Homework 5
Strings

Due Date: Monday, October 24th,, 2022 by 11:59:59 PM
Value: 40 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
"""
File:    FILENAME.py
Author:  YOUR NAME
Date:    THE DATE
Section: YOUR DISCUSSION SECTION NUMBER
E-mail:  YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

Creating Your HW5 Directory

```
linux3[1]% cd cmsc201/Homeworks
linux3[2]% mkdir hw5
linux3[3]% cd hw5
linux3[4]% 
```

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 HW5 {files go here}

Problem 1 - Hexadecimal Validation	hex_validation.py
Problem 2 - Hexadecimal Conversion	hex_conversion.py
Problem 3 - Timestamp Difference	timestamp_diff.py

For example you would do:

```
linux1[4]% submit cmsc201 HW5 timestamp_diff.py
hex_conversion.py hex_validation.py
Submitting hex_validation.py...OK
Submitting timestamp_diff.py...OK
Submitting hex_conversion.py...OK
linux1[5]% 
```

Coding Standards

Coding standards can be found here.

We will be looking for:

1. At least one inline comment per program explaining something about your code.
2. Constants above your function definitions, outside of the "if __name__ == '__main__':" block.
    a. A magic value is a string which is outside of a print or input statement, but is used to check a variable, so for instance:
        i. `print(first_creature_name, 'has died in the fight. ')` does not involve magic values.
        ii. However, `if my_string == 'EXIT':` exit is a magic value since it's being used to compare against variables within your code, so it should be:
```
EXIT_STRING = 'EXIT'
...
if my_string == EXIT_STRING:
```
    b. A number is a magic value when it is not 0, 1, and if it is not 2 being used to test parity (even/odd).
    c. A number is magic if it is a position in an array, like my_array[23], where we know that at the 23rd position, there is some special data. Instead it should be
    USERNAME_INDEX = 23
    my_array[USERNAME_INDEX]
    d. Constants in mathematical formulas can either be made into official constants or kept in a formula.
3. Previously checked coding standards involving:
    a. snake_case_variable_names
    b. CAPITAL_SNAKE_CASE_CONSTANT_NAMES
    c. Use of whitespace (2 before and after a function, 1 for readability.)

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings, lists, dicts.
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- Casting int(x), str(x), float(x), (technically bool(x))
- For loops, both *for i* and *for each* type.
- While loops
  - sentinel values, boolean flags to terminate while loops
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
  - 2d-lists if you want them/need them my_2d[i][j]
  - Append, remove
  - **list slicing**
- String operations, concatenation +, +=, split(), strip(), join(), upper(), lower(), isupper(), islower()
  - **string slicing**
- Print, with string formatting, with end= or sep=:
  - '{}'.format(var), '%d' % some_int, f-strings
  - Really the point is that we don't care how you format strings in Python
  - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.
- Using the functions provided to you in the starter code.
- Using import with libraries and specific functions **as allowed** by the project/homework.

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- break, continue
- **Dictionaries**
    - creation using dict(), or {}, copying using dict(other_dict)
    - .get(value, not_found_value) method
    - accessing, inserting elements, removing elements.
    - This won't be forbidden much longer
- **Creating your own Classes**
    - Neither will this, be patient.
- methods outside those permitted within allowed types
    - for instance str.endswith
    - list.index, list.count, etc.
- Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield
- The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).
- the following built in functions/keywords: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max, min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, round, set, setattr, slice, sorted, staticmethod, sum, super, type, vars, zip
- exit() or quit()
- If something is not on the allowed list, not on this list, then it is probably forbidden.
- The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.

Problem 1 - Validating String Text

This might look familiar to some of you! Your program, hex_validation.py will accept a string from a user. The program, **requiring** the use of **string splicing and a loop**, will look at each individual character within the string and determine whether the character entered by the user is a valid Hexadecimal value. You'll notice from here that Hexadecimal values can individually be both numbers (0-9) and characters (A-F).

Sample Output

```
linux5[201]% python3 hex_validation.py
Please enter a hexadecimal value: AF12
Index [0]: A : is a valid hex value
Index [1]: F : is a valid hex value
Index [2]: 1 : is a valid hex value
Index [3]: 2 : is a valid hex value

linux5[202]% python3 hex_validation.py
Please enter a hexadecimal value: AF 12G
Index [0]: A : is a valid hex value
Index [1]: F : is a valid hex value
Index [2]:   : is an INVALID hex value
Index [3]: 1 : is a valid hex value
Index [4]: 2 : is a valid hex value
Index [5]: G : is an INVALID hex value

linux5[203]% python3 hex_validation.py
Please enter a hexadecimal value: #4 (*
Index [0]: # : is an INVALID hex value
Index [1]: 4 : is a valid hex value
Index [2]:   : is an INVALID hex value
Index [3]: ( : is an INVALID hex value
Index [4]: * : is an INVALID hex value
```

Problem 2 - Converting a String Text

This might **also** look familiar to some of you! Your program, hex_conversion.py will accept a string from a user. The program, **requiring** the use of **string splicing (or string index) and a loop**, will convert a string (Hexadecimal value) into Decimal. A loop must also be used in creating the overall calculation. Validation is not required. To help you develop a plan, use this website. It gives all the details you need to convert.

Sample Output

```
linux5[213]% python3 hex_conversion.py
Please enter a hexadecimal value: AF12
The decimal value of that is: 44818

linux5[214]% python3 hex_conversion.py
Please enter a hexadecimal value: 1234
The decimal value of that is: 4660

linux5[213]% python3 hex_conversion.py
Please enter a hexadecimal value: FF12
The decimal value of that is: 65298
```

Problem 3 - Deciphering TimeStamps

Timestamps are used for so many applications. For this program, timestamps will come in the form: HH:MM:SS. You will not need to validate the data entered. Your program, timestamp_diff.py, will convert the difference of two timestamp values into seconds. Notice the use of military time. (More information about that here) Again the program will **require** the use of **string splicing** to complete the calculation.

Sample Output

```
linux5[231]% python3 timestamp_diff.py
Please enter timestamp #1: 02:48:43
Please enter timestamp #2: 04:53:53
That is 2 hour(s), 5 minute(s), and 10 seconds
But in seconds that is: 7510

Please enter timestamp #1: 02:48:43
Please enter timestamp #2: 08:33:12
That is 5 hour(s), 44 minute(s), and 29 seconds
But in seconds that is: 20669

Please enter timestamp #1: 02:48:43
Please enter timestamp #2: 18:33:12
That is 15 hour(s), 44 minute(s), and 29 seconds
But in seconds that is: 56669
```