Midterm Exam (Spring 2024)

Instructions

Please write your name and university-issued email address below in the space provided.

Name: ____________________________________

Email Address: ____________________________________

You will have 75 minutes to answer the questions contained herein. You may submit the exam at any time within that period. Once you begin the exam, you may not leave the room until you submit it.

You are expected to not consult with any other source of information during the exam period. No phones! Please take a moment before the exam starts to clear your desk of all materials except for your cheat sheet. There should be no talking for any reason during the exam period. If you have a question about the exam material, raise your hand and wait for an opportunity to ask an instructor for clarification.

FYI: as some students begin to finish the exam and leave the room, the professor may ask some of the remaining students to relocate desks, to achieve a more even space distribution.

When you are ready, you may begin. Good luck!

Evaluation

The weight of each question is detailed below. Partial credit may be awarded, and there is no penalty for guessing. The total weight adds up to more than 100%, so if you get stuck on one question, you are encouraged to move on and pick up as many points as possible.

Part I - Short Answer and Free Response	
Question	**Weight**
1 (Software vs Hardware)	8%
2 (Why Python?)	8%
3 (Datatypes)	12%
4 (Expressions and Operators)	12%
5 (Code Tracing)	12%
	52%

Part II - Crunch the YouTube Data	
Question	**Weight**
6-A (Video Info)	12%
6-B (Video Keywords / Tags)	12%
7 (Streams)	24%
	48%
Optional Bonus	
8 (Counters and Accumulators)	+8%
9 (Unit Testing)	+4%

Computer-based Systems and Software [16%]

1. What is the role, or purpose, of **software** within the context of a computer-based information system? What is the role, or purpose, of **hardware**? And what is the relationship between software and hardware in such a system?

2. When we write application software, we have a choice of which language to program in. What are three distinct reasons **why Python** is a good choice? In other words, what are some positive aspects or benefits of programming in the Python language (in general, and/or as opposed to other languages)?

    a.

    b.

    c.

Python Datatypes [12%]

3. For each of the following example Python objects, what is its **datatype**? FYI: for nested objects, specify only the datatype of the parent (outermost) object. Provide your answers in the column on the right.

    a. `“Spring Break, here we come :-D”` *Datatype*: ________________

    b. `False` *Datatype*: ________________

    c. `None` *Datatype*: ________________

    d. `3.14` *Datatype*: ________________

    e. `["vanilla", "chocolate", "strawberry"]` *Datatype*: ________________

    f. `"1010"` *Datatype*: ________________

    g. `{“make”: “Tesla”, “model”: “Cybertruck”}` *Datatype*: ________________

    h. `[{“students”: 24}, {“students”: 55}]` *Datatype*: ________________

    i. `100.0` *Datatype*: ________________

    j. `{“letters”: [1, 2, 5, 3, 99]}` *Datatype*: ________________

    k. `55` *Datatype*: ________________

    l. `"yolo"` *Datatype*: ________________

Python Expressions and Operators [12%]

4. For each of the following example Python expressions, specify its **resulting value**. In other words, if we were to evaluate or “print” the expression, **what would we see**? Provide your answers in the column on the right. If the result is textual in nature, make sure to use the proper case (i.e. we will be grading in a case sensitive way).

    a. `False or True` *Result*: ________________

    b. `True and False` *Result*: ________________

    c. `5 * 5 == 20` *Result*: ________________

    d. `5 + 5 != 10` *Result*: ________________

    e. `9 == “9”` *Result*: ________________

    f. `str(9) == “9”` *Result*: ________________

    g. `5 not in [1, 2, 5]` *Result*: ________________

    h. `“yo” in “Yolo”` *Result*: ________________

    i. `False or False or True` *Result*: ________________

    j. `False or None or "" or "yolo" or True` *Result*: ________________

    k. `"Coding is fun".lower().replace("fun", "super fun")`

        Result: ________________

    l. `"Coding is fun".title().replace("fun", "SupeR fuN")`

        Result: ________________

Code Tracing [12%]

5. Reference the code provided below. Assume this code exists in a single code cell, so lines above will affect the lines below. There are a number of **print statements**. For each print statement, **what result will we see**? Provide your answers in the column on the right.

```
    x = 5
    x = x * 2
    x += 5
a.  print(x)                                  Result: ____________


    x = str(x)
    x = x * 3
b.  print(x)                                  Result: ____________


    if x == 15:
        message = "YEP"
    elif x >= 90:
        message = "LARGE"
    elif isinstance(x, str):
        message = "INTERESTING"
    else:
        message = "NOPE"

c.  print(message)                            Result: ____________
```

Crunch the YouTube Data, Part I - Video [24%]

6. Given the Python variable called `video` provided on page 11 of the exam booklet, **write Python code** which references that variable to perform each of the following tasks:

    a. Display / "print" a human friendly representation of information about the video, including the following details:
        - Video Title, Video Author, Date Published, Video Description
        - Video Length, in minutes, rounded up to the nearest whole number using a ceiling operation (i.e. "7 minutes")
            - NOTE: video length is originally provided in seconds
            - HINT: use the `ceil()` function provided by the `math` module
        - Number of Views, formatted using a thousands separator (i.e. "758,357")
            - HINT: use the `format_number()` function provided on page 12. Assume this function is already in memory, and available for use.

