Practical-10

March 26, 2025

0.1 Practical 10 :-

Name: Sarthak Sanay
Enrollment No: 230031101611051

0.1.1 Problem Statement 1:-

Text Analysis and Word Counting:-
You are tasked with analyzing a given text document and generating a report containing the word count for each unique word. Additionally, you need to identify the most frequent word in the document. Write a Python function `analyze_text(text)` that takes a text document as input and returns a dictionary containing the word count for each unique word, as well as the most frequent word and its count.

[13]:
```python
def analyze_text(text):
    document = ""
    for char in text:
        if char.isalnum() or char.isspace():
            document += char

    words = document.split()
    unique_word = {}
    for word in words:
        if word in unique_word:
            unique_word[word] += 1
        else:
            unique_word[word] = 1

    most_frequent_word = ""
    max_count = 0
    for word, count in unique_word.items():
        if count > max_count:
            most_frequent_word = word
            max_count = count
    print("Most frequent word is:", most_frequent_word, "->", max_count)
    return unique_word

file = open("sample.txt", "r")
text = file.read()
text = text.lower()
file.close()

unique_word = analyze_text(text)
print("Word -> Count")
for key, value in unique_word.items():
    print(f"{key} -> {value}")
```

```
Most frequent word is: and -> 6
Word -> Count
python -> 3
is -> 1
an -> 1
interpreted -> 1
objectoriented -> 1
highlevel -> 2
programming -> 1
language -> 2
with -> 2
dynamic -> 3
semantics -> 1
its -> 1
built -> 1
in -> 2
data -> 1
structures -> 1
combined -> 1
typing -> 1
and -> 6
binding -> 1
make -> 1
it -> 1
very -> 1
attractive -> 1
for -> 3
rapid -> 1
application -> 1
development -> 1
as -> 3
well -> 1
use -> 1
a -> 1
scripting -> 1
or -> 2
glue -> 1
to -> 2
connect -> 1
```

```
existing -> 1
components -> 1
together -> 1
pythons -> 1
simple -> 1
easy -> 1
learn -> 1
syntax -> 1
emphasizes -> 1
readability -> 1
therefore -> 1
reduces -> 1
the -> 3
cost -> 1
of -> 1
program -> 2
maintenance -> 1
supports -> 1
modules -> 1
packages -> 1
which -> 1
encourages -> 1
modularity -> 1
code -> 1
reuse -> 1
interpreter -> 1
extensive -> 1
standard -> 1
library -> 1
are -> 1
available -> 1
source -> 1
binary -> 1
form -> 1
without -> 1
charge -> 1
all -> 1
major -> 1
platforms -> 1
can -> 1
be -> 1
freely -> 1
distributed -> 1
```

0.1.2 Problem Statement 2:-

Text Encryption and Decryption:-
You're tasked with creating a simple encryption and decryption tool for sensi-

tive messages. Write Python functions `encrypt_message(message, shift)` and `decrypt_message(encrypted_message, shift)` to encrypt and decrypt messages using the Caesar cipher technique.

[4]:
```
def encrypt_message(message, shift):
    cipher_text = ""

    for char in message:
        # to add blank space
        if char == ' ':
            cipher_text += char
            continue
        # for upper-case characters
        elif char.isupper():
            cipher_text += chr((ord(char) + shift - 65) % 26 + 65)
        # for lower-case characters
        elif char.islower():
            cipher_text += chr((ord(char) + shift - 97) % 26 + 97)
        # special characters remains unchanged
        else:
            cipher_text += char

    return cipher_text


def decrypt_message(encrypted_message, shift):
    plain_text = ""

    for char in encrypted_message:
        # to add blank space
        if char == ' ':
            plain_text += char
            continue
        # for upper-case characters
        elif char.isupper():
            plain_text += chr((ord(char) - shift - 65) % 26 + 65)
        # for lower-case characters
        elif char.islower():
            plain_text += chr((ord(char) - shift - 97) % 26 + 97)
        # special characters remains unchanged
        else:
            plain_text += char

    return plain_text


print("Text Encryption and Decryption:-")
print("Ciphertext:", encrypt_message("Sarthak", 3))
print("Plaintext:", decrypt_message("Khoor", 3))
```

```
Text Encryption and Decryption:-
Ciphertext: Vduwkdn
Plaintext: Hello
```

0.1.3 Problem Statement 3:-

Unique Email Addresses:-
You're given a list of email addresses where each address consists of a local name and a domain name separated by the '@' symbol. You need to determine the number of unique email domains.

[7]:
```
def unique_domains(emails):
    unique = set() # initalize empty set
    for email in emails:
        domain = email.split("@")[1]
        unique.add(domain)
    return unique

emails = ["hello@gmail.com", "support@hp.com", "sarthak@yahoo.com",␣
 ↪"sarthak@outlook.com", "arnav@gmail.com", "sales@hp.com", "vaibhav@yahoo.
 ↪com"]
print(unique_domains(emails))
print("Count of Unique Domains:", len(unique_domains(emails)))
```

```
{'gmail.com', 'hp.com', 'yahoo.com', 'outlook.com'}
Count of Unique Domains: 4
```