CSc 110 Assignment 7:
Lists and Tuples

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee. All materials provided to you for this work are copyrighted, these and all solutions you create for this work cannot be shared in any form (digital, printed or otherwise). Any violations of this will be investigated and reported to Academic Integrity.

Learning Outcomes:

When you have completed this assignment, you should understand:

How to design functions that and mutate list arguments
How to use Type Aliasing to design type hints
How to design functions that take lists of Tuples

Getting started

1. Create a new file called `assignment7.py` and open it in your Wing editor
2. Design the functions according to the specifications in the Function Specifications section.

Submission

1. Double check your file before submission to avoid a **zero grade** for your submission for issues described in the Grading section of this document:
    a. Open and run `assignment7.py` in your Wing editor using the green arrow.
    You should see no errors and no output after the shell prompt.
    If there are errors, you must fix them before submitting.
    If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    b. At the shell prompt in Wing (>>>) type the following: `import assignment7`
    You should see no errors and no output after the shell prompt.
    If there are errors, you must fix them before submitting.
    If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    c. At the shell prompt in Wing (>>>), make calls to the required functions to ensure you have named them correctly. Ensure the function is exhibiting the expected behaviour when called and does not contain any unexpected output. You should be able to make calls to the following functions with expected input. Note: When making these calls, you must precede it with the module name ie. `assignment7.multiply_by([1, 2], [3, 5])`
        i. multiply_by
        ii. create_date
        iii. create_show
        iv. is_actor_in_show
        v. get_titles
        vi. count_shows_before_date
        vii. get_shows_with_actor

2. Upload your `assignment7.py` containing the completed function designs to BrightSpace

Grading:

- Late submissions will be given a **zero grade**.
- The files you submit must be named **`assignment7.py`**
  The filenames must be EXACT for them to work with our grading scripts. Errors in the filenames will result in a **zero grade**.
  Example mistakes often made by students include but are not limited to: spelling errors, different case letters, space characters and incorrect extension (not using .py)
- Your function names and the order of your function arguments must match EXACTLY as specified in this document or you will be given a **zero grade**. Use the example tests we give you to ensure your function header is correct.
- Your submission must not contain any print statements that are not required in the specification or any top-level calls to functions. This unexpected code can cause the automated tester to crash and will result in a **zero grade**.
- We will do **spot-check grading** in this course. That is, all submissions are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors with Python 3. If you tested your configuration with setup.py file this would have verified you are using Python 3. Code that generates errors cannot be tested and will be given a **zero grade**.

Marks will be awarded for correctness, considering:

- the function signature matches the description given (has the name and arguments EXACTLY as specified)
- the function has the expected behaviour

and for code quality according to software engineering properties such as:

- documentation in docstring format: type hints, including return type, purpose, examples as demonstrated in lecture.
- Test coverage – examples within the function docstring cover all boundary cases of all conditions within your function
- readability
  - use of whitespace
  - splitting complex computation or long statements across multiple lines
- meaningful variable names
  - lower case, starting with an alpha-character
- proper use of constants (avoid magic numbers)
  - defined above all function definitions
  - in UPPERCASE
- use of code constructs (functions, variables) to:
  - use of helper functions where appropriate
  - eliminate redundant code and redundant computation
  - make complex code easier to read

Function Specifications

You cannot use list comprehensions to solve any of these problems. If you don't know what these are, don't worry about it, you don't need to know about them for this course.
Examples provided are for explanation purposes. You can/should create your own example data for additional testcases. Your testcases do NOT need to use real Netflix data as in examples provided.

1. Design the function `multiply_by` that takes 2 lists as argument. The first list will contain elements that can be multiplied by an integer. The second list will contain non negative integers. The lists are not guaranteed to be the same length.
   The function should multiply every element in first list by the value at the corresponding position in the second list. Notice, the first list **will be changed** by the function but the second list should remain unchanged. Therefore the function does not return a value.
   For example:
   - if the first list is `[1, 2, 3]` and the second list is `[2, 4, 0]` then first list will be updated to `[2, 8, 0]`
   - if the first list is `[1, 2, 3]` and the second list is `[2, 4]` then first list will be updated to `[2, 8, 3]`
   - if the first list is `[1, 2, 3]` and the second list is `[2, 4, 0, 2]` then first list will be updated to `[2, 8, 0]`

2. Design a type alias to represent the following date information as a tuple:
   - the year number
   - the month number
   - the day number

The type alias should assume values comprise a valid date in the calendar.
For example, the following tuple below represents the date June 28, 2021: `(2021, 6, 28)`

REMINDER: type aliases should be at the TOP of your file under any import statements. If they are not there we will not find them to grade them.

