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?

Sol1: PdfFileReader() needs to be opened in read-binary mode by passing 'rb' as the second argument to open(). Likewise, the File object passed to PyPDF2. PdfFileWriter() needs to be opened in write-binary mode with 'wb'.

Sol2: You can get a Page object by calling the getPage() method on a PdfFileReader object and passing it the page number of the page you're interested in—in our case, 5.

Sol3: getNumPages() method of PdfFileReader class stores the no pages in a PDF document.

Sol4: If a PdfFileReader object’s PDF is encrypted with the password swordfish and you're not aware of it. first read the Pdf using the PdfFileReader Class. PdfFileReader class provides a attribute called isEncrypted to check whether a pdf is encrypted or not. the method returns true if a pdf is encrypted and vice versa.  
if pdf is encrypted use the decrypt() method provided by PdfFileReader class first then try to read the contents/pages of the pdf, else PyPDF2 will raise the following error PyPDF2.utils.PdfReadError: file has not been decrypted.

Sol5: PyPDF2 Package provides 2 methods to rotate a page:

1. rotateClockWise() -> For Clockwise rotation
2. 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.

Sol6: The structure of a document is represented by three different data types in python-Docx. At the highest level, a Document object represents the entire document. The Document object contains a list of Paragraph objects for the paragraphs in the document. (A new paragraph begins whenever the user presses ENTER or RETURN while typing in a Word document.) Each of these Paragraph objects contains a list of one or more Run objects.

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.

Sol7:

# Example Program

from docx import Document

doc = Document("sample\_file.docx") # Path of the Docx file

print(doc.paragraphs) # Prints the list of Paragraph objects for a Document

for paragraph in doc.paragraphs:

print(paragraph.text) # Prints the text in the paragraph

Sol8: 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.

Sol9:

bold = True # Style Set to Bold

bold = False # Style Not Set to Bold

bold = None # Style is Not Applicable

Sol10:

# Example Program

from docx import Document

document = Document()

document.add\_paragraph("Learn python programming")

document.save('mydocument.docx')

Sol11:

# Example Program

from docx import Document

doc = Document()

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

doc.save('hello.docx')

Sol12:

The levels for a heading in a word document can be specified by using the level attribute inside the add\_heading method. There are a total of 5 levels statring 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 their's font-sizes in decreasing order.