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

PdfFileReader() File objects should be opened in 'rb' mode (read binary), indicating that we are opening the file for reading binary data.

Similarly PdfFileWriter() File objects should be opened in 'wb' mode (write binary) to indicate that we are opening the file for writing binary data.

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

To get a Page object for page 5 from a PdfFileReader object, we can use the getPage() method and pass the index of the desired page (zero-based index).

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

The PdfFileReader class in the PyPDF2 library has a property called numPages that stores the number of pages in the PDF document. we can access this property to retrieve the total number of pages.

**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?**

If a PdfFileReader object's PDF is encrypted with the password "swordfish," we need to decrypt the PDF before we can obtain Page objects from it.

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

To rotate a page in PyPDF2, we can use the rotateClockwise() or rotateCounterClockwise() methods provided by the Page object. These methods allows us to rotate the page by 90 degrees clockwise or counterclockwise, respectively. After rotating the page, we can save the changes to a new PDF file or overwrite the existing one.

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

Run Object:

A "Run" object usually represents a continuous span of text within a paragraph that has a consistent style or formatting. It is often used to apply specific formatting to a portion of text, such as font size, font family, bold, italic, underline, etc. In a document, a paragraph can contain multiple runs, each with its own distinct formatting. For example, within a single paragraph, you may have one run of bold text followed by a run of italicized text.

Paragraph Object:

A "Paragraph" object represents a block of text that is typically separated from other paragraphs by line breaks or other spacing. It is a structural unit that helps organize the content of a document. A paragraph may contain one or more runs to handle different formatting within the paragraph. The formatting of a paragraph itself can also be controlled, including properties such as alignment, indentation, spacing, bullet points, numbering, and more.

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

doc.paragraphs is used to obtain the list of Paragraph objects from the Document object doc. We can then iterate over the paragraphs using a loop or access them individually for further processing.

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

A Run object has bold, underline,italic,strike and outline variables

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

False: It indicates that the bold formatting should be turned off.

True: It indicates that the bold formatting should be applied.

None: If it is set to None, it might indicate that no explicit instruction for bold formatting is given and the default behavior or previous setting should be used.

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

Import the Document class from the docx module. Then create a new Document object by calling Document(), which initializes an empty Word document.

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

First import the Document class from the docx module. Create a new Document object by calling Document(). Then add a paragraph to the document using the add\_paragraph() method, passing the desired text 'Hello, there!' as an argument. Finally save the document with the save() method, specifying the desired file name (e.g., 'document.docx').

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

integer from 0 to 4 represent the levels of headings available in Word documents.

The integer 0 makes the heading the Title style, which is used for the top of the document. Integers 1 to 4 are for various heading levels, with 1 being the main heading and 4 the lowest subheading.