ax.plot(time, log_growth,  'orange', linewidth = 5)
```

```
# set up our axes and title
ax.set_title("Exploring logistic growth", fontsize = 24) # adds a title
ax.set_xlabel("Time", fontsize = 14) # adds x label
ax.set_ylabel("Population size", fontsize = 14) # adds y label

ax.tick_params(axis = "both", labelsize = 14) # increases font size for both␣
 ↪axes

plt.show()
```

1.0.2 4.1 Coding check in answer

[ ]:
```
x = np.linspace(0,20,60)
y1 = np.sin(x)
y2 = np.cos(x)

print(x)
print(y1)
print(y2)
```

[ ]:
```
fig, ax = plt.subplots()
ax.plot(x,y1, 'red', linewidth = 3, label = "sine")
ax.plot(x,y2, 'blue', linewidth = 3, label = "cosine")

ax.set_xlabel("Hours in the day", fontsize = 15)
ax.set_ylabel("Hunger", fontsize = 15)

ax.tick_params(axis = "both", labelsize = 15) # increases font size for both␣
 ↪axes
ax.legend()

plt.show()
```

1.0.3 7.1 Coding check in answer

[ ]:
```
x = np.linspace(0,20,60)
y1 = np.sin(x)
y2 = np.cos(x)

fig, ax = plt.subplots(nrows = 1, ncols = 3, figsize = (10,3))

ax[0].plot(x,y1, 'red', linewidth = 3, label = "sine") # plotting sin curve on␣
 ↪first plot
```

```
ax[1].plot(x,y2, 'blue', linewidth = 3, label = "cosine") # plotting cos curve␣
 ↪on second plot

ax[2].plot(x,y1, 'red', linewidth = 3, label = "sine") # sin curve on third plot
ax[2].plot(x,y2, 'blue', linewidth = 3, label = "cosine") # cos curve on third␣
 ↪plot


ax[0].set_xlabel('Sin plot', fontsize = 15)
ax[1].set_xlabel('Cos plot', fontsize = 15)
ax[2].set_xlabel('Sin and cos plot', fontsize = 15)


ax[0].tick_params(axis = "both", labelsize = 14) # increases font size for both␣
 ↪axes
ax[1].tick_params(axis = "both", labelsize = 14) # increases font size for both␣
 ↪axes
ax[2].tick_params(axis = "both", labelsize = 15) # increases font size for both␣
 ↪axes

# bonus

ax[0].legend(loc = "upper right")
ax[1].legend(loc = 'lower right')
ax[2].legend(loc = 'center')

plt.tight_layout()

plt.savefig(savedir+'sin_and_cos.png')
```

1.0.4 10.1 Coding check in answer

[ ]:
```
july_data = np.loadtxt(loaddir+'city_temps_july.csv', delimiter=',',skiprows =␣
 ↪1)

seattle = july_data[:,5]
chicago = july_data[:,2]
```

[ ]:
```
num_bins = 8 # setting the number of bins for our histogram

fig, ax = plt.subplots()
ax.hist(seattle, num_bins, density = True, color = "orange", label = "Seattle",␣
 ↪alpha = 0.5) # alpha is transparency
ax.hist(chicago, num_bins, density = True, color = "green", label = "Chicago",␣
 ↪alpha = 0.5)
```

```
ax.set_xlabel('Temperature (F)')
ax.set_ylabel('Density')
ax.legend()

plt.show()
```

1.0.5 11.4 Coding challenge answer

[ ]:
```
loaddir = '../data/' #Make sure the paths end in '/'
filenames = ['juno_june21.txt','austin_june21.txt','seattle_june21.txt',␣
 ↪'philadelphia_june21.txt']

juno_data = np.loadtxt(loaddir+filenames[0], delimiter = '\t') # \t means tab
austin_data = np.loadtxt(loaddir+filenames[1], delimiter = '\t')
seattle_data = np.loadtxt(loaddir+filenames[2], delimiter = '\t')
philadelphia_data = np.loadtxt(loaddir+filenames[3], delimiter = '\t')


weather_dat = np.array([juno_data,seattle_data,philadelphia_data,austin_data])
```

[ ]:
```
high_temps = [weather_dat[0,:,0],weather_dat[3,:,0]] # juno and austin highs

mini_states = ["Juno",'Austin']

fig, ax1 = plt.subplots()

# rectangular box plot
bplot1 = ax1.boxplot(high_temps,
                     vert=True,  # vertical box alignment
                     patch_artist=True,  # fill with color
                     labels=mini_states)  # will be used to label x-ticks

colors = ['seagreen', 'aqua']
for patch, color in zip(bplot1['boxes'], colors):
    patch.set_facecolor(color)

plt.show()
```

1.0.6 14.1 Homework equations

[ ]:
```
def plot_SIR(Init_S, Init_I, Init_R, beta, gamma, MaxTime):

    S = [0]*MaxTime # initialize a vector that is 200 elements long of zeros
    I = [0]*MaxTime
    R = [0]*MaxTime

    S[0] = Init_S # setting the first value to the initial conditions
    I[0] = Init_I
    R[0] = Init_R

    for i in range(1,MaxTime,1):
        S[i] = S[i-1] - beta*S[i-1]*I[i-1] # susceptible equation
        I[i] = I[i-1] + beta*S[i-1]*I[i-1] - gamma*I[i-1] # infected equation
        R[i] = R[i-1] + gamma*I[i-1]

    fig, ax = plt.subplots()
    ax.plot(S, label = "Susceptible")
    ax.plot(I, label = "Infected")
    ax.plot(R, label = "Recovered")

    ax.set_xlabel("Time", fontsize = 15)
    ax.set_ylabel("Proportion of population", fontsize = 15)
    ax.legend()

    plt.show()
```

[ ]:
```
params_set1 = {"Init_S": 0.99, "Init_I": 0.01, "Init_R":0, "beta":0.8, "gamma":
 ↪0.1, "MaxTime":100}
plot_SIR(**params_set1)
```

1.1 15 Bonus Homework Assignment

[ ]:
```
np.random.seed(2)

N = 1000

x = np.random.lognormal(mean = 1, sigma = 0.5, size = N)
y = np.random.normal(loc = 1, scale = 5, size = N)
colors = np.random.rand(N)

fig, ax = plt.subplots(nrows=2,ncols=2)

ax[0,0].scatter(x,y, c = colors, alpha = 0.8)
```

```
ax[0,1].scatter(y,x, c = colors, alpha = 0.8)

ax[1,0].hist(x, 30, color = "blue", density=True)
ax[1,1].hist(y, 30, color = "pink", density=True)


ax[0,0].set_xlabel('Lognormal')
ax[0,0].set_ylabel('Normal')

ax[0,1].set_xlabel('Normal')
ax[0,1].set_ylabel('Lognormal')

ax[1,0].set_xlabel('Lognormal')
ax[1,1].set_xlabel('Normal')

plt.tight_layout()
```

[ ]: