W. P. Carey School of Business
Arizona State University

CIS 325: Programming for Business Analytics

Assignment 3 (Individual work)

Submission Instructions

- Submit your Python script according to the guidelines in this document as a .py file with the following naming convention: "A3-[ASURITE ID].py", where you will replace the text "[ASURITE ID]" with your ASURITE ID (= your ASU login ID).

  For example, if your ASURITE ID is "abcd12", then your submission file name would be "A3-abcd12.py".

- On the top (header) of your submission .py file, add your name and email address as shown below:

```
"""
@author: [YOUR FULL NAME]
@email: [YOUR EMAIL ADDRESS]
"""
```

- Add Python comment statements (such as "*# Answer to Question 1*") in your submission .py file to separate your answers between questions.

- Not following the above submission instructions will result in a **0.2 point reduction** from your grade.

Question 1 (1 points)

A problem you are trying to solve has one data structure named dict1 in the starter code below. You have been asked to do the following:

- Use a for loop statement to go through the keys in *descending* order in your dict1 dictionary. For each iteration in your loop statement, print the key-value pairs in your dict1 dictionary. Hint: consider using statements reversed(), sorted() and dict1.keys() in your solution.

- Your script should provide console output as indicated in the sample output below.

Starter code:

```
dict1 = {'e': 2, 'j': 4, 'a': 3, 't': 6, 'q': 1}

### Place your code below this line ###



### Place your code above this line ###
```

Expected output:

```
key: t ; value: 6
key: q ; value: 1
key:  j ; value: 4
key: e ; value: 2
key: a;  value: 3
```

Question 2 (1 points)

When working with data sets, it is useful to be able to categorize, for instance, a list of words by their first letters as a dictionary of lists, as outlined in our textbook (in section 3.1 under the "dict" section, in the subsection named "Default values", pages 63–64). Our textbook covers the dictionary setdefault() method, which is a useful, clean and more efficient method to create default dictionaries from lists. You have been asked to do the following:

- Search for the documentation of the dictionary setdefault() method. Then write a brief description of the dictionary setdefault() method as Python comments. Your description must contain (1) what setdefault() does, (2) syntax of setdefault(), and (3) what are the arguments (indicate required or optional) and what they do.
- Identify from the textbook (in section 3.1 under the "dict" section, in the subsection named "Default values") the relevant setdefault() method that creates a list of words by their first letters as a dict of lists.
- Implement this method in the starter code below. The correct implementation of this technique should result in the sample Python console output.

Starter code:

```
mylist = ['action', 'table', 'tennis', 'apple', 'trap']

### (WRITE YOUR DESCRIPTION OF THE dictionary setdefault() METHOD)

### Place your code below this line ###



### Place your code above this line ###

print('dictA:', dictA)
```

Expected output:

```
dictA: {'a': ['action', 'apple'], 't': ['table', 'tennis', 'trap']}
```

Question 3 (0.5 points)

A problem you are trying to solve has two Python sets A and B as seen in the starter code below.

- Set A: multiples of 2 between 1 and 20.
- Set B: multiples of 5 between 1 and 20.

You have been asked to do the following:

1. Create a new set variable named U that has all items from both sets A and B without any duplicates.
2. Create a new set variable named **I** that has all common items that exist in both sets A and B. That is, items that are both multiples of 2 and 5.
3. Create a new set variable named S that has all elements in either A or B, but not both.

Starter code:

```
A = set(range(2, 21, 2))
B = set(range(5, 21, 5))

### Place your code below this line ###


### Place your code above this line ###

print('U:', U)
print('I:', I)
print('S:', S)
```

Expected output:

```
U: {2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 20}
I: {10, 20}
S: {2, 4, 5, 6, 8, 12, 14, 15, 16, 18}
```

Question 4 (1 points)

A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself (= it can only be divided by 1 and itself).1 The first few prime numbers are {2, 3, 5, 7, 11, …}. The code below checks if the variable num is a prime number or not.

```
num = 11
result = 'Yes'

if num > 1:
    for i in range(2, num):
        # check for divisors
        if num % i == 0:
            result = 'No'
            break
else:
    result = 'No'

print('Is num a prime number?', result)
```

The above code outputs the following to the console:

```
Is num a prime number? Yes
```

Based on the above code, create a Python function named is_prime with a single positional argument named num that determines if num is a prime number or not. Your is_prime function must return the boolean True to the function caller (using the function return statement) if the positional argument num is a prime number. Otherwise, it should return the boolean False to the function caller.

