



Python's Future in Clinical Trials: Innovations and Collaborative Advancements

Presentation Number: OS03

Presenter Name: Sohan Lal

Email: sohan.l225@geninvo.com

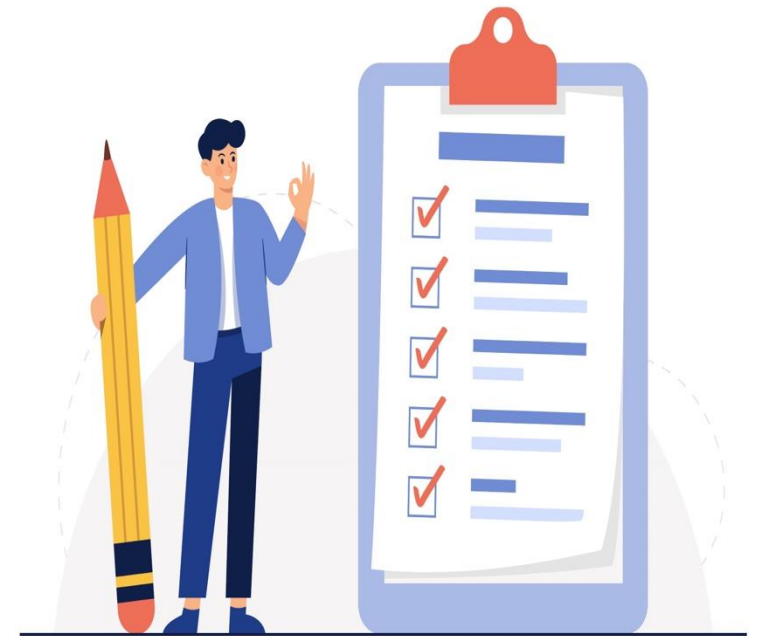
www.geninvo.com

North America | UK | Europe | India



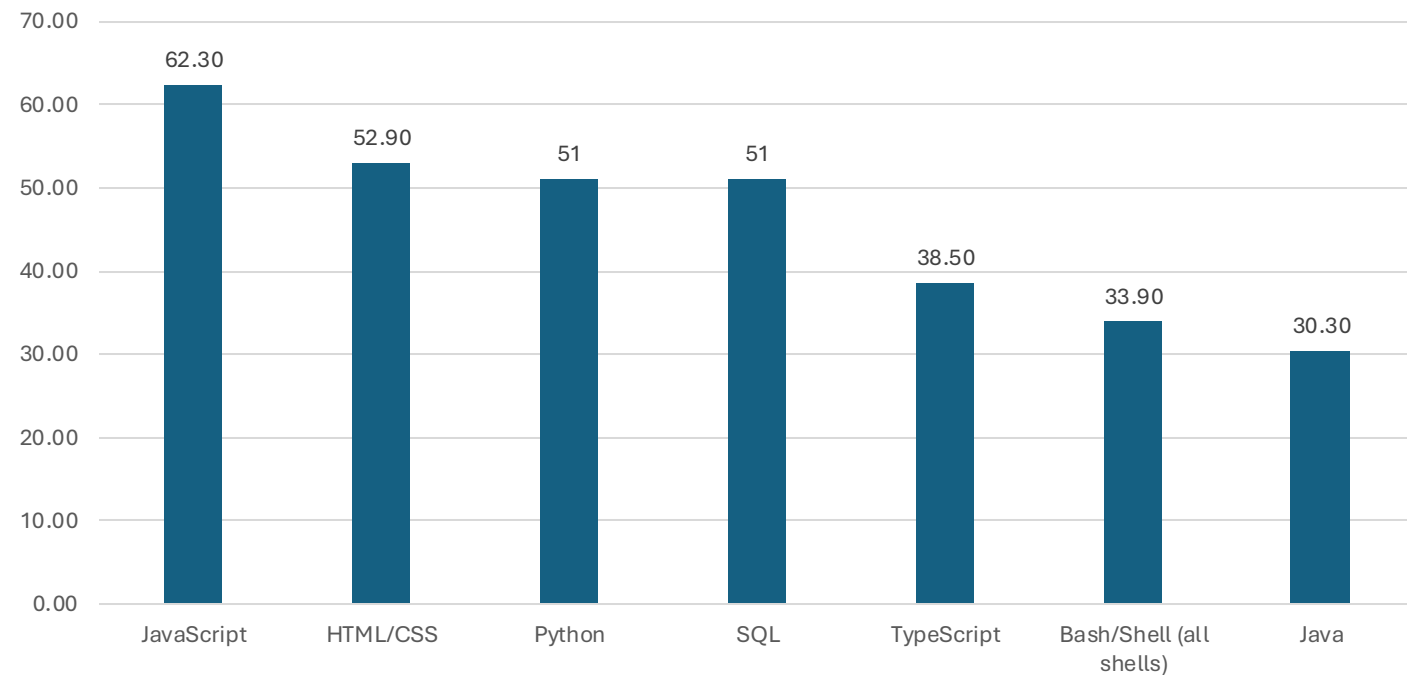
Agenda

- **Trend of Programming Language?**
- **Introduction to Python?**
- **PDFs handling in Python.**
- **PDFs Data Extraction Example**
- **Docx/Doc handling in Python.**
- **Python-Docx Data Extraction Example**
- **NLP for pharma using python**
- **Synthetic Data Generation**



What are the most popular programming languages now?

Most used programming languages among developers worldwide as of 2024



www.geninvo.com
North America | UK | Europe | India

What is Python?



P I V O T

Powerful

Intuitive

Versatile

Optimized

Thriving
Community



www.geninvo.com
North America | UK | Europe | India

Python Popular Libraries for Pharma

Pandas

Numpy

SASPy

PymuPDF2 / Python-
Docx

PyDICOM

NLTK

faker

Spacy

Bert/Mbert



www.geninvo.com
North America | UK | Europe | India

What is Panda?



C L E A R

Complete

Large Scale

Efficient

Analytical

Ready to
use



www.geninvo.com
North America | UK | Europe | India

NLP in python for Pharma

- Python allows seamless integration with clinical data management systems (CDMS), enabling automated document data extraction, analysis, and reporting of clinical data.
- NLP for pharma literature mining, NLP for drug discovery, NLP for drug named entity recognition, and NLP and AI for strategic clinical trial design.
- NLP helps in extracting meaningful insights from these documents, improving efficiency and accuracy in healthcare research

Components of NLP

- Semantic Analysis: Deals with **how** words are arranged in a sentence to ensure grammatical correctness.
- Syntactic Analysis: Deals with **what** the words mean, i.e., the meaning and interpretation of the sentence



www.geninvo.com
North America | UK | Europe | India

Powerful NLP Libraries

Powerful NLP Libraries

- **SpaCy:** Fast and efficient library for industrial-strength NLP tasks, such as tokenization, named entity recognition (NER), and part-of-speech tagging.
- **NLTK:** A comprehensive library with tools for text processing, stemming, and more, suited for research and experimentation.
- **Transformers (Hugging Face):** Provide cutting-edge pre-trained models for NLP models, allowing for the extraction of complex insights from text documents which including specialized versions like **BERT (Bidirectional Encoder Representations from Transformers)**, **mBERT (Multilingual BERT)** and **CT-BERT** for healthcare and clinical applications. These models can be easily integrated into Python workflows, providing powerful capabilities for extracting insights from clinical texts.



Leveraging PyMuPDF For PDF Handling in Python

PyMuPDF (also known as Fitz) is a powerful Python library used for extracting text, images, and metadata from PDF documents. It can then transform the text into various desired formats, such as JSON, CSV, Excel, plain text, HTML, or XML.

Key Features of PyMuPDF

- **Text Extraction:** supports extracting structured text, including tables or lists.
- **Extracting specific elements:** Extract images embedded within PDFs and provides access to document metadata (e.g., title, author, creation date, etc.)
- **Advanced Search and Pattern Recognition:** Allows for searching for specific terms, phrases, or patterns within PDF content.



www.geninvo.com
North America | UK | Europe | India

Examples of PDF output

```
import fitz # PyMuPDF

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

# Extract text from the first page
page = doc.load_page(0) # 0-based index for the first page
text = page.get_text("text")

print(text)
```

```
# Open the PDF file
doc = fitz.open("clinical_trial_report.pdf")

# Extract text from the first page
page = doc.load_page(0) # 0-based index for the first page
text_blocks = page.get_text("blocks") # Extract text in blocks
```

```
[In [366]: text
Out[366]: 'ALEPRO trial \nENGOT-0V70/BGOG-0V70 \n
\nVersion 1.0 (8 Feb 2023) \n \n \nPROTOCOL SYNOPSIS \n
\nTitle of clinical Trial («Trial») \nPhase II, open
label, multicenter study of abemaciclib and letrozole in
\nEstrogen receptor positive rare ovarian cancer
\nProtocol \nShort \nTitle \nAcronym \nALEPRO \nTrial
Phase (I, II, III, IV) \n2 \nSponsor name \nUniversity
Hospitals Leuven (UZ Leuven) \nCoordinating Investigator
\nEls Van Nieuwenhuysen \n Contact Address CI
\nHerestraat 49, 3000 Leuven \n Contact Email CI
\nEls.vannieuwenhuysen@uzleuven.be \n Contact
Phone CI \n0032/16342531 \nEudraCT number
```

```
~',
(76.58399963378906,
142.09996032714844,
518.1669311523438,
165.49998474121094,
'Title of clinical Trial («Trial») \nPhase II, open
abel, multicenter study of abemaciclib and letrozole in
\nEstrogen receptor positive rare ovarian cancer \n',
3,
0),
(76.58399963378906,
173.0599822998047,
201.63734436035156,
183.61997985839844,
'Protocol \nShort \nTitle \n',
4,
0),
```



www.geninvo.com
North America | UK | Europe | India

Leveraging python-docx For docx Handling in Python

Python-docx is a powerful Python library used for creating and updating Microsoft Word (.docx) files. It enables users to automate document creation, manipulation, and metadata extraction, all while providing flexibility in text formatting, tables, images, and more.

Key Features of python-docx :

- Automate the generation of clinical study reports (CSRs), safety summaries, and patient reports.
- Extract metadata to ensure compliance with regulatory requirements and track document versions.
- Use NLP with python-docx to extract key information from clinical trial reports, ensuring better data management and analysis



www.geninvo.com
North America | UK | Europe | India

Examples of Python-Docx output

```
from docx import Document

# Function to read and extract text from a .docx file
def read_docx(file_path): # Load the .docx file
    doc = Document(file_path)
    text = ""
    for para in doc.paragraphs:
        text += para.text + "\n" # Add a newline after each paragraph

    return text

file_path = 'SAP.docx' # Replace with your file path
doc_text = read_docx(file_path)

print(doc_text)
```

```
from docx import Document

# Function to read tables from a .docx file
def read_docx_tables(file_path):
    doc = Document(file_path)
    tables_data = []
    for table in doc.tables:
        table_data = [] # Loop through each row in the table
        for row in table.rows:
            row_data = [] # Loop through each cell in the row
            for cell in row.cells:
                row_data.append(cell.text.strip())
            table_data.append(row_data)
        tables_data.append(table_data)
    return tables_data

tables = read_docx_tables(file_path)
print(tables)
```

```
In [12]: print(doc_text)
title page      1
TABLE OF CONTENTS      2
List Of Abbreviations      4
1.      INTRODUCTION      5
2.      OBJECTIVES      5
3.      STUDY OVERVIEW      5
3.1     Study Design      5
3.2     Sample Size      7
3.3     Randomization and Unblinding Procedures      8
4.      STUDY ENDPOINTS/Outcomes      8
5.      HYPOTHESES TESTING      9
6.      ANALYSIS SUBSETS      10
6.1     Safety Population      10
6.2     Modified Intent to Treat Population (mITT Population)      10
```

```
[['Abbreviation or special term', 'Explanation'],
 ['AE', 'Adverse Event'],
 ['ANOVA', 'Analysis of Variance'],
 ['BMI', 'Body Mass Index'],
 ['CFR', 'Code of Federal Regulations'],
 ['eCRF', 'Electronic Case Report Form'],
 ['CRO', 'Contract Research Organization'],
 ['DCF', 'Data Clarification Forms'],
 ['EC', 'Ethics Committee'],
 ['ED', 'Early Discontinuation'],
 ['EDC', 'Electronic Data Capture'],
 ['EOT', 'End of Treatment'],
 ['FDA', 'Food and Drug Administration'],
 ['GCP', 'Good Clinical Practice'],
 ['hCG', 'Human Chorionic Gonadotropin'],
 ['HEENT', 'Head, Eyes, Ears, Nose and Throat'],
```



www.geninvo.com
North America | UK | Europe | India

Examples of Python-Docx output1

CSR File : CSR Final_1.0_13oct2021(headings)_updated_table.docx

(67/4,115 [1.6%] subjects), and “time window deviations” (23/4,115 [0.6%] subjects).

Protocol deviations and COVID-19 protocol deviations are summarized in Table 11 and Table 14.1.1.5, respectively.

