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

When working with PdfFileReader() and PdfFileWriter() from the PyPDF2 library, you should open the file objects in the following modes:

* For PdfFileReader(), you should open the file in binary reading mode. This is done by using 'rb' as the mode:

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

reader = PdfFileReader(file)

# Further processing...

* For PdfFileWriter(), you should open the file in binary writing mode. This is done by using 'wb' as the mode:

with open('output.pdf', 'wb') as file:

writer = PdfFileWriter()

# Add pages to the writer

writer.write(file)

The 'b' in 'rb' and 'wb' stands for binary. It’s used because PDF files are binary files, and not text files.

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

You can get a Page object for page 5 from a PdfFileReader object using the getPage() method. Remember that page numbers in PyPDF2 are zero-indexed, so page 5 would be accessed as page 4. Here’s an example:

from PyPDF2 import PdfFileReader

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

reader = PdfFileReader(file)

page5 = reader.getPage(4) # Get the 5th page

# Now you can work with page5

In this example, page5 is a Page object representing the 5th page of the PDF. You can now work with this object to extract text, add annotations, etc.

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

The PdfFileReader object has a property called numPages that stores the number of pages in the PDF document. Here’s how you can use it:

from PyPDF2 import PdfFileReader

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

reader = PdfFileReader(file)

print(f"The PDF document has {reader.numPages} pages.")

In this example, reader.numPages will give you 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?**

If a PDF is encrypted, you must decrypt it before you can obtain Page objects from it. You can do this using the decrypt() method of the PdfFileReader object. Here’s an example:

from PyPDF2 import PdfFileReader

with open('encrypted.pdf', 'rb') as file:

reader = PdfFileReader(file)

reader.decrypt('swordfish') # Decrypt the PDF

page1 = reader.getPage(0) # Now you can get Page objects

In this example, reader.decrypt('swordfish') will decrypt the PDF using the password ‘swordfish’. After this, you can use getPage() and other methods as usual. If the password is incorrect, decrypt() will return 0 and the PDF will remain encrypted.

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

In Python, you can use libraries like PyPDF2 or PyMuPDF (fitz) to rotate a page in a PDF document. Here are some examples:

**Using PyPDF2:**

from PyPDF2 import PdfFileReader, PdfFileWriter

pdf\_in = open('original.pdf', 'rb')

pdf\_reader = PdfFileReader(pdf\_in)

pdf\_writer = PdfFileWriter()

for pagenum in range(pdf\_reader.numPages):

page = pdf\_reader.getPage(pagenum)

if pagenum % 2: # Rotate odd-numbered pages

page.rotateClockwise(180)

pdf\_writer.addPage(page)

pdf\_out = open('rotated.pdf', 'wb')

pdf\_writer.write(pdf\_out)

pdf\_out.close()

pdf\_in.close()

[This script opens a PDF file, rotates odd-numbered pages by 180 degrees, and writes the result to a new PDF1](https://www.johndcook.com/blog/2015/05/01/rotating-pdf-pages-with-python/).

**Using PyMuPDF (fitz):**

import fitz

doc = fitz.open("mypdf.pdf") # Open the PDF

page = doc[n] # Read page n (zero-based)

page.setRotate(-90) # Rotate page by 90 degrees counter-clockwise

doc.save(doc.name, incremental=True) # Update the file

doc.close()

[This script opens a PDF file, rotates a specific page by 90 degrees counter-clockwise, and saves the changes](https://www.johndcook.com/blog/2015/05/01/rotating-pdf-pages-with-python/)[2](https://stackoverflow.com/questions/64834321/how-can-i-arbitarily-rotate-rearrange-etc-pdf-pages-in-python).

Please replace 'original.pdf', 'mypdf.pdf', and n with your actual PDF file name and the page number you want to rotate. Make sure to install the necessary libraries using pip:

pip install PyPDF2

pip install PyMuPDF

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

[In Python’s python-docx module, a Paragraph object and a Run object are used to manipulate Word documents1](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/).

* A **Paragraph** is one of the primary building blocks of a document. [Each paragraph contains one or more Run objects1](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/). [A paragraph in Word is a bit like a paragraph in English: it’s a distinct section of a document, separated by a blank line or a new line1](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/).
* [A **Run** object is a contiguous run of text with the same style](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/)[2](https://stackoverflow.com/questions/69999702/how-does-runs-works-in-python-docx). [A single Paragraph can contain multiple Run objects with different formatting](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/)[3](https://www.transtutors.com/questions/what-is-the-difference-between-a-paragraph-object-and-a-run-object-how-do-you-obtain-6702302.htm). [For example, if you have a sentence where one word is bold and the rest of the text is not, that sentence would contain two Run objects: one for the bold word, and one for the rest of the sentence](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/)[2](https://stackoverflow.com/questions/69999702/how-does-runs-works-in-python-docx).

Here’s an example to illustrate this:

from docx import Document

doc = Document()

para = doc.add\_paragraph()

run1 = para.add\_run('This is normal text. ')

run2 = para.add\_run('This is bold text.')

run2.bold = True

doc.save('test.docx')

In this example, the paragraph contains two runs. The first run has the text "This is normal text. " and the second run has the text “This is bold text.” which is formatted as bold.

[Remember, the Run objects are contained within Paragraph objects, and they are used to apply character-level formatting, such as font name and size, bold, italic, etc](https://www.geeksforgeeks.org/working-with-paragraphs-in-python-docx-module/)[3](https://www.transtutors.com/questions/what-is-the-difference-between-a-paragraph-object-and-a-run-object-how-do-you-obtain-6702302.htm).