Correctness Checks: If you were to run the following function calls, your output should be as follows:

```
>>> r = is_prime(1)
>>> print(r)
False
>>>  print(type(r))
<class 'bool'>

>>> r = is_prime(5)
>>> print(r)
True

>>> r = is_prime(6)
>>> print(r)
False

>>> a = 'Prime' if is_prime(11) else 'Not prime'
>>> print(a)
Prime
```

Question 5 (1 points)

A problem you are trying to solve has a Python string named lorem that has been split word-by-word using the string **split()** method with an argument sep = ' ' to create a new list named lorem_words, as seen in the starter code below. You have been asked to do the following:

- Search for the documentation of the string split() method. Then write a brief description of the string **split()** method as Python comments. Your description must contain (1) what **split()** does, (2) syntax of **split()**, and (3) what are the arguments (indicate required or optional) and what they do. Lastly, in your own words, explain how the variable lorem_words got its value after running the first two lines of the starter code based on your description of the string **split()** method.
- Create a new dictionary named word_len.
- Use a for loop with a sorted() statement (ascending order) for your lorem_words list.
- Inside your for loop statement, add key-value pairs to your new dictionary named word_len, using each word in the lorem_words list as the dictionary key, and the dictionary value being the length of the word (= number of characters) being added to your dictionary named word_len.
- The last line of your script has a print(word_len) statement that will provide the sample output below.

Starter code:

```
lorem = 'ut enim ad minim veniam quis nostrud exercitation ullamco laboris nisi aliquip'
lorem_words = lorem.split(sep=' ')
print(lorem_words)

### (WRITE YOUR DESCRIPTION OF THE string split() METHOD)

### (EXPLAIN IN YOUR OWN WORDS, HOW lorem_words GOT ITS VALUE)

### Place your code below this line ###


### Place your code above this line ###

print(word_len)
```

Expected output:

```
{'ad': 2, 'aliquip': 7, 'enim': 4, 'exercitation': 12, 'laboris': 7, 'minim': 5,
'nisi': 4, 'nostrud': 7, 'quis': 4, 'ullamco': 7, 'ut': 2, 'veniam': 6}
```

Question 6 (1 points)

Write a function that accepts a string and returns a dictionary with each unique character in the string as keyand its corresponding number of occurrences as value.

- Create a function named get_char_count_dict with one positional argument named txt.
- Inside your new function, you will need to check for the type of variable being passed in your positional argument named txt.
  - If the variable type is a string, you will proceed with the function.
  - If the variable type is not a string, your function will use the return statement to provide a valueof −1 to the function caller and will not proceed any further.
  - HINT: Let's say that you have a Python variable named x. To check if this variable x is a string, youcan either use isinstance(x, str) or type(x) == str. Check out: https://pythonprinciples.com/blog/check-if-var-is-string
- After ensuring that the variable type of your positional argument named txt is a string, your new function will first change any uppercase characters in txt to lowercase using the string lower() method, so that the function is **not** case-sensitive.
- After converting everything in txt to lowercase, create a new empty dictionary that will store your results. Then use a for loop to iterate over all characters in txt as follows:
  - If a character is not a key in your dictionary, create a new key-value entry in your dictionary withthe character as key and 1 as its value.
  - If a character is already a key in your dictionary, increment its existing value by 1.
  - HINT: Let's say you have a Python dictionary named D. You can check if this dictionary D has akey named key1 by the following statement: key1 in D
  - Do **not** count whitespaces (whitespace character in Python is simply ' ').
  - When the for loop finishes, your dictionary should have each character (as key) and its corresponding number of occurrences in txt (as value).
- Use the function return statement to provide the newly constructed dictionary of character occurrencecounts of the input txt to the function caller and your function will not proceed any further.

Correctness Checks: If you were to run the following function calls, your output should be as follows:

```
>>> D = get_char_count_dict('little')
>>> print(D)
{'l': 2, 'i': 1, 't': 2, 'e': 1}

>>> D = get_char_count_dict('LiTtle')
>>> print(D)
{'l': 2, 'i': 1, 't': 2, 'e': 1}

>>> D = get_char_count_dict(127)
>>> print(D)
-1

>>> s1 = 'The only impossible journey is the one you never begin!'
>>> D = get_char_count_dict(s1)
>>> print(D)
{'t': 2, 'h': 2, 'e': 8, 'o': 5, 'n': 5, 'l': 2, 'y': 3, 'i': 4, 'm': 1, 'p': 1, 's': 3,
'b': 2, 'j': 1, 'u': 2, 'r': 2, 'v': 1, 'g': 1, '!': 1}

>>> D = get_char_count_dict(' ')
>>> print(D)
{}
```

