**Question 1.** [5 marks]

Fill in the boxes below with what you would see in your Python shell if you were to type the following expressions. If that code would result in an error, then write ERROR and provide a brief explanation in the corresponding box.

<table>
<thead>
<tr>
<th>(a)</th>
<th>&gt;&gt;&gt; 'This is a String!' .upper()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'THIS IS A STRING!'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b)</th>
<th>&gt;&gt;&gt; print('la' * 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lalala</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(c)</th>
<th>&gt;&gt;&gt; (10 &gt; 3) or (10 / 0 == 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(d)</th>
<th>&gt;&gt;&gt; 'hi' + 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERROR Cannot concatenate a string with an int.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(e)</th>
<th>&gt;&gt;&gt; len('\n')</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(f)</th>
<th>&gt;&gt;&gt; season = 'summer'</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt;&gt;&gt; season[3:5]</td>
</tr>
<tr>
<td></td>
<td>'me'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(g)</th>
<th>&gt;&gt;&gt; type(10) == type(10 / 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(h)</th>
<th>&gt;&gt;&gt; 10 // 3 ** 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(i)</th>
<th>&gt;&gt;&gt; my_list = [108, 15, 2]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&gt;&gt;&gt; val = my_list.append(7)</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;&gt; my_list + []</td>
</tr>
<tr>
<td></td>
<td>[108, 15, 2, 7]</td>
</tr>
<tr>
<td></td>
<td>&gt;&gt;&gt; val</td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
</tbody>
</table>

Common Mistakes/Reminders:

- Forgetting the quotes in (a), (f)
- Adding quotes in (b). When we print this string no quotes should be displayed.
- (c): Python does lazy evaluation. Since True or (anything) evaluates to True, Python will not evaluate the (10 / 0 == 0) expression.
- (g): type(10 / 1) is float
- (h): Forgetting that exponentiation takes precedence over integer division.
- (i): List method **append**, modifies the existing list and returns value None. It does not return a new list; see the method description in the last page of the midterm.
Question 2. [3 marks]

Complete the docstring examples below by adding arguments that result in the return values shown. When filling in the arguments for each example call, keep in mind that there may be several correct answers. Providing any of them will earn you full marks as long as you respect all the specified requirements (see comment next to each example). The function description has been left out on purpose.

```python
def mystery(L):
    """ (list of bool) -> bool
    >>> mystery([False]) # 1-element list
        True
    >>> mystery([True, True]) # 2-element list with first given
        False
    >>> mystery([True, False]) # 2-element list - ([False, True] is also correct)
        True
    """

    result = False
    for item in L:
        result = (result and item) or (not result and not item)
    return result
```

Here are all possible example calls for this function for a 1-element and 2-element lists:

```python
>>> mystery([True])
False
>>> mystery([False])
True
>>> mystery([True, True])
False
>>> mystery([True, False])
True
>>> mystery([False, True])
True
>>> mystery([False, False])
False
```

Reminders:

- Always make sure your examples meet the type contract. Since this is a list, you need the square brackets.
- Try not to overcomplicate things when you fill in examples. Writing True is less error prone than writing 3 < 5.
Question 3. [5 marks]

(a) Read the header, body and precondition of the function below and then complete its docstring according to the function design recipe. Give a meaningful function name and provide the type contract, the description, and two examples that return different values.

```python
def first_uppercase_index(s):
    """ (str) -> int

    Precondition: s.isalpha() evaluates to True.

    Return the index of the first uppercase letter in s or -1 if no uppercase letter is found in s.
    
    >>> first_uppercase_index('helloThere')
    5
    >>> first_uppercase_index('hellothere')
    -1
    """

    i = 0
    while i < len(s):
        if s[i].isupper():
            return i
        i = i + 1
    return -1
```

(b) What does the precondition imply about `len(s)`? `len(s) > 0` (or you could write `len(s) >= 1`).

Reminders:

- You need to mention each parameter by name in your description.
- In this example your description should start with Return.
- Your examples should respect the type contract and any preconditions. For example, you should not include a string that contains digits or punctuation.
- This is an invalid example:
  ```python
  >>> first_uppercase_index(apple)
  You need the quotes. Otherwise it’s just a variable named apple.
  ```
- In a question that asks you about `len(s)`, you need to talk about the length of the string.
Question 4.  [5 MARKS]
Write the body of the following function according to its docstring description.

```python
def score_update(scores, min_score, update_points):
    """ (list of int, int, int) -> NoneType
    Precondition: each item in scores is >= 0.
    Modify each player's score in scores according to the following algorithm:
    A player with a score of 0 should get a new score of min_score.
    A player with a non-zero, even score should gain update_points.
    A player with an odd score should lose update_points, but if using this rule
    means that a player's updated score is less than min_score, the player should
    get a new score of min_score.
    >>> game_scores = [20, 0, 55, 7, 4]
    >>> score_update(game_scores, 10, 5)
    >>> game_scores
    [25, 10, 50, 10, 9]
    """
    for i in range(len(scores)):
        if scores[i] == 0:
            scores[i] = min_score
        elif scores[i] % 2 == 0:
            scores[i] = scores[i] + update_points
        elif scores[i] % 2 == 1:
            scores[i] = max(min_score, scores[i] - update_points)
            # Alternative implementation for the line above
            # that does not use max().
            # scores[i] = scores[i] - update_points
            # if scores[i] < min_score:
            #     scores[i] = min_score
```

Common Mistakes:

- Creating and returning a new list. This violates the type-contract and does not match the function description.
- Using `game_scores` instead of `scores`. `game_scores` is the variable name used in the docstring example; this is not something that you can use within the body of your function. You need to use the parameter name (i.e., `scores`) instead.
- Using a `for item in scores` loop and changing item. This does not mutate (modify) the list.
- Using separate if-statements. This will make any changes to scores propagate across if statements and so you may end up changing a scores element multiple times.
- Confusing the equality operator `==` with the assignment operator `=`.
Question 5.  [6 marks]

Write the body of the following function according to its docstring description.

def hidden_message(words):
    """ (list of str) -> str

    Preconditions:
    - len(words) >= 1
    - Each string in words is not empty.
    - Each string in words contains only alphanumeric characters.

    Return all the hidden alphabetic characters in words as a single string with each character in the order in which it appears in words. Add a single '-' character after each group of characters (empty or otherwise). All alphabetic characters hidden in the same list element make up a group; that group may be empty.

    >>> hidden_message(['12G3', 'r491', '541e', '12', 'a3t'])
    'G-r-e--at-
    >>> hidden_message(['746Job'])
    'Job-
    """

message = ''
for word in words:
    for char in word:
        if char.isalpha():
            message += char
        message += '-'
return message

Comments/Reminders:

- Use meaningful variable names so they help you when you code. For example, writing for index in words is misleading as index refers to a word (an element of the words list) and not an index.

- Whenever you use a variable in an expression, think first what its type is. For example, if you wrote for word in words you cannot use word as an index in your nested loop.

- Make sure that you don’t limit the functionality of your code. For example, do not assume that each word in words is either all alphabetic characters or all digits; the provided examples show otherwise.