CSCI 1070
Taming Big Data
Midterm
Thursday, October 17, 2019

First Name	KEY
Last Name	
SLUNetID	

Be sure to write your name on the top of each page.

	Grade
WRITING YOUR NAME ON EVERY PAGE	/ 3
Question 1 - Python Expressions	/ 10
Question 2 - Control Structures	/ 15
Question 3 - List Comprehension	/ 5
Question 4 - Interpreting Histograms	/ 10
Question 5 - Counties	/ 22
Question 6 - Cereal	/ 25
Total	/ 90

1. (10 points) Python Expressions

For each of the Python expressions below, write the output when the expression is evaluated. If an error occurs, write Error. Here is an example:

Example Expression: make_array(1, 2, 3, 4, 5) == 3

Example Answer: array([False, False, True, False, False])

(a) (2 pt) type('1.5')

str

(b) (2 pt) float(5.0) + str(5.0)

Error

(c) (2 pt) make_array(1 , 5) * np.arange(1, 10, 5)

array([1 , 30])

(d) (2 pt) make_array(1, 2, 3, 4, 5) + 5

array([6, 7, 8, 9, 10])

(e) (2 pt) 'I love CSC1 ' + 1070

Error

2. (15 points) Control Structures and List Comprehension

For each of the blocks of Python code below, write what will be printed when the code is run. Be sure that you make clear the type that is being printed out. For example:

- a string should be in single or double quotes -- 'a string'
- a list should be in square brackets -- [ ]
- an array should show up as array([ ])

Example Code:

```
even_number_list = []
for number in np.arange(1,10):
    if number % 2 == 0:
        even_number_list.append(number)
print(even_number_list)
```

Example Answer: [2, 4, 6, 8]

(a) (5 pt)

```
value_to_track = 1
for multiplier in np.arange(3):
    value_to_track *= multiplier
print(value_to_track)
```

$\emptyset$

(b) (5 pt)

```
colors = ['red', 'orange', 'yellow', 'green', 'blue',
'indigo', 'violet']
for color in colors:
    if len(color) >= 6:
        break
    print(color)
```

red

(c) (5 pt)

```
even_number_list = [2, 4, 6]
my_array = make_array(1, 10, 100)
for y in np.arange(10):
    if y in even_number_list:
        continue
    else:
        my_array = my_array + y

print(my_array)
```

4	10	100	
1	10	100	+0
2	11	101	+1
2	11	101	+2
5	14	104	+3
[illegible]	[illegible]	[illegible]	+4
10	19	109	+5
			+6
17	26	116	+7
			+8
			+9

array([34, 43, 133])

3. (5 pts) List Comprehension

(a) (2 pts) Which of the following blocks of code would produce the exact same output as the following list comprehension? Bubble in the correct answer.

```
list_1 = [i**2 for i in np.arange(1,10,5) if i % 2 == 0]
```

○

```
list_1 = []
for i in np.arange(1, 10, 5):
    if i % 2 == 0:
        list_1.append(i)
```

○

```
list_1 = []
for i in np.arange(1, 10, 5):
    if i % 2:
        list_1.append(i)
```

○

```
list_1 = make_array()
for i in np.arange(1, 10, 5):
    if i % 2 == 0:
        list_1.append(i**2)
```

[✓]

```
list_1 = []
for i in np.arange(1, 10, 5):
    if i % 2 == 0:
        list_1.append(i**2)
```

○ None of the above.

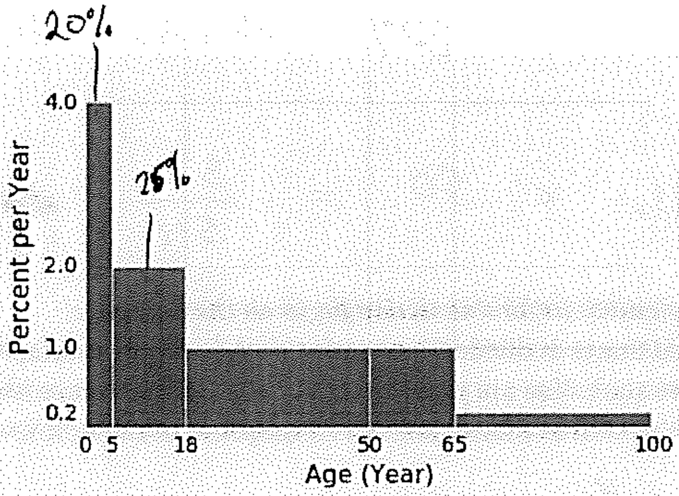
(b) (3 pt) What will list_1 evaluate to after the code in part (a) is executed?