Question 7 (1 points)

You have been provided the following conversion formulas between Fahrenheit and Celsius:

- Fahrenheit to Celsius conversion: $\text{Celsius} = (\text{Fahrenheit} - 32) \times \frac{5}{9}$
- Celsius to Fahrenheit conversion: $\text{Fahrenheit} = \left(\text{Celsius} \times \frac{9}{5}\right) + 32$

Starter code:

```
### Place your code below this line ###


# A = {dictionary comprehension} something like this...


### Place your code above this line ###

print(A)
```

Based on the starter code above, you need to do the following:

- Create a Python **dictionary comprehension** statement that converts integer Celsius degrees rangingfrom 0 to 35 (including both 0 and 35) to Fahrenheit.
- You will store your dictionary comprehension statement in a variable named A.
  - The key for your dictionary comprehension would be the range of Celsius degrees from 0 to 35.
  - The values for your dictionary comprehension would be the converted value to Fahrenheit fromyour keys that have the corresponding Celsius value.
- You will need to use the correct formula in this question (as shown above) in your dictionary comprehension.

Ensure your converted values are rounded up to 1 decimal value (you may use the round() function).

- Your dictionary comprehension will need to use one for loop with a range() statement that includes the number 0 through and including the number 35 in increments of 1.
- You will then need to print your dictionary A to the Python console resulting in the sample output below.

- Expected output:

```
{0: 32.0, 1: 33.8, 2: 35.6, 3: 37.4, 4: 39.2, 5: 41.0, 6: 42.8, 7: 44.6, 8: 46.4, 9:
↪ 48.2, 10: 50.0, 11: 51.8, 12: 53.6, 13: 55.4, 14: 57.2, 15: 59.0, 16: 60.8, 17: 62.6,
↪ 18: 64.4, 19: 66.2, 20: 68.0, 21: 69.8, 22: 71.6, 23: 73.4, 24: 75.2, 25: 77.0, 26:
↪ 78.8, 27: 80.6, 28: 82.4, 29: 84.2, 30: 86.0, 31: 87.8, 32: 89.6, 33: 91.4, 34: 93.2,
↪ 35: 95.0}
```

Question 8 (1.5 points)

The following code sorts a list of tuples named tups using the *second* element of each tuple as the sort key. As shown in the code, this is done by setting the argument key of the sorted() function as a lambda function, which takes an input argument named x and returns x[1] (= second element of x).

```
tups = [('English', 88), ('Science', 90), ('Maths', 97), ('History', 82)]
result = sorted(tups, key = lambda x: x[1])
print(result)
```

The above code outputs the following (sorted list of tuples) to the console:

```
[('History', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
```

Now write a function that sorts a list of dictionaries, as follows:

- Create a function named sort_by_selected_key with two positional arguments named dicts and keyname. The function will allow users to choose which key (using the positional argument keyname) to use for sorting the input list of dictionaries given by the positional argument dicts.
- Sort the list of dictionaries dicts by the key keyname in ascending order. Accomplish this by using the sorted() function with its argument key set as a lambda function. This lambda function should takea dictionary as its input argument and return the value for the key keyname from the input dictionary (this will be similar to the lambda function in the above example code with the list of tuples).
- Use the function return statement to provide the newly sorted list of dictionaries to the function caller and your function will not proceed any further.

Correctness Checks: If you were to run the following function calls, your output should be as follows:

```
# example list of dictionaries
employees = [{'name': 'John', 'age': 28, 'years': 2.5},
{'name': 'Lisa', 'age': 24, 'years': 3.1},
{'name': 'Ella', 'age': 31, 'years': 2.9}]
```

```
>>> D = sort_by_selected_key(employees, 'name') # sorting by name
>>> print(D)
[{'name': 'Ella', 'age': 31, 'years': 2. 9},
{'name': 'John', 'age': 28, 'years': 2.5},
{'name': 'Lisa', 'age': 24, 'years': 3.1}]

>>> D = sort_by_selected_key(employees, 'age') # sorting by age
>>> print(D)
[{'name': 'Lisa', 'age': 24, 'years': 3.1},
{'name': 'John', 'age': 28, 'years': 2.5},
{'name': 'Ella', 'age': 31, 'years': 2.9}]

>>> D = sort_by_selected_key(employees, 'years') # sorting by years
>>> print(D)
[{'name': 'John', 'age': 28, 'years': 2.5},
{'name': 'Ella', 'age': 31, 'years': 2.9},
{'name': 'Lisa', 'age': 24, 'years': 3.1}]
```