In [2]:
```
pip install PyPDF2
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting PyPDF2
  Obtaining dependency information for PyPDF2 from https://files.pythonhoste
d.org/packages/8e/5e/c86a5643653825d3c913719e788e41386bee415c2b87b4f955432f2
de6b2/pypdf2-3.0.1-py3-none-any.whl.metadata (https://files.pythonhosted.or
g/packages/8e/5e/c86a5643653825d3c913719e788e41386bee415c2b87b4f955432f2de6b
2/pypdf2-3.0.1-py3-none-any.whl.metadata)
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Using cached pypdf2-3.0.1-py3-none-any.whl (232 kB)
Installing collected packages: PyPDF2
Successfully installed PyPDF2-3.0.1
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
#1) Extract metaData

import PyPDF2

def extract_metadata(pdf_path):
    """Extract metadata from a PDF file."""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        metadata = reader.metadata
        for key, value in metadata.items():
            print(f"{key}: {value}")

# Example usage
extract_metadata("python.pdf")
```

```
/Author: Guido van Rossum, and the Python development team
/CreationDate: D:20180902004610Z
/Creator: LaTeX with hyperref package
/ModDate: D:20180902111920-07'00'
/Producer: xdvipdfmx (0.7.9)
/Title: Python Tutorial
```

In [7]:
```python
#2) Extract text
import PyPDF2

def extract_text(pdf_path):
    """Extract text from a PDF."""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        text = ""
        for page in reader.pages:
            text += page.extract_text() + "\n"
        return text

# Example usage
print(extract_text("python1.pdf"))
```

Essay on Python
Introduction to Python
Python is one of the most widely used programming languages today. It was cr
eated by Guido van
Rossum and first released in 1991. Inspired by the ABC programming language,
Python was
designed to be easy to read and simple to implement. Van Rossum wanted to ma
ke a language that
removed the complexities found in traditional programming languages, enablin
g developers to focus
on solving problems rather than fighting with syntax.
Python follows the philosophy of "there should be one - and preferably only
one - obvious way to do
it," as stated in the Zen of Python. This philosophy emphasizes readability
and simplicity, which
have been crucial to Python's global success. Over time, Python has develope
d into a language with
a massive ecosystem of libraries, frameworks, and tools, supporting a wide r
ange of industries from
web development to artificial intelligence.
Features and Applications of Python
Python stands out because of several key features. First, it is an interpret
ed language, meaning it
does not require compilation before execution. It supports dynamic typing, a
utomatic memory
management, and a vast standard library that makes it easy to write efficien
t and powerful programs
quickly. Python also encourages a programming style known as object-oriented
programming, but it
supports multiple paradigms including procedural and functional programming.
The applications of Python are almost endless:
- Web Development: Frameworks like Django and Flask have made backend develo
pment faster
and more efficient.
- Data Science and Machine Learning: Libraries such as Pandas, NumPy, Tensor
Flow, and
Essay on Python
scikit-learn are essential tools for data analysis, machine learning, and ar
tificial intelligence.
- Automation and Scripting: Python's simplicity makes it perfect for writing
small scripts to automate
repetitive tasks.
- Game Development: Libraries like Pygame allow developers to build simple g
ames and
prototypes.
- Embedded Systems and IoT: MicroPython and CircuitPython enable programming
on
microcontrollers.
- Education: Due to its simplicity, Python is often the first language taugh
t in schools and universities
worldwide.
Importance of Python Today
Today, Python is considered one of the most important languages in the techn
ology world.
According to the TIOBE Index and other programming language popularity ranki
ngs, Python has
consistently been among the top languages for several years. Its versatility
makes it a favorite
choice for startups, tech giants, researchers, and hobbyists alike.
Python's role in emerging technologies like artificial intelligence, machine
learning, and big data is

```
particularly noteworthy. Industries like healthcare, finance, and even aeros
pace are leveraging
Python to drive innovation. Moreover, the community around Python is large a
nd active, meaning
there is abundant support for learners and professionals alike.
In conclusion, Python's combination of simplicity, power, and flexibility ha
s made it a cornerstone of
modern software development. As technology continues to evolve, Python is li
kely to maintain, if not
expand, its influence in the coming decades. Its future remains bright, driv
en by a strong community,
Essay on Python
continued innovation, and an ever-growing range of applications.
```