3. Design a type alias to represent the following Netflix show information as a tuple:
   - The type of show
   - The title of the show
   - A list of show director names
   - A list of show actor names
   - The date the show was added (you are going to make use of your data alias here)

   For example, the tuple below represents the Netflix show that is
   - has a show type of: Movie
   - with the title: The Invention of Lying
   - directed by the following 2 directors: Ricky Gervais and Matthew Robinson
   - acted in by the following 10 actors: Ricky Gervais, Jennifer Garner, Jonah Hill, Louis C.K., Jeffrey Tambor, Fionnula Flanagan, Rob Lowe, Tina Fey, Donna Sorbello, and Stephanie March
   - date the show was added to Netflix: January 1, 2020 (notice: this value is also tuple in the date format as specified in date alias specification above)

```
('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'],
['Ricky Gervais', 'Jennifer Garner', 'Jonah Hill', 'Louis C.K.', 'Jeffrey
```

```
Tambor', 'Fionnula Flanagan', 'Rob Lowe', 'Tina Fey', 'Donna Sorbello',
'Stephanie March'], (2020, 1, 1))
```

4. Design the function `create_date` that takes a string representing a valid date and returns a date tuple (as described in specification 2).
   The string passed to the function is guaranteed to be in the following format: 'day-month-year' where,
   - day is a 2 digit integer representing a valid day of the month
   - month is the first 3 letters of a valid month, where the first letter is uppercase
   - year is a 2 digit integer representing a year in the 2000s

   Your function should assume the values in the given string comprise a valid date.

   For example,
   if the function is called with the string `('10-Jan-18')` it will return the tuple `(2018, 1, 10)`
   if the function is called with the string `('22-Feb-00')` it will return the tuple `(2000, 2, 22)`

   TIP: Notice the values within the string are separated by − characters. Experiment with the string method `split` in your shell. The `split` string method is called on a string and takes an additional string as an argument to split on and returns the original string split into list of strings.
   Try the following in the shell:

   ```
   s = '22-Feb-00'
   returned_list = s.split('-')
   ```
   what does `returned_list` look like?

   ```
   s = 'ab:cd:ef'
   returned_list = s.split(':')
   ```
   what does `returned_list` look like?

   ```
   s = ''
   returned_list = s.split(':')
   ```
   what does `returned_list` look like?

   NOTE: in this question you are guaranteed the string will not be empty since it must be in the 'day-month-year' format described above.

5. Design the function `create_show` that takes 5 strings as arguments and creates and returns a Netflix Show tuple (as described in specification 3).
   The strings passed to the function will be in the following order and format:
   - the type of show, ie. `'Movie'`
   - the title of the show, ie. `'The Invention of Lying'`
   - the names of the directors of the show separated by `:` characters,
     ie. here is a string with 2 directors:
     `'Ricky Gervais:Matthew Robinson'`
   - the names of the actors of the show separated by `:` characters,
     ie. here is a string with 5 actors:
     `'Ricky Gervais:Jennifer Garner:Jonah Hill:Rob Lowe:Tina Fey'`
   - the date the show was added to Netflix in format 'day-month-year' where,
     - day is a 2 digit integer representing a valid day of the month
     - month is the first 3 letters of a valid month, where the first letter is uppercase
     - year is a 2 digit integer representing a year in the 2000s

     Your function should assume the values in the given string comprise a valid date and will not be an empty string.
     ie. here is a string representing January 2, 2020: `'02-Jan-18'`

   Examples:

   The following call:
   ```
   create_show('Movie', 'The Invention of Lying', 'Ricky Gervais:Matthew Robinson',
   'Ricky Gervais:Jennifer Garner:Jonah Hill:Rob Lowe:Tina Fey', '02-Jan-18')
   ```
   Should return the following tuple:
   ```
   ('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'], ['Ricky
   Gervais', 'Jennifer Garner', 'Jonah Hill', 'Rob Lowe', 'Tina Fey'], (2018, 1, 2))
   ```

   The following call:
   ```
   create_show('TV Show', 'The Mind Explained', '', 'Emma Stone', '12-Sep-09')
   ```
   Should return the following tuple:
   ```
   ('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2009, 9, 12))
   ```

   The following call:
   ```
   create_show('Movie', 'The Bad Kids', 'Keith Fulton:Louis Pepe', '', '01-Apr-17')
   ```
   Should return the following tuple:
   ```
   ('Movie', 'The Bad Kids', ['Keith Fulton', 'Louis Pepe'], [], (2017, 4, 1))
   ```

