Final

Due Dec 15, 2021 at 3:30pm **Points** 240 **Questions** 14
Available Dec 15, 2021 at 8:30am - Dec 15, 2021 at 3:30pm about 7 hours
Time Limit None

Instructions

You have up to **3:30PM Wednesday December 15, 2021** to work on this **240 points** final exam.

It is highly suggested to answer **at least two points per minute**.

You are limited to use:

- **printed, computer files, or online materials listed on the syllabus**
- **your own notes written on paper, computer files, or cloud drive**
- **this course space on the ELMS/Canvas**
- **any Python3 IDE on your computer or vSmith**

No searches nor other materials are permitted!

We will **copy your answer to each question from ELMS then paste** into Python3 IDE for grading, so make sure your:

- **indentations are correct and working**
- **import all and only libraries used in your answer**
- **initialize all and only variables used in your answer**

Clarification questions in **VMH 1212** between **1:30-3:20PM Wednesday December 15, 2021**.

This quiz was locked Dec 15, 2021 at 3:30pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	415 minutes	235 out of 240

Correct answers are hidden.

Score for this quiz: **235** out of 240
Submitted Dec 15, 2021 at 3:30pm
This attempt took 415 minutes.

Question 1

0 / 0 pts

Honor Pledge

"I pledge on my honor that I have not given or received any unauthorized assistance on this examination."

Please type your **full name** and **UID** here:

Your Answer:

Liangrui Lu

missing UID

Question 2

9 / 10 pts

Implement nested if statements (without multi-way) to output the letter grade given an input decimal **score** in the grading scheme below.

Letter Grade	Range of Scores
A+	At or above 99 up to 100
A	At or above 94 and below 99
A-	At or above 90 and below 94
B+	At or above 87 and below 90
B	At or above 84 and below 87
B-	At or above 80 and below 84
F	All other inputs

-1 should not compare two decimal numbers using '>=' or '<='

Question 3

9 / 10 pts

Implement multi-way (no nested) if statement to output the letter grade given an input decimal **score** in the grading scheme below.

Letter Grade	Range of Scores
A+	At or above 99 up to 100
A	At or above 94 and below 99
A-	At or above 90 and below 94
B+	At or above 87 and below 90
B	At or above 84 and below 87
B-	At or above 80 and below 84
F	All other inputs

Your Answer:

```
grade = float(input("Input grade: "))
if (grade > 100) | (grade < 80):
    print("F")
else:
    if grade >= 99:
        print("A+")
    else:
        if grade >= 94:
            print("A")
        else:
            if grade >= 90:
                print("A-")
            else:
                if grade >= 87:
                    print("B+")
                else:
                    if grade >= 84:
                        print("B")
                    else:
                        print("B-")
```

```
grade = float(input("Input grade: "))
if (grade > 100) | (grade < 80):
    print("F")
elif grade >= 99:
    print("A+")
elif grade >= 94:
    print("A")
elif grade >= 90:
    print("A-")
elif grade >= 87:
    print("B+")
elif grade >= 84:
    print("B")
else:
    print("B-")
```

-1 should not compare two decimal numbers using '>=' or '<='

Question 4

10 / 10 pts

Implement a for-loop statement to add up all positive odd numbers smaller than or equal to **threshold** (which is a positive integer variable) using the incremental **number** (which is an integer counter variable) and store the sum in **total** (which is the accumulator integer variable).

Your Answer:

```
threshold = int(input("Input threshold: "))
total = 0
for number in range(1, threshold+1, 2):
    total += number
print(total)
```

Question 5

10 / 10 pts

Implement a while-loop statement to add up all positive odd numbers smaller than or equal to **threshold** (which is a positive integer variable) using the incremental **number** (which is an integer counter variable) and store the sum in **total** (which is the accumulator integer variable).

