Data 6, Summer 2024 **Ginsborg and Vargas-Navarro**

Wrap-Up & Review

Discussion 5 *Summer 2024*

As this is our last discussion today we will focus on reflecting on the semester and preparing for the final.

1 Conceptual Review

1.1 Dictionaries

According to the Composing Programs Textbook "A dictionary contains key-value pairs, where both the keys and values are objects. The purpose of a dictionary is to provide an abstraction for storing and retrieving values that are indexed not by consecutive integers, but by descriptive keys." We make dictionaries using the below syntax.

```
new_dict = {key1 : value1, key2: value2}
```

We can access elements in dictionaries using bracket notation.
`new_dict[key1]` will give us `value1`.
We can also update the values assigned to keys or add new keys to the dictionary using this syntax `new_dict[new_key] = new_value.`
To access the keys and values of a dictionary we can use **dictName.keys()** and **dictName.values()** respectively.

Dictionaries Practice

1. Create a new dictionary **staff** with the keys "tutor1" and "tutor2" and values "Elizabeth" and "Kenneth"

2. Update the dictionary to have "SuMin" assigned to the key "GSI"

3. Assign the key "allStaff" to the previous 3 keys values added together

1.2 Iteration

Python's built-in `for` can be used to create `for` loops. We can use them to do two main things: first, they allow us to iterate through arrays, manipulating each element as we wish. Alternatively, we can use `for` loops to repeat lines of code many times. Examples of how `for` loops can be used are below.

```
for item in some_array:
  print(item)

for i in np.arange(1000):
  print('Hello')
```

Python's built-in `while` can be used to create `while` loops. While loops allow us to repeat a body of code as long as a condition is true. They can be useful when we don't know how many times we'll want to repeat our code.

```
i = 0
while i < 10:
  print(i)
  i +=1
```

Iteration Practice

1. Write code to sum the first 30 odd numbers using both a for loop and while loop

2. Write code to find factorial of a number n. You should make a function. Feel free to use for or while loops.

3. Write code to sum together all the numbers in a dictionary **numbers** whose values are all differnt ints.

1.3 Sampling and Randomness

The following is taken with some adjustment from a Data 8 tutoring worksheet.
Population vs. Sample
In data science, we often want to be able to make a general statement about a population of individuals. Unfortunately, time and resource constraints generally prevent scientists from having access to data about entire populations of individuals. For that reason, we examine parts of the population called samples. For example, you may be interested in knowing the percentage of all eligible voters who are planning on voting for Candidate A. Since asking everyone in the U.S. who they plan to vote for is clearly infeasible, we will have to take a sample. Note our sample is selected from a list of possible things to be sampled called our `Sampling Frame` which may or may not align well the Population.

Sometimes, we will want to sample from a pre-existing table. To do so, we can use the following table method: `tbl.sample(sample_size, with_replacement, weights)`

In other cases, we may have an array we need to sample from. In this case, we can use the following function: `np.random.choice(array, sample_size)`
We can also choose random numbers using `np.random.randint(start, stop, size)`

Sampling Practice

1. Generate 5 random numbers between 1 and 10 inclusive using any approach. Discuss why you chose your method.

2. Randomly simulate 10 flips of a coin biased towards heads such that the probability of heads is 70% and the probability to tails is 30%. You will have to make an array or table first.

1.4 Comparisons

The comparison operators $<, <=, >, >=$, `==`, and `!=` allow us to easily compare most data types.

operator	action
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equality
!=	not equal

Some examples are that $3 > 2$ is True and that "Berkeley" > "Stanford" is False since it compares alphabetically for strings. These comparison operators also work element wise on arrays. Since all comparison operators return booleans we can combine them using the operators `and, or, not` which have the following behavior.

operator	action	example
and	True only if left and right booleans are True	$5 > 4$ and $3 >$ True
or	True if any boolean is True	True or $1/0$ is True
not	negates the boolean to it's right	not $5 > 4$ is False