6. Design the function `get_titles` that takes a list of Netflix show tuples (as described in specification 3). The function should return a list of the titles of each of Netflix shows in the list in the order they appear in the given list of Netflix show tuples.
   For Example, if the list passed is:
   ```
   [('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'], ['Ricky
   Gervais', 'Jennifer Garner', 'Jonah Hill', 'Rob Lowe', 'Tina Fey'], (2018, 1, 2)),
   ('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2009, 9, 12))]
   ```
   the function should return the list: `['The Invention of Lying', 'The Mind Explained']`

7. Design the function `is_actor_in_show` that takes the following two arguments:
   a Netflix show tuple (as described in specification 3) and the name of an actor. The function should return True if the given actor is an actor in the given Netflix show tuple and False otherwise. The function should ignore case in the comparison of names.
   NOTE: The lists within the tuples should remain unchanged.
   Examples:
   If the movie passed is:
   ```
   ('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'], ['Ricky
   Gervais', 'Jennifer Garner', 'Jonah Hill', 'Rob Lowe', 'Tina Fey'], (2018, 1, 2))
   ```
   and the actor passed is: `'Rob Lowe'`
   the function should return `True`

   If the movie passed is:
   ```
   ('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'], ['Ricky
   Gervais', 'Jennifer Garner', 'Jonah Hill', 'Rob Lowe', 'Tina Fey'], (2018, 1, 2))
   ```
   and the actor passed is: `'roB lowE'`
   the function should return `True`

   If the movie passed is:
   ```
   ('Movie', 'The Invention of Lying', ['Ricky Gervais', 'Matthew Robinson'], ['Ricky
   Gervais', 'Jennifer Garner', 'Jonah Hill', 'Rob Lowe', 'Tina Fey'], (2018, 1, 2))
   ```
   and the actor passed is: `'Emma Stone'`
   the function should return `False`

8. Design the function `count_shows_before_date` that takes the following two arguments:
   list of Netflix show tuples (as described in specification 3) and a date tuple (as described in specification 2). The function should return a count of the number only the Netflix show tuples for which the date they were added is BEFORE the given date in the calendar.
   Examples:
   If the list passed is the following 4 element list:
   ```
   [('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
   'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
   Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
    ('Movie', 'The Bad Kids', ['Keith Fulton', 'Louis Pepe'], [], (2017, 4, 1)),
    ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
   Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
   'Jemima Kirke'], (2018, 9, 21)),
   ('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]
   ```
   and the date passed is: `(2018, 12, 12)`

the function should return the value `2`
since the dates of only `'The Bad Kids'` and `'Maniac'` are before `(2018, 12, 12)`

9. Design the function `get_shows_with_actor` that takes the following two arguments:
a list of Netflix show tuples (as described in specification 3) and the name of an actor. The function should return a list of only the Netflix show tuples that the given actor has acted in. The function should ignore case in the comparison of names. The values in the returned list should be in the order they appear in the given list of Netflix show tuples.
NOTE: The lists within the tuples should remain unchanged.
Examples:
If the list passed is the following 4 element list:

```
[('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
 ('Movie', 'The Bad Kids', ['Keith Fulton', 'Louis Pepe'], [], (2017, 4, 1)),
 ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
'Jemima Kirke'], (2018, 9, 21)),
('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]
```

and the actor passed is: `'Emma Stone'`
the function should return the following 3 element list:

```
[('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
 ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
'Jemima Kirke'], (2018, 9, 21)),
('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]
```

If the list passed is the following 4 element list:

```
[('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
 ('Movie', 'The Bad Kids', ['Keith Fulton', 'Louis Pepe'], [], (2017, 4, 1)),
 ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
'Jemima Kirke'], (2018, 9, 21)),
('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]
```

and the actor passed is: `'emmA sTone'`
the function should return the following 3 element list:

```
[('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
 ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
'Jemima Kirke'], (2018, 9, 21)),
('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]
```

If the list passed is the following 4 element list:

```
[('Movie', 'Superbad', ['Greg Mottola'], ['Jonah Hill', 'Michael Cera',
'Christopher Mintz-Plasse', 'Bill Hader', 'Seth Rogen', 'Martha MacIsaac', 'Emma
Stone', 'Aviva Baumann', 'Joe Lo Truglio', 'Kevin Corrigan'], (2019, 9, 1)),
 ('Movie', 'The Bad Kids', ['Keith Fulton', 'Louis Pepe'], [], (2017, 4, 1)),
 ('TV Show', 'Maniac', [], ['Emma Stone', 'Jonah Hill', 'Justin Theroux', 'Sally
Field', 'Gabriel Byrne', 'Sonoya Mizuno', 'Julia Garner', 'Billy Magnussen',
'Jemima Kirke'], (2018, 9, 21)),
```

`('TV Show', 'The Mind Explained', [], ['Emma Stone'], (2019, 9, 12))]`
and the actor passed is: `'Ricky Gervais'`
the function should return an empty list: `[]`