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

In Python, if you’re using the python-docx module, you can obtain a list of Paragraph objects from a Document object as follows:

from docx import Document

# Load the document

doc = Document('your\_document.docx')

# Get the list of paragraph objects

paragraphs = doc.paragraphs

# Now, 'paragraphs' is a list of Paragraph objects

for para in paragraphs:

print(para.text)

In this code, replace 'your\_document.docx' with the path to your actual Word document. [The doc.paragraphs attribute is a list of Paragraph objects representing the paragraphs in the document1](https://stackoverflow.com/questions/66981100/how-to-extract-a-range-of-paragraphs-from-an-existing-docx-file-using-python-doc). [Each Paragraph object has a .text attribute that contains the text of the paragraph](https://stackoverflow.com/questions/66981100/how-to-extract-a-range-of-paragraphs-from-an-existing-docx-file-using-python-doc)[2](https://stackoverflow.com/questions/62307174/how-to-extract-text-from-paragraphs-and-table-using-python-module-from-word-docu).

Please note that you need to have the python-docx module installed. If it’s not installed, you can install it using pip:

pip install python-docx

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

[In Python, when dealing with Word documents using the python-docx module, a Run object has bold, underline, italic, strike, and outline variables1](https://www.solutioninn.com/study-help/automate-the-boring-stuff-with-python/what-type-of-object-has-bold-underline-italic-strike-and-1116592). [These variables are used to apply character-level formatting to the text within the Run2](https://www.transtutors.com/questions/what-type-of-object-has-bold-underline-italic-strike-and-outline-variables-what-is-t-6702303.htm).

Here’s an example:

from docx import Document

doc = Document()

para = doc.add\_paragraph()

run = para.add\_run('This is some text.')

run.bold = True

run.italic = True

run.underline = True

# Note: python-docx does not support strike or outline directly

doc.save('formatted.docx')

In this code, a Run object is created with the text “This is some text.” The bold, italic, and underline properties of the Run object are set to True, which applies the corresponding formatting to the text.

Please note that python-docx does not directly support strike (strikethrough) or outline formatting. [For more advanced formatting, you might need to use other libraries or work directly with the underlying XML](https://www.solutioninn.com/study-help/automate-the-boring-stuff-with-python/what-type-of-object-has-bold-underline-italic-strike-and-1116592)[3](https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/font-object-bold-italic-size-strikethrough-underline-weight-properties-example).

Remember to install the necessary library using pip if it’s not already installed:

pip install python-docx

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

[In Python, when using the python-docx module to manipulate Word documents, the bold attribute of a Run object can be set to True, False, or None1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012).

* [True: Setting the bold attribute to True will make the text within the Run object bold1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012).
* [False: Setting it to False will make the text within the Run object not bold1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012).
* [None: Setting it to None will make the Run object use the bold setting of the style that’s currently applied to it1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012). [This means if the style of the text is set to bold, it will remain bold; if it’s not bold, it will remain not bold1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012).

Here’s an example:

from docx import Document

doc = Document()

para = doc.add\_paragraph()

run1 = para.add\_run('This is normal text. ')

run1.bold = False

run2 = para.add\_run('This is bold text. ')

run2.bold = True

run3 = para.add\_run('This text follows the style\'s bold setting.')

run3.bold = None

doc.save('test.docx')

In this example, the first run will not be bold, the second run will be bold, and the third run will follow the bold setting of the current style.

[Remember, the bold attribute only affects the Run object it’s applied to, and not any other Run objects in the same Paragraph1](https://www.transtutors.com/questions/what-is-the-difference-between-setting-the-bold-variable-to-true-false-or-none-how-d-9283956.htm)[2](https://github.com/Vivektumulu/python-assignment1/blob/main/assignment%2012)

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

In Python, you can use the python-docx module to create a Document object for a new Word document. Here’s how you can do it:

from docx import Document

# Create a new Document object

doc = Document()

# Now you can add paragraphs, headings, etc. to 'doc'

para = doc.add\_paragraph('Hello, World!')

# Save the document

doc.save('new\_document.docx')

In this code, a new Document object is created. A paragraph with the text “Hello, World!” is added to the document. [Finally, the document is saved as 'new\_document.docx’1](https://www.geeksforgeeks.org/python-working-with-docx-module/).

Please replace 'new\_document.docx' with your desired file name and path. Make sure to install the necessary library using pip if it’s not already installed:

pip install python-docx

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

In Python, you can use the python-docx module to add a paragraph to a Document object. Here’s how you can do it:

from docx import Document

# Assume 'doc' is a Document object

# doc = Document()

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

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

# Now 'para' is a Paragraph object containing the text 'Hello, there!'

In this code, a new paragraph with the text “Hello, there!” is added to the Document object stored in the variable doc. The add\_paragraph method returns a Paragraph object that you can use to further manipulate the paragraph.

Remember to install the necessary library using pip if it’s not already installed:

pip install python-docx

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

[In Python, when using the python-docx module to work with Word documents, you can add headings of different levels to a document using the add\_heading method1](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/). [The levels of headings are represented by integers from 0 to 91](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/).

Here’s what each level represents:

* [0: Title of the document1](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/).
* [1 to 9: Heading levels 1 to 91](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/). [The size of the heading decreases as the level increases1](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/).

For example, you can add a level 1 heading like this:

from docx import Document

doc = Document()

doc.add\_heading('This is a level 1 heading', 1)

[In this code, ‘This is a level 1 heading’ is added as a level 1 heading to the document1](https://www.geeksforgeeks.org/working-with-titles-and-heading-python-docx-module/). Remember to replace 'This is a level 1 heading' with your actual heading text.

Please note that you need to have the python-docx module installed. If it’s not installed, you can install it using pip:

pip install python-docx