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

A. n Python's PyPDF2 library, the **PdfFileReader()** and **PdfFileWriter()** objects should be opened in binary mode by passing the mode argument **'rb'** (read binary) and **'wb'** (write binary), respectively, when creating the **PdfFileReader()** or **PdfFileWriter()** object. This is because PDF files are binary files, and the binary mode ensures that the file is read or written correctly without any modifications to its binary data.

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

A. get a **Page** object for page 5 from a **PdfFileReader** object, we can use the **getPage()** method and pass in the index of the page we want (remembering that page indexing starts at 0). Example:

with open('example.pdf', 'rb') as f:

pdf\_reader = PdfFileReader(f)

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

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

A. The **numPages** variable of a **PdfFileReader** object 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?

A. If a PdfFileReader object's PDF is encrypted with the password "swordfish", we must decrypt it first by calling the **decrypt('swordfish')** method on the PdfFileReader object.

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

A. To rotate a page in a PDF document using PyPDF2, we can use the **rotateClockwise()** and **rotateCounterClockwise()** methods of a **PageObject** instance.

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

A. A Paragraph object in python-docx represents a single paragraph of text in a Word document and contains one or more Run objects, each of which has a specific font style and formatting.

A **Run** object in **python-docx** represents a contiguous run of text in a **Paragraph** object with the same font and formatting. A **Paragraph** object can have multiple **Run** objects to represent text with different styles or formatting.

In summary, a **Paragraph** object represents a single paragraph of text, while a **Run** object represents a contiguous run of text with the same formatting within 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?

A.

paragraphs = doc.paragraphs

for paragraph in paragraphs:

print(paragraph.text)

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

A. The **Font** object has the variables **bold**, **underline**, **italic**, **strike**, and **outline**.

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

A. In Python, the bold variable is used to determine if a font is bold or not in some text.

* **False** means the text is not bold
* **True** means the text is bold
* **None** means the text's boldness is inherited from the surrounding context, rather than being explicitly set to bold or not bold.

So, if the **bold** variable is **False**, the text is not bold; if it's **True**, the text is bold; and if it's **None**, the text's boldness depends on the context in which it is displayed.

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

A. To create a Document object for a new Word document in Python, we can use the **docx.Document()** function from the **python-docx** module.

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

A.

doc = Document()

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

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

A. In Word documents, the following integers represent the levels of headings:

* 0: None
* 1: Heading 1
* 2: Heading 2
* 3: Heading 3
* 4: Heading 4
* 5: Heading 5
* 6: Heading 6
* 7: Heading 7
* 8: Heading 8
* 9: Heading 9