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

Answer=

The PdfFileReader() and PdfFileWriter() classes are part of the PyPDF2 library, which is used to read and write PDF files in Python. When creating a new PdfFileReader or PdfFileWriter object, you need to pass in a file object that represents the PDF file you want to read or write.

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

Answer=

We can use the getPage() method of the PdfFileReader object to get a Page object for a specific page. The method takes the page number as an argument, where the first page is numbered 0. To get the fifth page, you would use:

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

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

Answer=

The numPages variable of the 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?

Answer=

Before you can obtain Page objects from a PdfFileReader object whose PDF is encrypted with a password, you must first decrypt the PDF using the decrypt method of the PdfFileReader object and passing the correct password as an argument. In this case, you would use the password "swordfish" to decrypt the PDF.

pdf\_reader = PdfFileReader(open('encrypted.pdf', 'rb'))

pdf\_reader.decrypt('swordfish')

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

Answer=

To rotate a page in a PDF document using the PyPDF2 library, you can use the rotateClockwise or rotateCounterClockwise methods of the Page object, depending on the desired rotation direction.

from PyPDF2 import PdfFileReader, PdfFileWriter

pdf\_reader = PdfFileReader(open('input.pdf', 'rb')) # read the pdf

page = pdf\_reader.getPage(0)

page.rotateClockwise(90) # rotate the page clockwise

page.rotateCounterClockwise(90) # rotate the page counter-clockwise

pdf\_writer = PdfFileWriter() # create a new pdf writer

pdf\_writer.addPage(page) # add the rotated page

with open('output.pdf', 'wb') as pdf\_file: # write the rotated pdf

pdf\_writer.write(pdf\_file)

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

Answer=

A Paragraph object represents a paragraph in a Word document. It can contain one or more Run objects and other objects such as inline\_shapes and tables. A Paragraph object can be used to set the formatting of the entire paragraph, such as the alignment, font, and spacing.

A Run object represents a portion of a paragraph in a Word document that has the same formatting. A Run object can be used to apply formatting to a specific portion of text within a paragraph, such as bolding or underlining a word.

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

Answer=

To obtain a list of Paragraph objects for a Document object stored in a variable named doc, you can use the paragraphs property of the Document object. This property returns a list of Paragraph objects for the document, which we can then iterate over or manipulate as needed.

from docx import Document

doc = Document('example.docx')

paragraphs = doc.paragraphs

for para in paragraphs:

print(para.text)

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

Answer=

The type of object that has bold, underline, italic, strike, and outline variables is a Run object in the python-docx library. A Run object represents a portion of a paragraph in a Word document that has the same formatting. These variables are properties of the Run object, and they can be used to apply or check formatting to that specific portion of text, such as bolding, underlining, italicizing, striking or outlining a word or a group of words.

from docx import Document  
  
doc = Document('example.docx') #Open a document  
run = doc.paragraphs[0].runs[0] # Find the first run  
print(run.bold) # check if the run is bold  
print(run.underline) # check if the run is underlined  
print(run.italic) # check if the run is italic  
print(run.strike) # check if the run is struck  
print(run.outline) # check if the run is outlined

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

Answer=

* Runs can be further styled using text attributes. Each attribute can be set to one of three values:

1. True (the attribute is always enabled, no matter what other styles are applied to the run),
2. False (the attribute is always disabled),
3. None (defaults to whatever the run’s style is set to)

* True always makes the Run object bolded and False makes it always not bolded, no matter what the style’s bold setting is. None will make the Run object just use the style’s bold setting

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

Answer=

By Calling the docx.Document() function.

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

Answer=

import docx  
doc = docx.Document()  
  
doc.add\_paragraph('Hello there!')  
doc.save('hellothere.docx')

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

Answer=

* integer from 0 to 4
* The arguments to add\_heading() are a string of the heading text and an integer from 0 to 4. The integer 0 makes the heading the Title style, which is used for the top of the document. Integers 1 to 4 are for various heading levels, with 1 being the main heading and 4 the lowest subheading