Your Answer:

```
threshold = int(input("Input threshold: "))
total = 0
number = 1
while number <= threshold:
    total += number
    number += 2
print(total)
```

Question 6

20 / 20 pts

Display countdown number pattern:

- Initialize a boolean variable to False to control the while loop below.
- Implement a while loop to accept an integer number between 1 and 9 from user.
- If user did not enter an integer, prompt an error message and ask user to try again.
- If the integer was not between 1 and 9, prompt an error message and ask user to try again.
- Display the pattern of countdown numbers as follows:

Please enter an integer number between 1 and 9? abc

```
abc is not an integer!
```

Please enter an integer number between 1 and 9? 1.23

```
1.23 is not an integer!
```

Please enter an integer number between 1 and 9? 0

```
0 is not between 1 and 9!!
```

Please enter an integer number between 1 and 9? 1

```
1
```

Please enter an integer number between 1 and 9? 3

```
3 2 1
  2 1
    1
```

Your Answer:

```
control = False
while not control:
    input_string = input("Please enter an integer number between
1 and 9? ")
    if not str.isnumeric(input_string):
        print("%s is not an integer!" % input_string)
    else:
        input_num = int(input_string)
        if (input_num > 9) or (input_num < 1):
            print("%s is not between 1 and 9!!" % input_string)
        else:
            for row in range(input_num):
                for col in range(row):
                    print(' ', end=' ')
                for col in range(input_num-row, 1, -1):
                    print(col, end=' ')
                print(1)
            control = True
```

Question 7

10 / 10 pts

Execute random number generator to randomly scramble positive integers 1 to 12, then write one integer per line into a text file, named **Final_YourLastName_YourFirstName.txt**.

Your Answer:

```
import random
outfile = open("Final_Lu_Liangrui.txt", 'w')
num_list = list(range(1, 13))
for trial in range(12):
    outfile.write("%d\n" % num_list.pop(random.randint(1, 12-tria
l)-1))
outfile.close()
```

Question 8

10 / 10 pts

Implement a function **average(required, optional)** to return the average of two numbers:

- The first argument is required, but the second argument is optional with **default 0**.
- Write a doc-string to be displayed for the **help(average)** function.

Your Answer:

```
def average(required, optional=0):
    """This function is used to calculate the average of two numb
ers:
       The first argument is required, but the second argument is
optional with default 0."""
    return (required+optional) / 2
```

Question 9

10 / 10 pts

Implement a recursive function **sum_up_to(threshold)** to return the sum of all positive integers up to the threshold.

- The function returns 0, if the threshold value is not a positive integer.
- Write a doc-string to be displayed for the **help(sum_up_to)** function.

Your Answer:

```
def sum_up_to(threshold):
    """
    A recursive function to return the sum of all positive intege
rs up to the threshold.
    The function returns 0, if the threshold value is not a posit
ive integer.
    """
    if (threshold <= 0) | (not threshold.is_integer()):
        return 0
    return int(threshold) + sum_up_to(threshold - 1)
```

Question 10

13 / 14 pts

Open NBA_Team_Salary.xlsx (https://umd.instructure.com/courses/1315367/files/63763326/download?download_frd=1) and read the **Summary** worksheet into a pandas data frame using the **Rk** column as index.

- Display the tabular data in the data frame.
- Report number of teams.
- Report the average team salary of every year.

Your Answer:

```
import pandas as pd

df = pd.read_excel('NBA_Team_Salary.xlsx', index_col='Rk')
# pd.set_option('display.max_columns', 0)
# pd.set_option('display.max_rows', 0)
print(df)
print()
print("The number of teams is: %d" % df['Team'].value_counts(drop
na=True).sum())

# print(df.columns)
for col in range(1, len(df.columns)):
    # print(df.columns[col])
    print(df.columns[col], df[df.columns[col]].mean())
```

-1 should not output more than two decimal places

Question 11

24 / 24 pts

Read [BostonHousingData.csv](https://umd.instructure.com/courses/1315367/files/63763315/download?download_frd=1) **(https://umd.instructure.com/courses/1315367/files/63763315/download?download_frd=1)** into a pandas data frame.

- Apply 2-means, 3-means and 4-means clustering on **MEDV** against **RM**.
- Display three subplots in one figure: from left to right, 2-means, 3-means and 4-means clustering results.

Write at least five lines of comments.