[36]

4. (10 points) Interpreting Histograms

In 2018, the Centers for Disease Control released data about the effectiveness of the flu vaccine. The histogram below shows the distribution of the ages of people who got the vaccine and did ***not*** get the flu.

Bins are inclusive on the lower end and exclusive on the upper end. The numbers on the vertical axis are the heights of the bars. For example, the height of the bar over the 65 - 100 bin is 0.2. Units are provided on the axis labels.



(a) (2 pt) True or false: The percent of people in the 0 - 5 bin is two times the percent of people in the 5 - 18 bin.

True [✓ False]

(b) (2 pt) Explain your answer to part (a).

The percent in the 0-5 bin is 5×4=20 %

The percent in the 5-18 bin is 13×2 = 26%

(c) (2 pt) Define school-age children to be those who are at least 5 years old, but less than 18 years old. Fill in the blank with a number or arithmetic expression:

13×2 % of people are school-age children

(d) (4 pt) Define adults as people who are at least 18 years old. What is the proportion ***of adults*** who are less than 50 years old? Show your work.

Adults = $(50-18) \times 1 + (65-50) \times 1 + (100-65) \times .2$

$= 32 + 15 + 7$

$= 54\%$ of the total are adults

32% of the total are 18-50

proportion of adults < 50 $= \frac{32}{54}$

5. (22 points) Counties ***[questions for this section begin on the next page]***

Every state in the United States is divided into counties that do not overlap with each other, and together cover the entire state. You are provided a .csv file called *counties.csv* that contains one row for each county in the United States. Below is a snapshot of the first few rows of this csv:

```
county, state, area, pop, avg_income
Los_Angeles, CA, 4060, 8863164, 20786
Cook, IL, 946, 5105067, 21729
Harris, TX, 1729, 2818199, 19517
San_Diego, CA, 4205, 2498016, 19588
Orange, CA, 790, 2410556, 24400
Kings, NY, 71, 2300664, 16803
...
```

This csv contains five fields:

- **county**: the name of the county
- **state:** the two letter code for the state in which the county is located
- **pop:** the total number of people that reside in the county
- **avg_income**: the average household income in the state

forgot area

(a) (2 pts) Complete the following line of Python code to read this csv file into a Table object called `counties`. (Assume we have done all necessary imports for you already and that you don't need to write any import statements.)

counties = Table. read _ table ('counties.csv')

(b) (5 pts) Now that you have your `counties` table, we want to know which county has the largest population. Write the Python code to find the most populous county in the space below:

counties. sort ('pop', descending = True). take(0)

(c) (5 pts) Write the Python code that assigns the state that has the $\overset{\text{county w/}}{\vee}$ highest average income to the variable `highest_income_state`.

sorted = Counties. sort ('avg - income', descending = True)

sorted. take (0) . select ('state')

~~(d) (5 pts) Write the Python code that will create a new table that contains only the most populous county ***in each state***. The rows of this new table should be **county, state, pop,** and **avg_income** and there should be 50 rows (one for each state). You may name this table whatever you'd like.~~

~~Counties. group ("state~~

BONUS
PTS

(e) (5 pts) Write the Python code that computes the average income ***per person*** and adds this information to the counties table.

```
income = Counties. column ('avg-income')

population= counties. column ('pop')


Canties = counties. with_column (
    'pop-per-person' , income / population)
```

6. (25 pts) **Cereal**
The following table, named `cereal`, contains one row with nutritional information for each of a set of cereal types:

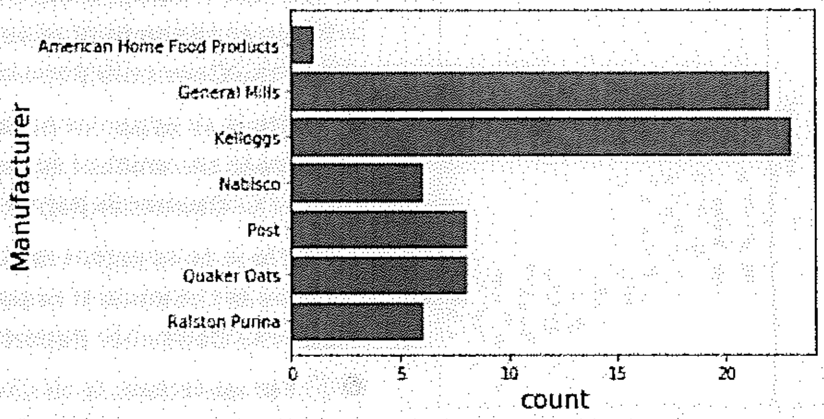
Cereal Name	Manufacturer	Calories	Protein	Fat	Dietary Fiber	Carbs	Sugars	Display Shelf	Rating
100%_Bran	Nabisco	70	4	1	10	5	6	3	68.403
100%_Natural_Bran	Quaker Oats	120	3	5	2	8	8	3	33.9837
All-Bran	Kelloggs	70	4	1	9	7	5	3	59.4255

... (71 rows omitted)

The table contains the columns:

- **Cereal Name:** a string, the name of the cereal
- **Manufacturer:** a string, the company that manufactures the cereal
- **Calories:** an int, calories per serving
- **Protein:** an int, grams of protein per serving
- **Fat:** an int, grams of fat per serving
- **Dietary Fiber:** an int, grams of fiber per serving
- **Sugars:** an int, grams of sugar per serving
- **Display Shelf:** an int, what shelf the cereal is displayed on in the supermarket (1, 2 or 3)
- **Rating:** a float, giving a rating of the cereal on a scale of 1 - 100 from Consumer Reports

(a) (1 pt) Which of the following lines of code will produce the figure below?

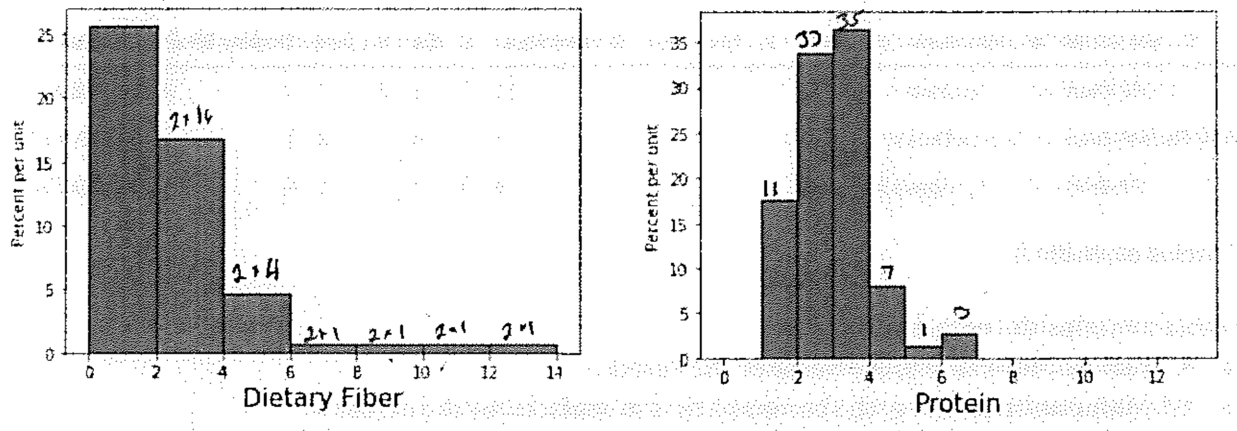


○ cereal.hist('Manufacturer')
○ cereal.barh('Manufacturer')
● cereal.group('Manufacturer').barh('Manufacturer')
○ cereal.group('Manufacturer').plot('Manufacturer')