Table 11 Protocol Deviations (Safety Population)

Type	XYZ1234	Placebo	Total
Category [n (%)]	(N=3,082)	(N=1,033)	(N=4,115)
Any Important Protocol Deviation	1,837 (59.6)	591 (57.2)	2,428 (59.0)
Any Major Protocol Deviation	157 (5.1)	48 (4.6)	205 (5.0)
03 – Investigational Product	63 (2.0)	22 (2.1)	85 (2.1)
06-Visit Window	53 (1.7)	16 (1.5)	69 (1.7)
05-Endpoint Data	28 (0.9)	11 (1.1)	39 (0.9)
09-Prohibited Co-Medication	11 (0.4)	3 (0.3)	14 (0.3)
02-Exclusion Criteria	2 (0.1)	0	2 (0.0)
04-Assignment Safety	2 (0.1)	0	2 (0.0)

QC Rule

Source to CSRSource TableCSR to Source

Select Table *

Table 14.1.1.4 Protocol Deviations (Safety Population)

Table 14.1.1.4 Protocol Deviations (Safety Population)

Type Category [n (%)]	XYZ1234	Placebo	Total
(No Match)	(N=3082)	(N=1033)	(N=4115)
Any Important PD (Match)	1837 (59.6) (Match)	591 (57.2) (Match)	2428 (59.0) (Match)
Any Major PD (Match)	157 (5.1) (Match)	48 (4.6) (Match)	205 (5.0) (Match)
Any Major PD 03- Investigational Product (Match)	63 (2.0) (Match)	22 (2.1) (Match)	85 (2.1) (Match)
Any Major PD 06- Visit Window (Match)	53 (1.7) (Match)	16 (1.5) (Match)	69 (1.7) (Match)

Synthetic Data Generation Using FAKER

Faker library in Python can be incredibly useful for generating synthetic, realistic-looking data that can be used in testing, simulations, or anonymization. Since clinical trials often involve sensitive patient information, using fake data instead of real-world data can help ensure privacy while still providing a robust testing environment

Use of Faker for Clinical Study Data

1. Generating Patient Information
2. Simulating Medical Conditions
3. Generating Clinical Trial Enrollment Data
4. Simulating Laboratory Results



www.geninvo.com
North America | UK | Europe | India

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

# Generate 100 random age values between 18 and 75
#np.random.seed()
ages = np.random.normal(loc=47, scale=9, size=200)
#ages = np.clip(ages, 18, 75)