    Desired Output:

    ```
    TITLE: What is Machine Learning?
    AUTHOR: codebasics
    PUBLISHED: 2018-06-30
    DESCRIPTION: What is Machine Learning? This is an introduction to ...
    LENGTH: 7 minutes
    VIEWS: 758,357
    ```

    Solution Code:

b. Display / "print" the number of keywords or tags for this `video`. Sort them in alphabetical order, then loop through and print each on its own line, formatted as uppercase.

Desired Output:

```
------------
TAGS: 16
------------
... MACHINE LEARNING BASICS
... MACHINE LEARNING BASICS FOR BEGINNERS
... MACHINE LEARNING PLAYLIST
... MACHINE LEARNING PYTHON
... MACHINE LEARNING PYTHON TUTORIAL
... MACHINE LEARNING TUTORIAL
... MACHINE LEARNING TUTORIAL FOR BEGINNERS
... MACHINE LEARNING TUTORIAL PYTHON
... MACHINE LEARNING TUTORIALS
... MACHINE LEARNING USING PYTHON
... MACHINE LEARNING WITH PYTHON
... ML FOR BEGINNERS
... ML TUTORIAL
... ML TUTORIAL FOR BEGINNERS
... PYTHON MACHINE LEARNING
... PYTHON MACHINE LEARNING TUTORIAL
```

Solution Code:

Crunch the YouTube Data, Part II - Streams [24%]

7. Given the Python variable called `streams` provided on page 12 of the exam booklet, **write Python code** which references that variable to perform each of the following tasks:

   a. Display / "print" a count of the number of total streams.
   b. Filter the streams to keep only the ones that have a "mime_type" value of "video/mp4". We will call these the "video streams". Display / "print" a count of the number of video streams.
   c. Sort the video streams by "bitrate" in ascending order, then loop through them, and display / "print" info about each on a new line, including the video's mime type, resolution, and bitrate. Format the bitrate with thousands separators, using the `format_number()` function provided on page 12. Assume this function is already in memory, and available for use.

   Desired Output:

```
----------------
STREAMS: 20
VIDEO STREAMS: 8
----------------
... MIME TYPE: video/mp4 | RESOLUTION: 144p | BITRATE: 32,229
... MIME TYPE: video/mp4 | RESOLUTION: 240p | BITRATE: 53,424
... MIME TYPE: video/mp4 | RESOLUTION: 360p | BITRATE: 104,381
... MIME TYPE: video/mp4 | RESOLUTION: 360p | BITRATE: 151,927
... MIME TYPE: video/mp4 | RESOLUTION: 480p | BITRATE: 174,821
... MIME TYPE: video/mp4 | RESOLUTION: 720p | BITRATE: 180,985
... MIME TYPE: video/mp4 | RESOLUTION: 720p | BITRATE: 285,948
... MIME TYPE: video/mp4 | RESOLUTION: 1080p | BITRATE: 410,313
```

   Solution Code:

Counters and Accumulators [8% BONUS]

8. **Write Python code** that will provide a solution to the prompt below..

   Prompt:

   You received a box of 100 chocolates for Valentine's Day! You immediately eat three chocolates. On each day afterwards, you continue to eat three chocolates per day, and you recruit one new friend per day to help you eat the chocolates. Once you recruit a friend, they eat three chocolates that day, and they continue to eat three chocolates per day as well. You and your growing number of friends continue to each eat three chocolates per day, until there are no more chocolates left in the box. Display / "print" the **number of days** it took to finish all the chocolates (i.e. 7), and the total / final **number of friends** (excluding yourself) that helped you (i.e. 7).

   Solution Code:

Custom Functions and Unit Testing [4% BONUS]

9. Reference the custom function called `format_number()` provided on page 12 of the exam booklet. Write one or more unit tests to codify our expectations about what the function is supposed to do. When executed, these unit tests should determine whether the function is working properly. No need to test invalid inputs in this case.
   - HINT: use the `assert` keyword to perform your test(s)

   Solution Code:

This page has been left intentionally blank. Feel free to make notes on it. Its contents will not be evaluated.

