Discussion 3

DSC 20, Spring 2023

Midterm 1 Practice

Agenda

we're going to run through select questions from past exams

Some questions have been subtly modified

playlist of walkthroughs can be found here

Notes

- The more recent the practice exam, the more accurate to your exam it will be
- Do the practice exams by writing (for practice!)
- Skip topics you don't know

Midterm Logistics

- Make sure to bring pen/pencil/eraser and your **student ID**
- Exam takes place Friday, April 28 in MANDE B-210 (in lecture)

Midterm Grading

- Multiple choice questions are generally "all-or-nothing"
- Exceptions exist (ex. if we have an "other" option and it's fill in the blank)
- Free response questions are always awarded partial credit
- Even if you're lost on the question, write down the important steps to the code

Question 1

What should be the values for x,y,z if I want to print out "summer!"

Note: Don't forget the order of evaluation! (NAO)

```
In [ ]: if x and y and z:
            print('fall')
        elif not x and not y:
            print('winter...')
        elif not z or x:
            print('spring?')
        else:
            print('summer!')
```

Question 1 Solution

```
In [3]:
x = False
y = True
z = True

if x and y and z:
    print('fall')
elif not x and not y:
    print('winter...')
elif not z or x:
    print('spring?')
else:
    print('summer!')
```

```
summer!
```

Question 2

Given 2 lists, remove the elements from the first list if they're present in the second one.

Your solution must be 1 line and use filter

note: filter has not been covered quite yet at the time of this discussion, don't worry if this question is confusing - consider it more as a preview of friday's lecture (It is a covered topic on midterm 1).

In [2]:

```
def up_join(lst1, lst2):
    """
    Function that removes elements from lst1
    if they're present in lst2.

    Solution must be 1 line and utilize filter

    Args:
        lst1 (list): list of values to be considered
        lst2 (list): list of values to be considered
    Returns:
        a filtered version of lst1

    >>> grades = ['A', 'B', 'C', 'D', 'F']
    >>> grade_filter = ['D', 'F']
    >>> up_join(grades, grade_filter)
    ['A', 'B', 'C']
    """
    # Write your implementation here
    return
```

Question 2 Solution

```
In [5]: def up_join(lst1, lst2):
            """
            Function that removes elements from lst1
            if they're present in lst2.

            Solution must be 1 line and utilize filter

            Args:
                lst1 (list): list of values to be considered
                lst2 (list): list of values to be considered
            Returns:
                a filtered version of lst1

            >>> grades = ['A', 'B', 'C', 'D', 'F']
            >>> grade_filter = ['D', 'F']
            >>> up_join(grades, grade_filter)
            ['A', 'B', 'C']
            """
            return list(filter(lambda x: x not in lst2, lst1))
```

```
In [6]: up_join(['A', 'B', 'C', 'D', 'F'], ['D', 'F'])
```

```
Out[6]: ['A', 'B', 'C']
```

Question 3

Given a dictionary consisting of strings for keys and values and another string, create a new dictionary where the key is the new string and the value is the list of keys who had it as their value. **Write assert statements to check input**.

In [16]:
```
def flip_dict(owners, pet):
    """
    Function that creates a new dictionary consisting
    of 'pet' as its key and the keys of 'owners'
    that had 'pet' as its value.

    Args:
        owners (dictionary): dictionary of strings to be considered.
        pet (string): string to be considered.
    Returns:
        a dictionary with key 'pet' and value as a list
        consisting of matched keys in 'owners'
    Throws:
        AssertionError: if pet is not a string
        AssertionError: if owners is not a dictionary
        AssertionError: if the keys and values of owners are
        not all strings

    >>> sample = {'ben':'cats', 'charisse':'dogs', 'nikki':'cats'}
    >>> flip_dict(sample, 'cats')
    {'cats': ['ben', 'nikki']}
    """
```

```python
    # Write your implementation here
    return
```

Question 3 Solution

In [17]:
```python
def flip_dict(owners, pet):
    """
    Function that creates a new dictionary consisting
    of 'pet' as its key and the keys of 'owners'
    that had 'pet' as its value.

    Args:
        owners (dictionary): dictionary of strings to be considered.
        pet (string): string to be considered.
    Returns:
        a dictionary with key 'pet' and value as a list
        consisting of matched keys in 'owners'
    Throws:
        AssertionError: if pet is not a string
        AssertionError: if owners is not a dictionary
        AssertionError: if the keys and values of owners are
        not all strings

    >>> sample = {'ben':'cats', 'charisse':'dogs', 'nikki':'cats'}
    >>> flip_dict(sample, 'cats')
    {'cats': ['ben', 'nikki']}
    """
    assert isinstance(owners, dict)
    assert isinstance(pet, str)
    assert all([isinstance(key, str) for key in list(owners.keys())])
    assert all([isinstance(val, str) for val in list(owners.values())])
    output = {}
    output[pet] = []
    for owner, pet_type in owners.items():
        if pet_type == pet:
            output[pet].append(owner)
    return output
```

Question 3 Output

```
In [18]: flip_dict({'ben':'cats', 'charisse':'dogs', 'nikki':'cats'}, 'cats')

Out[18]: {'cats': ['ben', 'nikki']}

In [21]: flip_dict({'ben':'cats', 'charisse':'dogs', 'nikki':'cats'}, 'dogs')

Out[21]: {'dogs': ['charisse']}

In [20]: flip_dict({'test':1}, 'test')
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent 
call last)
Cell In[20], line 1
----> 1 flip_dict({'test':1}, 'test')

Cell In[17], line 23, in flip_dict(owners, pet)
     21 assert isinstance(pet, str)
     22 assert all([isinstance(key, str) for key in list(owners.
keys())])
---> 23 assert all([isinstance(val, str) for val in list(owners.
values())])
     24 output = {}
     25 output[pet] = []
```

```
AssertionError:
```

Question 4

Given a file containing an expression on each line, write a function that classifies them as 'energetic' if it ends with !, 'confused' if it ends with a ?, and 'neutral' otherwise.

sample file: files/mood.txt

```
hello
omg!
ok?
cool
```

In [31]:

```python
def text_classifier(filepath):
    """
    Function that classifies each word in a text file based
    on the punctuation it ends with:
    if the word ends with '!',  classify as 'energetic'
    if the word ends with '?', classify as 'confused'
    otherwise, classify as 'neutral'

    args:
        filepath (string): filepath of data to be considered
    returns:
        a list of classifications

    >>> text_classifier(files/mood.txt)
    ['neutral', 'energetic', 'confused', 'neutral']
    """
    # Write your implementation here
    return
```

Question 4 Solution

In [3]:
```
def text_classifier(filepath):
    """
    Function that classifies each word in a text file based
    on the punctuation it ends with:
    if the word ends with '!',  classify as 'energetic'
    if the word ends with '?', classify as 'confused'
    otherwise, classify as 'neutral'

    args:
        filepath (string): filepath of data to be considered
    returns:
        a list of classifications

    >>> text_classifier(files/mood.txt)
    ['neutral', 'energetic', 'confused', 'neutral']
    """
    with open(filepath, 'r') as f:
        data = f.readlines()
        # can also write as a for loop, no restriction
        #for this question
        return ['energetic' if x.strip()[-1]=='!' else \
    'confused' if x.strip()[-1]=='?' else 'neutral' for x in data]
```

In [4]:
```
text_classifier('files/mood.txt')
```

Out[4]:
```
['neutral', 'energetic', 'confused', 'neutral']
```

Question 5

Write a function that takes in a list of strings with the pattern of first_name, last_name, first_name,... and strings together first and last names with a single space in between. If the length of the list is odd, insert your own last name.

In [39]:

```python
def combine_names(names):
    """
    Function that concatenates every 2 strings with a space in between.
    If the length of the list is odd, use your own last name as the
    final element.

    Args:
        names (list): list of strings to be considered.
    Returns:
        a list consisting of full names.

    >>> combine_names(['Charisse', 'Hao', 'Nicole', 'Zhang'])
    ['Charisse Hao', 'Nicole Zhang']
    >>> combine_names(['Charisse', 'Hao', 'Ben'])
    ['Charisse Hao','Ben Chen']
    """
    # Write your implementation here
    return
```

Question 5 Solution

```
In [57]: def combine_names(names):
             """
             Function that concatenates every 2 strings with a space in between.
             If the length of the list is odd, use your own last name as the
             final element.

             Args:
                 names (list): list of strings to be considered.
             Returns:
                 a list consisting of full names.

             >>> combine_names(['Charisse', 'Hao', 'Nicole', 'Zhang'])
             ['Charisse Hao', 'Nicole Zhang']
             >>> combine_names(['Charisse', 'Hao', 'Ben'])
             ['Charisse Hao','Ben Chen']
             """
             output = []
             if len(names)%2 == 1:
                 names.append('Chen')
             for i in range(0,len(names), 2):
                 output.append(' '.join(names[i:i+2]))
             return output
```

```
In [55]: combine_names(['Charisse', 'Hao', 'Nicole', 'Zhang'])
```

```
Out[55]: ['Charisse Hao', 'Nicole Zhang']
```

```
In [56]: combine_names(['Charisse', 'Hao', 'Ben'])
```

```
Out[56]: ['Charisse Hao', 'Ben Chen']
```