Your Answer:

```
import pandas as pd
from scipy.cluster.vq import *
from pylab import *
from matplotlib.pyplot import plot, show  # plot different color
for each cluster by its index
df = pd.read_csv("BostonHousingData.csv")
data = np.array(df[["MEDV", "RM"]].values.tolist())
data = vstack(data)
# print(data)
plt.figure(figsize=(15, 4))
# Begin 2-means clustering on the left
plt.subplot(131)
centroids, _ = kmeans(data, 2)
# assign each sample to a cluster
index, _ = vq(data, centroids)  # plot different color for each c
luster by its index
# Show two clusters in red or blue dots of size 4.
plot(data[index == 0, 0], data[index == 0, 1], 'or', markersize=4
)
plot(data[index == 1, 0], data[index == 1, 1], 'ob', markersize=4
)
# Show two centroids in green squares of size 8.
plot(centroids[:, 0], centroids[:, 1], 'sg', markersize=8)
# Label axes and figure title.
plt.xlabel("MEDV", fontsize=14)
plt.ylabel("RM", fontsize=14)
plt.title("2-means clustering", fontsize=20)
# Begin 3-means clustering on the middle
plt.subplot(132)
centroids, _ = kmeans(data, 3)
# assign each sample to a cluster
index, _ = vq(data, centroids)
# Show three clusters in red, blue or magenta dots of size 4.
plot(data[index == 0, 0], data[index == 0, 1], 'or', markersize=4
)
plot(data[index == 1, 0], data[index == 1, 1], 'ob', markersize=4
)
plot(data[index == 2, 0], data[index == 2, 1], 'om', markersize=4
)
# Show three centroids in green squares of size 8.
plot(centroids[:, 0], centroids[:, 1], 'sg', markersize=8)
# Label axes and figure title.
```

```
plt.xlabel("MEDV", fontsize=14)
plt.ylabel("RM", fontsize=14)
plt.title("3-means clustering", fontsize=20)
# Begin 4-means clustering on the right
plt.subplot(133)
centroids, _ = kmeans(data, 4)
# assign each sample to a cluster
index, _ = vq(data, centroids)
# Show four clusters in red, blue, magenta or yellow dots of size
4.
plot(data[index == 0, 0], data[index == 0, 1], 'or', markersize=4
)
plot(data[index == 1, 0], data[index == 1, 1], 'ob', markersize=4
)
plot(data[index == 2, 0], data[index == 2, 1], 'om', markersize=4
)
plot(data[index == 3, 0], data[index == 3, 1], 'oy', markersize=4
)
# Show four centroids in green squares of size 8.
plot(centroids[:, 0], centroids[:, 1], 'sg', markersize=8)
# Label axes and figure title.
plt.xlabel("MEDV", fontsize=14)
plt.ylabel("RM", fontsize=14)
plt.title("4-means clustering", fontsize=20)
show()
```

Question 12

32 / 32 pts

A small factory makes three types of testing kits in any combination.

- All three different types provide the same quality of test results.
- Each type requires time (in minutes/unit) on each of two machines in the following manner:

	Type x	**Type y**	**Type z**
Machine 1	2	5	6
Machine 2	4	6	8

- Machine 1 operates exactly 405 minutes per day.
- Machine 2 operates exactly 570 minutes per day.
- The factory is required to make exactly 100 testing kits per day.

How many each type of testing kits can this factory make per day?

Visualize the production data in a bar plot.

Write at least five lines of comments.

Your Answer:

```
from numpy import array
from numpy.linalg import solve
import matplotlib.pyplot as plt

A = array([[2,5,6],  # Machine 1 operates exactly 405 minutes per
day.
           [4,6,8],  # Machine 2 operates exactly 570 minutes per
day.
           [1,1,1]])  # The factory is required to make exactly 1
00 testing kits per day.
Y = array([[405], [570], [100]])
print(solve(A,Y))
# Get y-axis from solver solution
solution = solve(A,Y).transpose()[0]
# print(solution)
# Give x-axis as type name
type = ['Type x','Type y','Type z']
plt.bar(type,solution)
plt.xlabel("Type", fontsize=14)
plt.ylabel("Production amount", fontsize=14)
plt.title("Production data for each type of testing kits", fontsi
ze=20)
plt.show()
```

Question 13

38 / 40 pts

A real estate agent reported twelve recent apartment sales as follows.