*The variable on this page is to be used in conjunction with **Question 6-A and 6-B**. Feel free to detach this page and make notes on it. Its contents will not be evaluated. If you do detach it, write your name on it and remember to return it along with the rest of your exam booklet!*

```
video = {
    'age_restricted': False,
    'author': 'codebasics',
    'channel_id': 'UCh9nVJoWXmFb7sLApWGcLPQ',
    'channel_url': 'https://www.youtube.com/channel/xyz,
    'description': "What is Machine Learning? This is an introduction to
machine learning to begin the python machine learning tutorial series. This
video describes what is machine learning, deep learning, machine learning
application in real life. In next tutorial we will start writing python code
to solve a simple problem using machine learning...",
    'keywords': [
        'machine learning tutorial',
        'machine learning basics',
        'machine learning tutorial for beginners',
        'machine learning python',
        'python machine learning tutorial',
        'machine learning tutorial python',
        'machine learning with python',
        'ml tutorial',
        'machine learning using python',
        'machine learning tutorials',
        'machine learning python tutorial',
        'ml for beginners',
        'ml tutorial for beginners',
        'machine learning basics for beginners',
        'machine learning playlist',
        'python machine learning'
    ],
    'length_seconds': 410,
    'publish_date': '2018-06-30',
    'rating': None,
    'thumbnail_url': 'https://i.ytimg.com/vi/abc/def.jpg',
    'title': 'What is Machine Learning?',
    'video_id': 'gmvvaobm7eQ',
    'views': 758357,
    'watch_url': 'https://youtube.com/watch?v=gmvvaobm7eQ'
}
```

*The function on this page is to be used in conjunction with **Questions 6, 7, and 9**. The variable is to be used in conjunction with **Question 9**. Feel free to detach this page and make notes on it. Its contents will not be evaluated. If you do detach it, write your name on it and remember to return it along with the rest of your exam booklet!*

```
streams = [
    {'bitrate': 33429, 'filesize_mb': 1.636, 'mime_type': 'video/3gpp', 'resolution': '144p', 'stream_type': 'video'},
    {'bitrate': 151927, 'filesize_mb': 7.43, 'mime_type': 'video/mp4', 'resolution': '360p', 'stream_type': 'video'},
    {'bitrate': 180985, 'filesize_mb': 8.852, 'mime_type': 'video/mp4', 'resolution': '720p', 'stream_type': 'video'},
    {'bitrate': 410313, 'filesize_mb': 3.798, 'mime_type': 'video/mp4', 'resolution': '1080p', 'stream_type': 'video'},
    {'bitrate': 1634401,'filesize_mb': 10.765, 'mime_type': 'video/webm', 'resolution': '1080p', 'stream_type': 'video'},
    {'bitrate': 285948, 'filesize_mb': 2.547, 'mime_type': 'video/mp4', 'resolution': '720p', 'stream_type': 'video'},
    {'bitrate': 1014210, 'filesize_mb': 6.677, 'mime_type': 'video/webm', 'resolution': '720p', 'stream_type': 'video'},
    {'bitrate': 174821, 'filesize_mb': 1.673, 'mime_type': 'video/mp4', 'resolution': '480p', 'stream_type': 'video'},
    {'bitrate': 599956, 'filesize_mb': 3.864, 'mime_type': 'video/webm', 'resolution': '480p', 'stream_type': 'video'},
    {'bitrate': 104381, 'filesize_mb': 1.126, 'mime_type': 'video/mp4', 'resolution': '360p', 'stream_type': 'video',},
    {'bitrate': 387448, 'filesize_mb': 2.606, 'mime_type': 'video/webm', 'resolution': '360p', 'stream_type': 'video'},
    {'bitrate': 53424, 'filesize_mb': 0.713, 'mime_type': 'video/mp4', 'resolution': '240p', 'stream_type': 'video'},
    {'bitrate': 218215, 'filesize_mb': 1.505, 'mime_type': 'video/webm', 'resolution': '240p', 'stream_type': 'video'},
    {'bitrate': 32229, 'filesize_mb': 0.528, 'mime_type': 'video/mp4', 'resolution': '144p', 'stream_type': 'video'},
    {'bitrate': 90783, 'filesize_mb': 0.991, 'mime_type': 'video/webm', 'resolution': '144p', 'stream_type': 'video'},
    {'bitrate': 50094, 'filesize_mb': 2.389, 'mime_type': 'audio/mp4', 'resolution': None, 'stream_type': 'audio'},
    {'bitrate': 130641, 'filesize_mb': 6.337, 'mime_type': 'audio/mp4', 'resolution': None, 'stream_type': 'audio'},
    {'bitrate': 54047, 'filesize_mb': 2.497, 'mime_type': 'audio/webm', 'resolution': None, 'stream_type': 'audio'},
    {'bitrate': 75866, 'filesize_mb': 3.333, 'mime_type': 'audio/webm', 'resolution': None, 'stream_type': 'audio'},
    {'bitrate': 137460, 'filesize_mb': 6.143, 'mime_type': 'audio/webm', 'resolution': None, 'stream_type': 'audio'}
]
```

```
def format_number(large_number):
    """Formats a large number with thousands separators, for printing and logging.
        Param large_number (int) like 1000000000
        Returns (str) like '1,000,000,000'
        Example: format_number(1000000000)
    """
    return f"{large_number:,.0f}"
```