1. In what modes should the PdfFileReader() and PdfFileWriter() File objects will be opened?

2. From a PdfFileReader object, how do you get a Page object for page 5?

3. What PdfFileReader variable stores the number of pages in the PDF document?

4. If a PdfFileReader object’s PDF is encrypted with the password swordfish, what must you do before you can obtain Page objects from it?

5. What methods do you use to rotate a page?

6. What is the difference between a Run object and a Paragraph object?

7. How do you obtain a list of Paragraph objects for a Document object that’s stored in a variable named doc?

8. What type of object has bold, underline, italic, strike, and outline variables?

9. What is the difference between False, True, and None for the bold variable?

10. How do you create a Document object for a new Word document?

11. How do you add a paragraph with the text 'Hello, there!' to a Document object stored in a variable named doc?

12. What integers represent the levels of headings available in Word documents?

ANSWER

1. In PyPDF2, the `PdfFileReader()` object should be opened in read-binary mode ('rb'), and the `PdfFileWriter()` object should be opened in write-binary mode ('wb').

2. To get a Page object for page 5 from a `PdfFileReader` object, you can use the `getPage()` method like this:

```python

page = pdf\_reader.getPage(4) # Page numbers are zero-based, so page 5 is at index 4.

```

3. The number of pages in a PDF document stored in a `PdfFileReader` object can be obtained using the `numPages` attribute:

```python

num\_pages = pdf\_reader.numPages

```

4. If a PDF is encrypted with the password 'swordfish', you need to decrypt it using the `decrypt()` method before you can obtain Page objects:

```python

pdf\_reader.decrypt('swordfish')

```

5. To rotate a page, you can use the `rotateClockwise()` or `rotateCounterClockwise()` methods of the `Page` object. For example:

```python

page.rotateClockwise(90) # Rotate the page 90 degrees clockwise.

```

\*\*Word Document Manipulation (using python-docx):\*\*

6. In python-docx, a "Run" object represents a contiguous run of text with the same style, while a "Paragraph" object represents a paragraph of text that may contain multiple runs.

7. To obtain a list of Paragraph objects from a Document object named `doc`, you can use the `paragraphs` attribute, like this:

```python

paragraphs = doc.paragraphs

```

8. The "Run" object in python-docx has various style-related attributes like `bold`, `underline`, `italic`, `strike`, and `outline`.

9. In python-docx, the `bold` variable for a "Run" object can take three values:

- `True`: Text is bold.

- `False`: Text is not bold.

- `None`: The text inherits the style from the parent element (e.g., the Paragraph it belongs to).

10. To create a Document object for a new Word document, you can use the `Document()` constructor:

```python

from docx import Document

doc = Document()

```

11. To add a paragraph with the text 'Hello, there!' to a Document object stored in a variable named `doc`, you can use the `add\_paragraph()` method:

```python

doc.add\_paragraph('Hello, there!')

```

12. In Word documents, heading levels are typically represented by integers from 1 to 9, where 1 is the highest level (main heading) and 9 is the lowest level (sub-sub-sub-sub-sub-sub-sub-subheading).