Price of Apartment in $1000	Distance from Campus in Miles
55	1.5
51	3
60	1.7
75	1
55.5	3.1
49	1.6
65	2.3
61.5	2
55	4
45	5
75	0.6
65	2

Solve a linear regression model or line fit to estimate price of apartment given distance from campus.

- What is the intercept of price in $1000 given the distance in Miles from campus?
- What are the slope and the corresponding standard error?
- What are the correlation coefficient R-value and the two-sided *p*-value of this model?

Visualize the sales data as dots and the model estimation as a line in one plot.

Write at least five lines of comments.

Your Answer:

```
# Apply scipy.stats.linregress to perform linear regression on Ex
perience given Age.
from scipy.stats import *
import matplotlib.pyplot as plt
import numpy as np
# model linear regression
y = np.array([55,51,60,75,55.5,49,65,61.5,55,45,75,65])
x = np.array([1.5,3,1.7,1,3.1,1.6,2.3,2,4,5,0.6,2])
# print(linregress(x, y))
slope, intercept, r_value, p_value, slope_std_error = linregress(
x, y)
# estimate y
y_modeled = x * slope + intercept
# Print the estimation equation.
print("The intercept of price in $1000 given the distance in Mile
s from campus is: %f"%intercept)
print("The slope is: %f, and the corresponding standard error is:
%f"%(slope,slope_std_error))
print("The correlation coefficient R-value is: %f, and the two-si
ded p-value of this model is: %f"%(r_value, p_value))
# plot true and estimated y's
# Show the data in blue dots of size 2.
plt.plot(x, y, 'ob', markersize=2)
# Show the estimation equation in red line of width 4.
plt.plot(x, y_modeled, '-r', linewidth=4)
# Label axes and figure title.
plt.ylabel("Price of Apartment in $1000", fontsize =10)
plt.xlabel("Distance from Campus in Miles", fontsize =10)
plt.title("Linear Regression to estimate price of apartment \ngiv
en distance from campus.", fontsize =12)
plt.show()
```

-2 decimal precision should be trimmed in text outputs

Question 14

40 / 40 pts

A small factory makes two types of cloth face masks, a single-layer and a double-layer:

- Each single-layer requires two minutes on the cutting machine and two minutes on the assembly machine.
- Each double-layer requires two-and-half minutes on the cutting machine and five minutes in the assembly machine.
- The cutting machine can operate ten hours each day.
- The assembly machine can operate twelve hours each day.
- The factory makes profit of $2.50 on each single-layer and $4 on each double-layer.

How many each type of cloth face masks can this factory make to maximize the daily profit?

Solve and visualize this optimization problem in one plot.

Write at least five lines of comments.

Your Answer:

```
# model generation
# decision variables: x, y, x for single-layer and y for double-l
ayer mask:
# objective function was: max 2.5x + 4y
# objective function now: min -2.5x - 5y
c = [-2.5, -4]
# cutting machine time constraint: 2x + 2.5y <= 600
# cutting machine time constraint: 2x + 5y <= 720
a = [[2, 2.5], [2, 5]]
b = [600, 720]

import matplotlib.pyplot as plt
from scipy.optimize import linprog

# linear programming
res = linprog(c, a, b)
print(res)
# x = 2, y = 1


# create figure and display unit grids
fig, ax = plt.subplots()
plt.axis([0, 400, 0, 250])
# plot model
plt.plot([0, 300], [240, 0], '-b', linewidth=1)
plt.plot([0, 360], [144, 0], '-b', linewidth=1)
plt.fill_between([0, res.x[0]], [144, res.x[1]],
                 color='y')
plt.fill_between([res.x[0], 300], [res.x[1], 0],
                 color='y')
# plot solution
plt.annotate("max solution (%.0f, %.0f)" %
             (res.x[0], res.x[1]),
             xy=(res.x[0], res.x[1]),
             xytext=(res.x[0], res.x[1] + 20),
             arrowprops={"arrowstyle": "simple", "color": 'r'})
plt.grid(color='g', linestyle=':')
plt.xlabel("Single-layer masks production", fontsize=14)
plt.ylabel("Double-layer masks production", fontsize=14)
plt.title("Linear optimal for mask production", fontsize=20)
plt.show()
```

Quiz Score: **235** out of 240