Question 6

Given a dictionary that has lists as values, Write a function that returns a list that consists of the length of each list in the dictionary

In [62]:

```
def count_values(entries):
    """
    Function that counts the length of each value in a dictionary.
    Assume the values are only of type list.

    Args:
        entries(dictionary): dictionary of lists as
        values to be considered.
    Returns:
        a list where each element is the length of
        the dictionary value.

    >>> count_values({1: [1,2,3], 2:[3,4,5,6]})
    [3,4]
    """
    # Write your implementation here
    return
```

Question 6 Solution

```
In [63]: def count_values(entries):
             """
             Function that counts the length of each value in a dictionary.
             Assume the values are only of type list.

             Args:
                 entries(dictionary): dictionary of lists as
                 values to be considered.
             Returns:
                 a list where each element is the length of
                 the dictionary value.

             >>> count_values({1: [1,2,3], 2:[3,4,5,6]})
             [3,4]
             """
             return [len(entry) for entry in entries.values()]
```

```
In [64]: count_values({1: [1,2,3], 2:[3,4,5,6]})
```

```
Out[64]: [3, 4]
```

Question 7

Write a function that takes two lists of the same length that contains integers and returns true if the first list is strictly greater than the second list. **One line solution**.

```
In [67]: def greater_comparison(lst1, lst2):
             """
             Function that checks whether the integers of lst1
             are strictly greater than lst2.

             Args:
                 lst1 (list): list of integers to be considered
                 lst2 (list): list of integers to be considered
             Returns:
                 True if elements of lst1 are strictly greater
                 than those of lst2, False otherwise.

             >>> greater_comparison([10,20,30], [1,2,3])
             True
             >>> greater_comparison([0,0,4], [1,2,3])
             False
             """
             # Write your implementation here
             return
```

Question 7 Solution

```
In [71]: def greater_comparison(lst1, lst2):
             """
             Function that checks whether the integers of lst1
             are strictly greater than lst2.

             Args:
                 lst1 (list): list of integers to be considered
                 lst2 (list): list of integers to be considered
             Returns:
                 True if elements of lst1 are strictly greater
                 than those of lst2, False otherwise.

             >>> greater_comparison([10,20,30], [1,2,3])
             True
             >>> greater_comparison([0,0,4], [1,2,3])
             False
             """
             return all([lst1[idx] > lst2[idx] for idx in range(len(lst1))])
```

```
In [72]: greater_comparison([10,20,30], [1,2,3])
```

```
Out[72]: True
```

Question 8

Write a function that takes in a matrix and a number. It returns the result of the multiplication. **You may only use list comprehension and the solution must be 1 line**.

In [75]:
```python
def matrix_multiplication(matrix, coefficient):
    """
    Function that multiplies the given matrix by the coefficient.
    Solution must be one line list comprehension.

    Args:
        matrix (list): nested list representing a matrix
        coefficient (integer): int to be considered
    Returns:
        the resulting matrix after every element is
        multiplied by coefficient.

    >>> mtx = [[1,2,3],[4,5,6],[7,8,9]]
    >>> matrix_multiplication(mtx, 3)
    [[3, 6, 9], [12, 15, 18], [21, 24, 27]]
    """
    # Write your implementation here
    return
```

Question 8 Solution

```
In [77]:
def matrix_multiplication(matrix, coefficient):
    """
    Function that multiplies the given matrix by the coefficient.
    Solution must be one line list comprehension.

    Args:
        matrix (list): nested list representing a matrix
        coefficient (integer): int to be considered
    Returns:
        the resulting matrix after every element is
        multiplied by coefficient.

    >>> mtx = [[1,2,3],[4,5,6],[7,8,9]]
    >>> matrix_multiplication(mtx, 3)
    [[3, 6, 9], [12, 15, 18], [21, 24, 27]]
    """
    return [[element*coefficient for element in row] for row in matrix]
```

```
In [78]:
matrix_multiplication([[1,2,3],[4,5,6],[7,8,9]], 3)
```

```
Out[78]:
[[3, 6, 9], [12, 15, 18], [21, 24, 27]]
```

Thanks for Coming!

Next week will be content review for midterm 1 (with practice questions)