A paragraph object has a variety of **properties** that specify its placement within a container—typically a page—and the way it divides its contents into separate lines. You can access the formatting **properties** of a paragraph with the `ParagraphFormat` object available through the `ParagraphFormat` property of the paragraph, and you can set all the paragraph **properties** using a `paragraph style grouping` or apply them directly to a paragraph.

A **run** is an inline-level object that occurs within paragraphs or other block-level objects. A **run** object has a read-only `font` **property** providing access to a font object. A font object provides **properties** for getting and setting the character formatting for that **run**. You’ll need this feature for setting your hidden message’s text color to white.
<table>
<thead>
<tr>
<th>Page</th>
<th>Error</th>
<th>Correction</th>
<th>Print corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>Define a function that formats the spacing between paragraphs using python-docx's <code>paragraph_format</code> property.</td>
<td>Define a function that formats the spacing between paragraphs using python-docx's <code>paragraph_format</code> attribute.</td>
<td>Print 5</td>
</tr>
</tbody>
</table>
| 141  | # mutate  
  lock_wheel = int(randrange(0, len(combo)))  
  next_try[lock_wheel] = randint(0, len(combo)-1) | # mutate  
  lock_wheel = randrange(0, len(combo))  
  next_try[lock_wheel] = randint(0, 9) | Print 3 |
| 156  | . . . and adding the key/value pair (at any location, since dictionaries are unordered). | . . . and adding the key/value pair at any location. | Print 5 |
| 164  | Because of the very short training corpus, the moon is the only word pair with multiple keys. | Because of the very short training corpus, the moon is the only word pair with multiple values. | Print 4 |
| 171  | This is a far better solution than manually finding and commenting out `print()` statements! | This is a far better solution than manually finding and commenting out calls to `print()`! | Print 6 |
| 182  | Cool stars enter the Window this hot evening all  
  Heaven and earth ache | A line flap-flapping  
  Across the dark crimson sky  
  On this winter pond | Print 5 |
| 205  | The transformation to generate points over a unit disc is: \( x = \sqrt{r}\cos \)  
  The equations yield \((x, y)\) values between 0 and 1. | The transformation to generate points evenly over a unit disc is: \( x = \sqrt{r}\cos\theta \)  
  The equations yield \((x, y)\) values between \(-1\) and 1. | Print 3 |
| 218  | >>> from random import randint  
  >>> trials = 100000  
  >>> success = 0  
  >>> for trial in range(trials):  
  1 | >>> from random import randint  
  >>> trials = 100000  
  >>> success = 0  
  >>> for trial in range(trials):  
  1 | Print 2 |
| 250  | >>> prompt = '{0} [{}]: '.format(prompt, default)  
  >>> response = input(prompt)  
  >>> if not response and default: | >>> prompt = '{0} [{}]: '.format(prompt, default)  
  >>> response = input(prompt)  
  >>> if not response and default: | Print 2 |
<p>| 252  | Set the default to ‘sbc_blend’, since this is theoretically the most stable mix of the four choices. | Set the default to ‘bonds’, in order to see how this supposedly ‘safe’ choice performs. | Print 3 |</p>
<table>
<thead>
<tr>
<th>Page</th>
<th>Error</th>
<th>Correction</th>
<th>Print corrected</th>
</tr>
</thead>
<tbody>
<tr>
<td>259</td>
<td>... a 4 percent withdrawal rate (equal to $80,000 per year), a 30-year retirement, and 50,000 cases.</td>
<td>... a 4 percent withdrawal rate (equal to $80,000 per year), a 29-30-31 retirement range, and 50,000 cases.</td>
<td>Print 3</td>
</tr>
</tbody>
</table>
| 261  | ```
else:
    withdraw_infl_adj = withdraw_infl_adj_2
investments -= withdraw_infl_adj
investments = int(investments * (1 + i))
```                                                                                                                                             | ```
else:
    withdraw_infl_adj = withdraw_infl_adj_2
investments -= withdraw_infl_adj
investments = int(investments * (1 + i))
```                                                                                                                                             | Print 3         |
| 305  | You'll use the same transform.rotate() method you used to turn the satellite                                                                                                                             | You'll use the same transform.rotate() method you used to turn the satellite                                                                                                                            | Print 3         |
| 329  | The shell utilities module, shutil, provides high-level functions for working with files and folders, such as copying, moving, renaming, and deleting.                                                      | The shell utilities module, shutil, provides high-level functions for working with files and folders, such as copying, moving, and deleting.                                                              | Print 6         |
| 356  | ```
➎ first_digits[sample[0]] += 1
➏ data_count = [v for (k, v) in sorted(first_digits.items())]
```                                                                                                                                             | ```
➎ first_digits[sample[0]] += 1
➏ data_count = [v for (k, v) in sorted(first_digits.items())]
```                                                                                                                                             | Print 3         |
| 357  | Deletion                                                                                                                                                                                                | Like all Python dictionaries, first_digits is unordered.                                                                                                                                                 | Print 5         |
| 360  | This will work with no arguments, but set its size property to 15 and turn off the frame around the legend for an arguably more attractive result.                                                        | This will work with no arguments, but set its size attribute to 15 and turn off the frame around the legend for an arguably more attractive result.                                                        | Print 5         |
### Remove single-letter words from list if not 'a' or 'i'.

```python
word_list = ['a', 'nurses', 'i', 'stack', 'b', 'cats', 'c']
permissible = ('a', 'i')
for word in word_list:
    if len(word) == 1 and word not in permissible:
        word_list.remove(word)
print("{}").format(word_list_clean)
```

```python
word_list = ['a', 'nurses', 'i', 'stack', 'b', 'c', 'cat']
word_list_clean = []
permissible = ('a', 'i')
for word in word_list:
    if len(word) > 1:
        word_list_clean.append(word)
    elif len(word) == 1 and word in permissible:
        word_list_clean.append(word)
    else:
        continue
print("{}").format(word_list_clean)
```

### Print digrams

```python
print(*digrams, sep='
')
```

```python
--snip--
for k in mapped:
```

### Print sorted digrams

```python
print(*sorted(digrams, sep='\n')
```

```python
--snip--
for k in sorted(mapped):
```