In [10]:
```python
#merging
import PyPDF2

def merge_pdfs(pdf_list, output_path):
    """Merge multiple PDFs into one."""
    merger = PyPDF2.PdfMerger()
    for pdf in pdf_list:
        merger.append(pdf)
    merger.write(output_path)
    merger.close()

# Example usage
merge_pdfs(["python.pdf", "python1.pdf"], "merged.pdf")
print("merged successfully")
```

```
merged successfully
```

In [11]:
```python
#3) search text
import PyPDF2

def search_text(pdf_path, keyword):
    """Search for a keyword in a PDF."""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for page_num, page in enumerate(reader.pages):
            text = page.extract_text()
            if keyword.lower() in text.lower():
                print(f"Keyword found on Page {page_num + 1}")

# Example usage
search_text("python1.pdf", "innovation")
```

```
Keyword found on Page 2
Keyword found on Page 3
```

In [1]:
```
pip install pdf2docx
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Requirement already satisfied: pdf2docx in c:\users\priya\appdata\roaming\py
thon\python311\site-packages (0.5.8)
Requirement already satisfied: PyMuPDF>=1.19.0 in c:\users\priya\appdata\roa
ming\python\python311\site-packages (from pdf2docx) (1.25.5)
Requirement already satisfied: python-docx>=0.8.10 in c:\users\priya\appdata
\roaming\python\python311\site-packages (from pdf2docx) (1.1.2)
Requirement already satisfied: fonttools>=4.24.0 in c:\programdata\anaconda3
\lib\site-packages (from pdf2docx) (4.25.0)
Requirement already satisfied: numpy>=1.17.2 in c:\programdata\anaconda3\lib
\site-packages (from pdf2docx) (1.24.3)
Requirement already satisfied: opencv-python-headless>=4.5 in c:\users\priya
\appdata\roaming\python\python311\site-packages (from pdf2docx) (4.11.0.86)
Requirement already satisfied: fire>=0.3.0 in c:\users\priya\appdata\roaming
\python\python311\site-packages (from pdf2docx) (0.7.0)
Requirement already satisfied: termcolor in c:\users\priya\appdata\roaming\p
ython\python311\site-packages (from fire>=0.3.0->pdf2docx) (3.0.1)
Requirement already satisfied: lxml>=3.1.0 in c:\programdata\anaconda3\lib\s
ite-packages (from python-docx>=0.8.10->pdf2docx) (4.9.3)
Requirement already satisfied: typing-extensions>=4.9.0 in c:\users\priya\ap
pdata\roaming\python\python311\site-packages (from python-docx>=0.8.10->pdf2
docx) (4.13.2)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
#4) pdf to document
from pdf2docx import Converter

def pdf_to_word(pdf_path, word_path):
    """Convert a PDF to a Word document."""
    cv = Converter(pdf_path)
    cv.convert(word_path, start=0, end=None)
    cv.close()

# Example usage
pdf_to_word("python1.pdf", "output.docx")
print("sucessfully converted")
```

```
[INFO] Start to convert python1.pdf
[INFO] [1/4] Opening document...
[INFO] [2/4] Analyzing document...
[INFO] [3/4] Parsing pages...
[INFO] (1/3) Page 1
[INFO] (2/3) Page 2
[INFO] (3/3) Page 3
[INFO] [4/4] Creating pages...
[INFO] (1/3) Page 1
[INFO] (2/3) Page 2
[INFO] (3/3) Page 3
[INFO] Terminated in 0.46s.

sucessfully converted
```

In [6]:
```python
#5) sign digitally
from PyPDF2 import PdfReader, PdfWriter
from PyPDF2.generic import NameObject, TextStringObject

def sign_pdf(pdf_path, output_path, signature_text):
    """Add a digital signature to a PDF."""
    reader = PdfReader(pdf_path)
    writer = PdfWriter()

    for page in reader.pages:
        writer.add_page(page)

    writer.add_metadata({
        NameObject("/Signature"): TextStringObject(signature_text)
    })

    with open(output_path, "wb") as output_file:
        writer.write(output_file)

# Example usage
sign_pdf("python1.pdf", "signed.pdf", "Signed by Priyanka")
print("signed successfully")
```

```
signed successfully
```