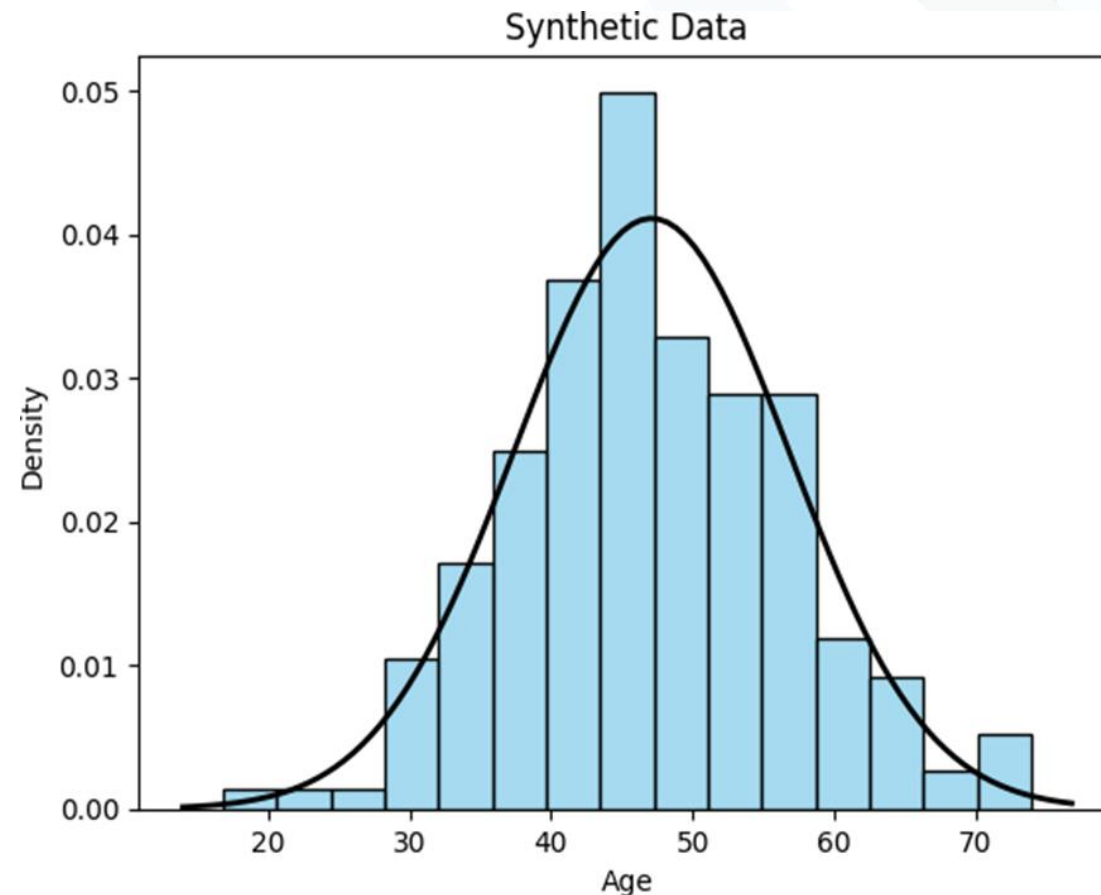
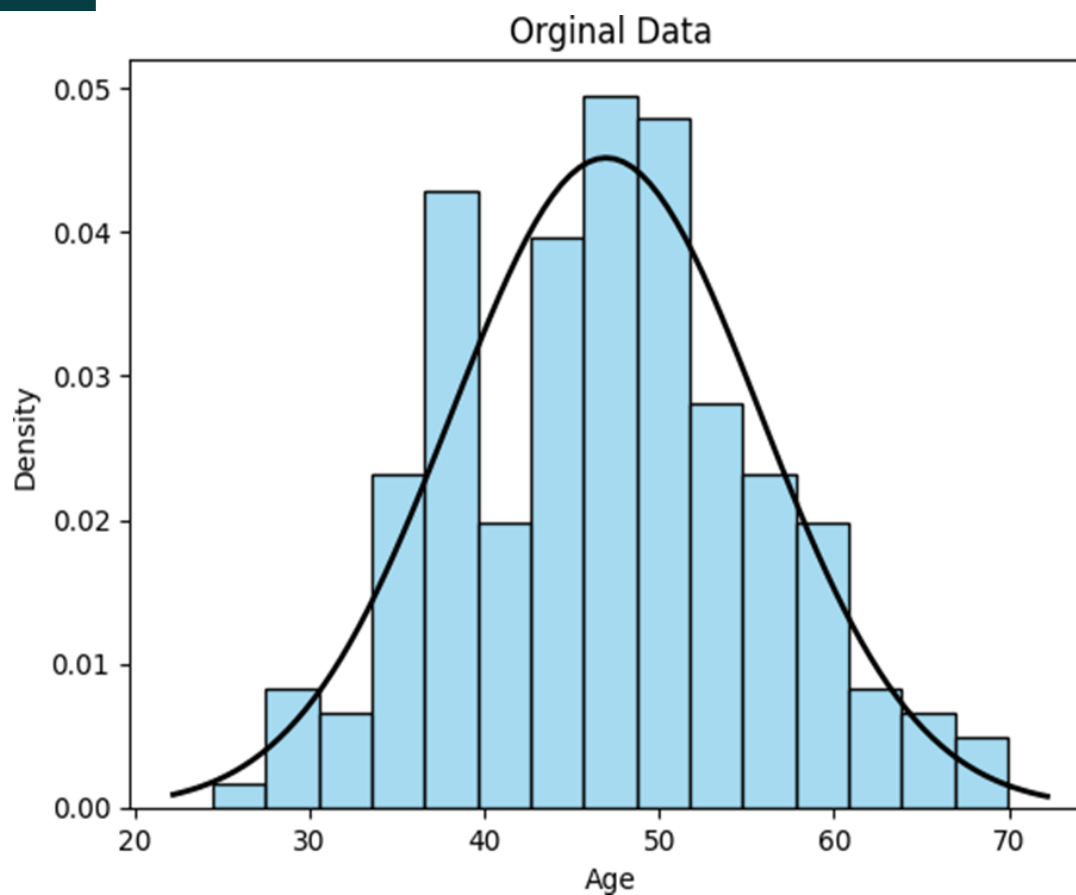
# Create the histogram
sns.histplot(ages, bins=15, kde=False, color='skyblue', stat='density')

# Add the normal distribution curve
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(ages), np.std(ages))
plt.plot(x, p, 'k', linewidth=2)

# Add labels and title
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Synthetic Data')

# Show the plot
plt.show()
```

Synthetic Data Generation Using FAKER



Questions & Answer



www.geninvo.com
North America | UK | Europe | India



References

- [How can natural language processing help model informed drug development?: a review – PMC](#)
- <https://support.sas.com/en/software/saspy.html>
- <https://phdservices.org/python-in-clinical-research/>
- [Synthetic data generation methods in healthcare: A review on open-source tools and methods - ScienceDirect](#)



www.geninvo.com
North America | UK | Europe | India



Ideas
Delivered
Intuitively

Thank You!

+1-833-GEN-INVO (436-4686) | info@geninvo.com | www.geninvo.com