32 8 2222

$\frac{32 \; 16}{48}$

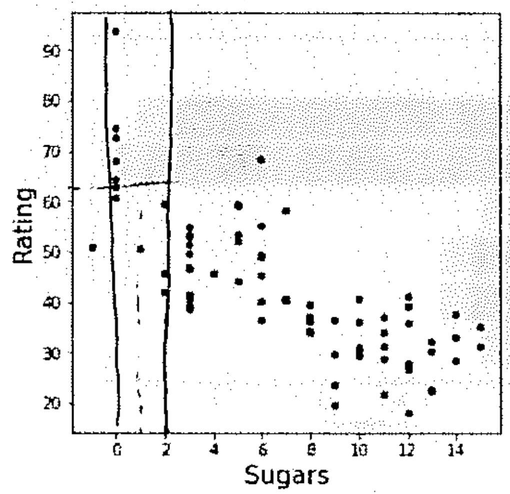
(b) (3 pt) All the cereals are represented in each of the two histograms below. Which conclusion or conclusions are justified by the histograms? Remember that all data are integers. **Bubble in all correct answers.**



- [✓ About half of the cereals have at least 2 grams of fiber per serving.]
- ○ Most cereals have more grams of protein than grams of fiber per serving.
- ○ Most cereals have more grams of fiber than grams of protein per serving.
- [✓ The number of cereals with at least 4 grams of fiber per serving is about the same as the number of cereals with at least 4 grams of protein per serving.]
- ○ Cereals with more fiber per serving tend to have more protein per serving.

OK if not circled ~16 vs ~11

(c) (3 pt) Which conclusion of conclusions are justified by the scatter plot below? **Bubble in all correct answers.**



- ○ There is an apparent positive correlation between the amount of sugar per serving and the Consumer Reports rating.
- [✓ There is an apparent negative association between the amount of sugar per serving and the Consumer Reports rating.]
- ○ There is no apparent negative association between the amount of sugar per serving and the Consumer Reports rating.
- [✓ For most cereals, their Consumer Reports ratings would likely be reduced if the manufacturer added more sugar.]
- ○ For most cereals, their Consumer Reports ratings would likely be increased if the manufacturer added more sugar.
- ○ For most cereals, their Consumer Reports ratings would likely be unchanged if the manufacturer added more sugar.

(d) (4 pt) Suppose that we want to predict the Consumer Report rating for a new cereal based on its amount of sugar. To do this, we first want to average the ratings of all cereals in `cereal` who sugar content per serving differs from that of the new cereal by no more than 1 gram. Fill in the blanks below to define a function called `predict_rating` that takes in an amount of sugar in grams and returns the prediction of the Consumer Reports rating.

```
def predict_rating(sugar):

    # comparison_group are the cereals within 1 gram of the new cereals
    # sugar levels

    comparison_group = Cereal.where('Sugars', are.between(sugar-1, sugar+1))

    return comparison_group.column('Rating').mean()
```

(e) (4 pt) Based on Parts (a) - (d), in approximately which range will the predicted rating be for a cereal with 1 gram of sugar per serving?

- ○ between 30 and 35
- ○ between 40 and 45
- ○ between 50 and 55
- [✓ between 60 and 65]

(f) (4 pt) Write the Python code below that will add a new column to the table `cereal` containing the prediction from the `predict_rating` function for the sugars value in each row. The column should be called 'Predicted Rating'.

```
predicted = cereal.apply(predict_rating, "Sugars")

Cereal = cereal.with_column(
        'Predicted Rating',
        predicted

)
```

(g) (2 pt) Next, we want to see how well our predictions do on average. Write a Python expression that evaluates the average absolute error for all of our predictions. The absolute error is the absolute value of the difference between the predicted rating and the actual rating.

abs(counties.column('Rating') -

counties.column('Predicted Rating'))

(h) (4 pt) Write the Python Code that would generate the following table. The table contains the average number of calories per serving in the different cereal brands grouped by Manufacturer **and** Display Shelf.

Display Shelf	American Home Food Products	General Mills	Kelloggs	Nabisco	Post	Quaker Oats	Ralston Purina
1	0	106.667	107.5	86.6667	105	100	102.5
2	100	111.429	111.429	95	110	113.333	0
3	0	114.444	107.5	70	110	80	127.5

Cereal. pivot ('Manufacturer', 'Display Shelf',

values = 'Calories',

collect = np. mean)

don't remember the term $\longrightarrow$