**ANSWERS**

**Q1. In what modes should the PdfFileReader() and PdfFileWriter() File objects will be opened?**

**Ans.** When using the PdfFileReader() and PdfFileWriter() objects in PyPDF2 library, you don't need to open the file in any specific mode. Instead, you pass the file object itself when initializing these objects. For example:

from PyPDF2 import PdfFileReader, PdfFileWriter

file = open('example.pdf', 'rb')

pdf\_reader = PdfFileReader(file)

pdf\_writer = PdfFileWriter()

In the example above, the file is opened in 'rb' mode (read binary) to ensure proper handling of the PDF file.

**Q2. From a PdfFileReader object, how do you get a Page object for page 5?**

**Ans.** To get a Page object for page 5 from a PdfFileReader object, you can use the getPage() method and pass the page number (zero-indexed) as an argument. Here's an example:

page\_number = 4 # Page 5 (zero-indexed)

page = pdf\_reader.getPage(page\_number)

**Q3. What PdfFileReader variable stores the number of pages in the PDF document?**

**Ans.** The numPages attribute of a PdfFileReader object stores the number of pages in the PDF document. You can access it using the following syntax:

num\_pages = pdf\_reader.numPages

**Q4. If a PdfFileReader object’s PDF is encrypted with the password swordfish, what must you do before you can obtain Page objects from it?**

**Ans.** If a **PdfFileReader** object's PDF is encrypted with the password "swordfish," you need to use the **decrypt()** method to decrypt it before obtaining **Page** objects. Here's an example:

pdf\_reader.decrypt('swordfish')

**Q5. What methods do you use to rotate a page?**

**Ans.** To rotate a page in PyPDF2, you can use the rotateClockwise() or rotateCounterClockwise() methods of a Page object. These methods take the rotation angle in degrees as an argument. Here are a couple of examples:

page.rotateClockwise(90) # Rotate 90 degrees clockwise

page.rotateCounterClockwise(180) # Rotate 180 degrees counter-clockwise

**Q6. What is the difference between a Run object and a Paragraph object?**

**Ans.** In the context of python-docx library, a **Run** object represents a contiguous run of text within a paragraph with the same formatting. It is the smallest unit of text in Word documents and can have its own formatting properties. On the other hand, a **Paragraph** object represents a single paragraph in a Word document and contains multiple runs.

**Q7. How do you obtain a list of Paragraph objects for a Document object that’s stored in a variable named doc?**

**Ans.** To obtain a list of Paragraph objects from a Document object stored in the variable doc, you can use the paragraphs attribute. Here's an example:

paragraphs = doc.paragraphs

The **paragraphs** attribute returns a list of **Paragraph** objects.

**Q8. What type of object has bold, underline, italic, strike, and outline variables?**

**Ans.** The **Run** object in python-docx has the variables **bold**, **underline**, **italic**, **strike**, and **outline** to represent different formatting options. It allows you to apply various formatting styles to specific runs of text within a paragraph.

**Q9. What is the difference between False, True, and None for the bold variable?**

**Ans.** In python-docx, the **bold** variable for a **Run** object can have three possible values:

* + **True** indicates that the text is formatted as bold.
  + **False** indicates that the text is not formatted as bold.
  + **None** indicates that the text inherits the bold formatting from its containing paragraph.

**Q10. How do you create a Document object for a new Word document?**

**Ans.** To create a **Document** object for a new Word document using the python-docx library, you can simply call the **Document()** constructor without any arguments. Here's an example:

from docx import Document

doc = Document()

**Q11. How do you add a paragraph with the text 'Hello, there!' to a Document object stored in a variable named doc?**

**Ans.** 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 and pass the text as an argument, like this: doc.add\_paragraph('Hello, there!').

**Q12. What integers represent the levels of headings available in Word documents?**

**Ans.** The integers 0 to 8 represent the levels of headings available in Word documents, with 0 being the highest level (e.g. Heading 1) and 8 being the lowest level (e.g. Heading 8).