**Assignment-12**

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

Ans: In Python's PyPDF2 library, when using the PdfFileReader() and PdfFileWriter() classes to read from and write to PDF files respectively, both objects should be opened in binary mode using the 'rb' and 'wb' modes, respectively.

1. 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, passing in the index of the desired page (remembering that page indices start at 0).

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

Ans: In Python, the number of pages in a PDF document can be obtained using the ‘numPages’ property of a ‘PdfFileReader’ object.

1. 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 a password, you must first decrypt the PDF by calling the ‘decrypt(password)’ method with the password as an argument. In this case, since the password is "swordfish", you would call the method with that password. you can then obtain ‘Page’ objects from the ‘PdfFileReader’ object using the ‘getPage(page\_number)’ method.

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

Ans: To rotate a page in a PDF using the PyPDF2 library in Python, you can use the ‘rotateClockwise()’ and ‘rotateCounterClockwise()’ methods of the ‘PageObject’ class.

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

Ans: In the context of working with the ‘python-docx’ library in Python, a ‘Paragraph’ object represents a single paragraph of text in a Word document, while a ‘Run’ object represents a contiguous run of text within a paragraph that has a consistent set of character style formatting.

1. 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 that's stored in a variable named ‘doc’ using the ‘python-docx’ library in Python, you can use the ‘paragraphs’ property of the ‘Document’ object.

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

Ans: The ‘Run’ object in the ‘python-docx’ library in Python has the ‘bold’, ‘underline’, ‘italic’, ‘strike’, and ‘outline’ properties. These properties allow you to apply various formatting styles to a contiguous run of text within a ‘Paragraph’ object.

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

Ans: In the context of the python-docx library in Python, the bold property of a Run object can be set to three different values: True, False, or None.

* If **bold** is set to **True**, the text in the **Run** object will be displayed in bold.
* If **bold** is set to **False**, the text in the **Run** object will not be displayed in bold.
* If **bold** is set to **None**, the bold formatting of the text in the **Run** object will be inherited

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

Ans: In the ‘python -docx’ library in Python, you can create a ‘Document’ object for a new Word document using the following code:

from docx import Document

# Create a new Word document

document = Document()

# Add content to the document

paragraph = document.add\_paragraph('This is the first paragraph.')

document.add\_page\_break()

paragraph = document.add\_paragraph('This is the second paragraph.')

# Save the document to disk

document.save('example.docx')

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

Ans: In the ‘python-docx’ library in Python, you can add a paragraph with the text 'Hello, there!' to a ‘Document’ object stored in a variable named ‘doc’ using the following code:

from docx import Document

# Create a new Word document

doc = Document()

# Add a paragraph to the document with the text 'Hello, there!'

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

# Save the document to disk

doc.save('example.docx')

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

Ans: In Word documents, the integer values used to represent the levels of headings are from 1 to 9, where 1 is the highest level and 9 is the lowest level. Here's a summary of the heading levels and their corresponding integer values:

* Heading 1: Level 1
* Heading 2: Level 2
* Heading 3: Level 3
* Heading 4: Level 4
* Heading 5: Level 5
* Heading 6: Level 6
* Heading 7: Level 7
* Heading 8: Level 8
* Heading 9: Level 9