Comparisons Practice

1. Evaluate the follow expressions:

   (a) `(5%2 == 1) and (2 in np.arange(10))`

   (b) `True or False and False or True`

   (c) "a+" > "B"

2. Given an array of numbers called `numbers` count all the numbers divisible by 3. You should not use a loop.

2 Past Data8 Exam Practice

`Data 8 FA23 Midterm Q4 Modified`

Every day, Coco solves a random New York Times crossword and keeps track of how long it takes to solve each one. She stores this data in a table called crosswords, with each row representing 1 puzzle. A sample of the table is provided below:

Date	Published	Solved	Minutes	Seconds
10-05	Thursday	Friday	12	32
10-06	Friday	Tuesday	28	51
10-08	Sunday	Sunday	5	23
10-09	Monday	Saturday	10	5

Table 1: Coco's Crosswords

The table has the following columns:

- *Date*: (string) the month and day that the crossword was published
- *Published*: (string) the day of the week the crossword was published
- *Solved: (string)* the day of the week the crossword was solved
- *Minutes*: (int) the number of minutes the crossword took to solve, rounded down
- *Seconds*: (int) the number of seconds past mins that the crossword took to solve

(a) (3.0 pt) Coco writes a function that takes in a row from the crosswords table and converts the Minutes and Seconds columns into only seconds. For example, the first row shown in the sample above would return 732. She writes the following partially completed code:

```
def duration_in_seconds(row):
    return ____________
```

Fill in the blank to complete the function. Reminder: There are 60 seconds in 1 minute

(b) (5.0 points) Coco wants to use the duration_in_seconds function to generate the duration for every puzzle in the crosswords table and add it to her table as a new column name Total Duration. She writes the following partially completed code:

```
durations = crosswords.____(a)_____
crosswords = ____(b)____
```

i. (2.0 pt) Fill in blank (a).

ii. (3.0 pt) Fill in blank (b).

(c) (4.0 pt) After running the code in the previous question, Coco next wants to create a table where each unique publishing day of the week gets its own row, each unique solving day of the week gets its own column, and the values inside the table correspond to the average time in seconds it took to solve the puzzles for each combination of published and solved day of the week. She writes the following partially completed code:

```
crosswords._________
```

Fill in the blank. Recall: The crosswords table has columns Date, Published, Solved, Minutes, Seconds, and Total Duration.

(d) (2.0 pt) Coco's roommate, Aryna, would like to see how Total Duration varies between puzzles published on Saturdays and puzzles published on Mondays. Which of the following lines of code could help with this?
Select all that apply.

- ☐ crosswords.scatter('Published', 'Total Duration')
- ☐ crosswords.hist('Total Duration')
- ☐ crosswords.pivot('Total Duration', 'Published')
- ☐ crosswords.hist('Total Duration', group='Published')
- ☐ crosswords.scatter('Published', 'Total Duration', group='Published')

(e) (6.0 points) Aryna has been secretly reviewing Coco's puzzles and scoring them for correctness. She's created a separate table called puzzles that contains every puzzle that Coco has solved. It has the following columns:

- *Edition*: (string) the month and day that the crossword was published (for example, '10-05' for October 5)
- *Score*: (float) the percent of letters that Coco answered correctly

Aryna suspects that Coco doesn't do that well on her crosswords when solving them on Saturdays because she's busy playing sports all day. She writes the following partially completed code, which assigns min_score to the minimum score that Coco ever received when solving a puzzle on a Saturday.

```
min_score = np.min(crosswords._____(a)_____.where(_____(b)_____)._____(c)_____)
```

Recall: The crosswords table has columns *Date*, *Published, Solved, Minutes, Seconds*, and *Total Duration*.

i. (2.0 pt) Fill in blank (a).

ii. (2.0 pt) Fill in blank (b).

iii. (2.0 pt) Fill in blank (c).

2. (22.0 points) Shorts

UBA, a media company based in New York, wants to start integrating Youtube videos into its morning show.

