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

**Ans.** The `PdfFileReader()` and `PdfFileWriter()` File objects should be opened in binary mode using the 'rb' and 'wb' flags respectively.

Opening a file in binary mode is necessary when working with PDF files because PDF files contain binary data, not text data. Therefore, opening a PDF file in text mode can result in data corruption or loss. By specifying 'rb' or 'wb' flags, the file is opened in binary mode, allowing you to read or write the PDF data without any corruption or loss.

2. 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 in Python, you can use the `getPage()` method of the `PdfFileReader` object, passing the zero-based index of the page you want (i.e., 4 for the fifth page, since the index starts at 0). Here's an example:

**from PyPDF2 import PdfFileReader**

**# Open the PDF file**

**with open('example.pdf', 'rb') as pdf\_file:**

**# Create a PdfFileReader object**

**pdf\_reader = PdfFileReader(pdf\_file)**

**# Get the fifth page (index 4)**

**page\_5 = pdf\_reader.getPage(4)**

After running this code, the variable `page\_5` will contain a `Page` object representing the fifth page of the PDF file. You can then use methods of the `Page` object (e.g., `extractText()`) to extract information from the page.

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

**Ans.** The PdfFileReader variable that stores the number of pages in the PDF document is called **"numPages".**

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?

**Ans.** Before you can obtain Page objects from a PdfFileReader object's PDF that is encrypted with the password "swordfish", you must first decrypt the PDF by providing the correct password. You can do this by calling the `decrypt()` method on the PdfFileReader object and passing the correct password as its argument. Once the PDF has been successfully decrypted, you can obtain Page objects from it as usual.

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

**Ans:** PyPDF2 Package provides 2 methods to rotate a page:

**rotateClockWise()** -> For Clockwise rotation

**rotateCounterClockWise()** -> For Counter Clockwise rotation

The PyPDF2 package only allows you to rotate a page in increments of 90 degrees. You will receive an AssertionError otherwise.

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

**Ans.** In python-docx library, both Run and Paragraph objects are used to represent different parts of a Word document.

A Paragraph object represents a single paragraph of text in a Word document. It can contain multiple Run objects, which are used to represent a contiguous section of text that shares the same formatting (e.g., font, color, bold/italic/underline, etc.). So, a paragraph can be thought of as a container for one or more runs.

In other words, a Paragraph is a high-level object that represents a chunk of text, while a Run is a lower-level object that represents a specific section of that text with a specific formatting.

7. 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 for a Document object stored in a variable named doc in Python, you can use the `paragraphs` attribute of the `doc` variable. This attribute is a list of all the Paragraph objects in the Document.

Here is an example:

**from docx import Document**

**# Load the document**

**doc = Document('example.docx')**

**# Get the list of paragraphs**

**paragraphs = doc.paragraphs**

**# Iterate through the list of paragraphs and print their text**

**for paragraph in paragraphs:**

**print(paragraph.text)**

In this example, we load a Document from a file called 'example.docx' and then obtain a list of Paragraph objects using the `paragraphs` attribute. We then iterate through this list and print the text of each Paragraph object.

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

**Ans.**  **Run** object has bold, underline, italic, strike, and outline variables. The text in a Word document is more than just a string. It has font, size, color, and other styling information associated with it.

A style in Word is a collection of these attributes. A Run object is a contiguous run of text with the same style. A new Run object is needed whenever the text style changes.

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

**Ans**. bold = True # Style Set to Bold

bold = False # Style Not Set to Bold

bold = None # Style is Not Applicable

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

**Ans.** To create a Document object for a new Word document in Python, you can use the python-docx library. Here are the steps:

**->** Install the python-docx library by running the following command in your terminal or command prompt:

**pip install python-docx**

**->** Import the library in your Python code:

**import docx**

**->** Create a new Document object using the docx.Document() constructor:

**doc = docx.Document()**

This will create a new, empty Word document in memory.

**->** You can now start adding content to the document by calling the appropriate methods on the Document object. For example, to add a paragraph of text, you can use the add\_paragraph() method:

**doc.add\_paragraph('This is a paragraph of text.')**

**->** Once you have finished creating the document, you can save it to disk using the save() method:

**doc.save('my\_document.docx')**

This will save the document to a file named "my\_document.docx" in the current working directory.

That's it! You now know how to create a new Word document in Python using the python-docx library.

11. 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 in Python, you can use the following code:

**# Import the necessary libraries**

**from docx import Document**

**from docx.shared import Inches**

**# Create a new paragraph with the text "Hello, there!"**

**new\_paragraph = "Hello, there!"**

**# Open the Document object stored in the variable named doc**

**doc = Document()**

**# Add the new paragraph to the Document object**

**doc.add\_paragraph(new\_paragraph)**

**# Save the Document object**

**doc.save('document\_name.docx')**

In this code, we first import the necessary libraries, including the `Document` class from the `docx` library and the `Inches` class from the `docx.shared` module.

We then create a new paragraph with the text "Hello, there!" and store it in a variable named `new\_paragraph`.

Next, we create a new `Document` object and store it in the variable named `doc`.

We then use the `add\_paragraph()` method of the `Document` class to add the new paragraph to the Document object.

Finally, we save the Document object using the `save()` method and specifying the name of the file.

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

**Ans.** The levels for a heading in a word document can be specified by using the level attribute inside the add\_heading method. There is a total of 5 levels starting for 0 t0 4. where level 0 makes a headline with the horizontal line below the text, whereas the heading level 1 is the main heading. Similarly, the other headings are sub-heading with theirs font-sizes in decreasing order.