In [8]:
```python
#6) split into seperate pages
import PyPDF2

def split_pdf(pdf_path):
    """Split a PDF into individual pages."""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for i, page in enumerate(reader.pages):
            writer = PyPDF2.PdfWriter()
            writer.add_page(page)
            with open(f"page_{i+1}.pdf", 'wb') as output_file:
                writer.write(output_file)

# Example usage
split_pdf("python1.pdf")
print("successfully splitted")
```

```
successfully splitted
```

In [9]:
```
#9) extract pages
import PyPDF2

def extract_pages(pdf_path, pages, output_path):
    """Extract specific pages from a PDF."""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        writer = PyPDF2.PdfWriter()
        for page_num in pages:
            writer.add_page(reader.pages[page_num - 1])
        with open(output_path, 'wb') as output_file:
            writer.write(output_file)

# Example usage
extract_pages("python.pdf", [1, 3, 5], "extracted.pdf")
print("extracted sucessfully")
```

```
extracted sucessfully
```

In [13]:
```
pip install reportlab
```

```
Defaulting to user installation because normal site-packages is not writeabl
e
Collecting reportlab
  Obtaining dependency information for reportlab from https://files.pythonho
sted.org/packages/52/15/4702e132ae36beb8daf3e20a92f166451148c4a89650cc9d3f19
b3c66714/reportlab-4.4.0-py3-none-any.whl.metadata (https://files.pythonhost
ed.org/packages/52/15/4702e132ae36beb8daf3e20a92f166451148c4a89650cc9d3f19b3
c66714/reportlab-4.4.0-py3-none-any.whl.metadata)
  Downloading reportlab-4.4.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pillow>=9.0.0 in c:\programdata\anaconda3\lib
\site-packages (from reportlab) (9.4.0)
Requirement already satisfied: chardet in c:\programdata\anaconda3\lib\site-
packages (from reportlab) (4.0.0)
Downloading reportlab-4.4.0-py3-none-any.whl (2.0 MB)
   ---------------------------------------- 0.0/2.0 MB ? eta -:--:--
   ---------------------------------------- 0.0/2.0 MB ? eta -:--:--
   ---------------------------------------- 0.0/2.0 MB ? eta -:--:--
    --------------------------------------- 0.0/2.0 MB 262.6 kB/s eta 0:00:0
8
   -- ------------------------------------- 0.1/2.0 MB 731.4 kB/s eta 0:00:0
3
   ---------- ----------------------------- 0.6/2.0 MB 3.3 MB/s eta 0:00:01
   ------------------------ -------------- 1.2/2.0 MB 5.3 MB/s eta 0:00:01
   ------------------------------ -------- 1.5/2.0 MB 5.4 MB/s eta 0:00:01
   ---------------------------------- --- 1.8/2.0 MB 5.4 MB/s eta 0:00:01
   ---------------------------------------- 1.9/2.0 MB 5.0 MB/s eta 0:00:01
   ---------------------------------------- 2.0/2.0 MB 4.8 MB/s eta 0:00:00
Installing collected packages: reportlab
Successfully installed reportlab-4.4.0
Note: you may need to restart the kernel to use updated packages.
```

In [15]:
```python
#10) add watermark
from PyPDF2 import PdfReader, PdfWriter
from reportlab.pdfgen import canvas

def create_watermark(text, watermark_pdf):
    """Create a watermark PDF with text."""
    c = canvas.Canvas(watermark_pdf)
    c.setFont("Helvetica", 20)
    c.drawString(200, 500, text)  # Adjust position as needed
    c.save()

def add_watermark(input_pdf, output_pdf, watermark_text):
    """Add a watermark text to all pages."""
    watermark_pdf = "temp_watermark.pdf"
    create_watermark(watermark_text, watermark_pdf)

    reader = PdfReader(input_pdf)
    watermark_reader = PdfReader(watermark_pdf)
    watermark_page = watermark_reader.pages[0]
    writer = PdfWriter()

    for page in reader.pages:
        page.merge_page(watermark_page)  # Overlay watermark
        writer.add_page(page)

    with open(output_pdf, 'wb') as output_file:
        writer.write(output_file)

# Example Usage
add_watermark("python1.pdf", "output.pdf", "Confidential")
print("sucessfully added")
```

```
sucessfully added
```

In [ ]:
```python

```