To evaluate some propsects, Cory, the CEO, puts together a table called `videos` that contains a random sample of Youtube videos published in the last year. The first few rows are shown here:

Name	ID	Views	#Shorts	Date
INSANE strawberry trick! #Shorts	UC6D1L2vxEAg	329461822	True	05-08
Adele - Easy On Me (Official Video)	UComP_epzeKz	272980726	False	10-14
BTS () 'Permission to Dance'	UC3IZKseVp	480881920	False	07-09
BTS () 'Butter'	UC3IZKseVp	746499500	False	05-20

The table has the following columns:

- *Name*: (string) the **video**'s name
- *ID*: (string) the **channel**'s ID in Youtube's database
- *Views*: (int) the number of times the video has been watched
- *#Shorts*: (bool) whether the video is a short
- *Date*: (string) the month and day the video was published

(a) (3.0 pt) Complete this Python expression, which evaluates to the name of the most watched video.

```
videos._________.item(0)
```

(b) (4.0 points)

Complete this Python expression, which evaluates to an array with two items (in any order): the average number of views for #Shorts and the average number of views for non-#Shorts.

```
videos._________(_________)._________('Views average')
          (a)       (b)         (c)
```

i. (1.0 pt) Which of these could fill in blank (a)?

◯ `sort`

◯ `where`

◯ `group`

ii. (2.0 pt) Fill in blank (b).

iii. (1.0 pt) Which of these could fill in blank (c)?

◯ `select`

◯ `take`

◯ `column`

(c) (3.0 pt) Complete this Python expression, which visualizes the distribution of the number of views for videos with names that include a `#` character (i.e. a hashtag).

```
videos._________
```

Recall: The `videos` table has columns *Name*, *ID*, *Views*, *#Shorts*, and *Date*.

(d) (4.0 points)

Complete this Python expression, which visualizes the distribution of the counts of #Shorts vs. non-#Shorts, including only videos that have more than 10,000 views.

```
videos.where(_________)._________._________
               (a)         (b)         (c)
```

Recall: The `videos` table has columns *Name*, *ID*, *Views*, *#Shorts*, and *Date*.

i. (2.0 pt) Fill in blank (a).

ii. (1.0 pt) Which of these could fill in blank (b)?

- ◯ `column('#Shorts')`
- ◯ `column('Views')`
- ◯ `pivot('#Shorts', 'Views')`
- ◯ `pivot('Views', '#Shorts')`
- ◯ `group('Views')`
- ◯ `group('#Shorts')`

iii. (1.0 pt) Which of these could fill in blank (c)?

- ◯ `hist('count')`
- ◯ `hist('#Shorts')`
- ◯ `hist('Views')`
- ◯ `barh('#Shorts', 'count')`
- ◯ `barh('Views', 'count')`

(e) (3.0 pt) Complete this Python expression, which evaluates to a table with one row for each unique date and three columns: the date, as well as the total number of views of #Shorts and non-#Shorts videos on that date. Any column labels are acceptable.

```
videos._______________
```

Recall: The `videos` table has columns *Name*, *ID*, *Views*, *#Shorts*, and *Date*.

(f) (5.0 points)

Cory's colleague Bradley notices that the `videos` table doesn't contain the names of the channels.

Bradley creates an additional table called `channels` that contains all Youtube channels and has two columns:

- *Identifier*: (string) the channel's ID in Youtube's database
- *Channel*: (string) the channel's name

Bradley suspects that channels with exactly 4 characters in their name such as Vevo could be good channels to partner with since even their lowest performing videos have many views.

Complete her code, which assigns `min_views` to the minimum number of views received by a video posted to a channel with a 4-character name.

Hint: Calling `len` on a string returns the number of characters in the string.

```
channels = channels.with_column('Name Length', _________)
                                                  (a)

min_views = np.min(videos._________.where('Name Length', 4)._________)
                              (b)                                (c)
```

Recall: The `videos` table has columns *Name*, *ID*, *Views*, *#Shorts*, and *Date*.

i. (2.0 pt) Fill in blank (a).

ii. (2.0 pt) Fill in blank (b).

iii. (1.0 pt) Fill in blank (c).