ASSIGNMENT - 5.5

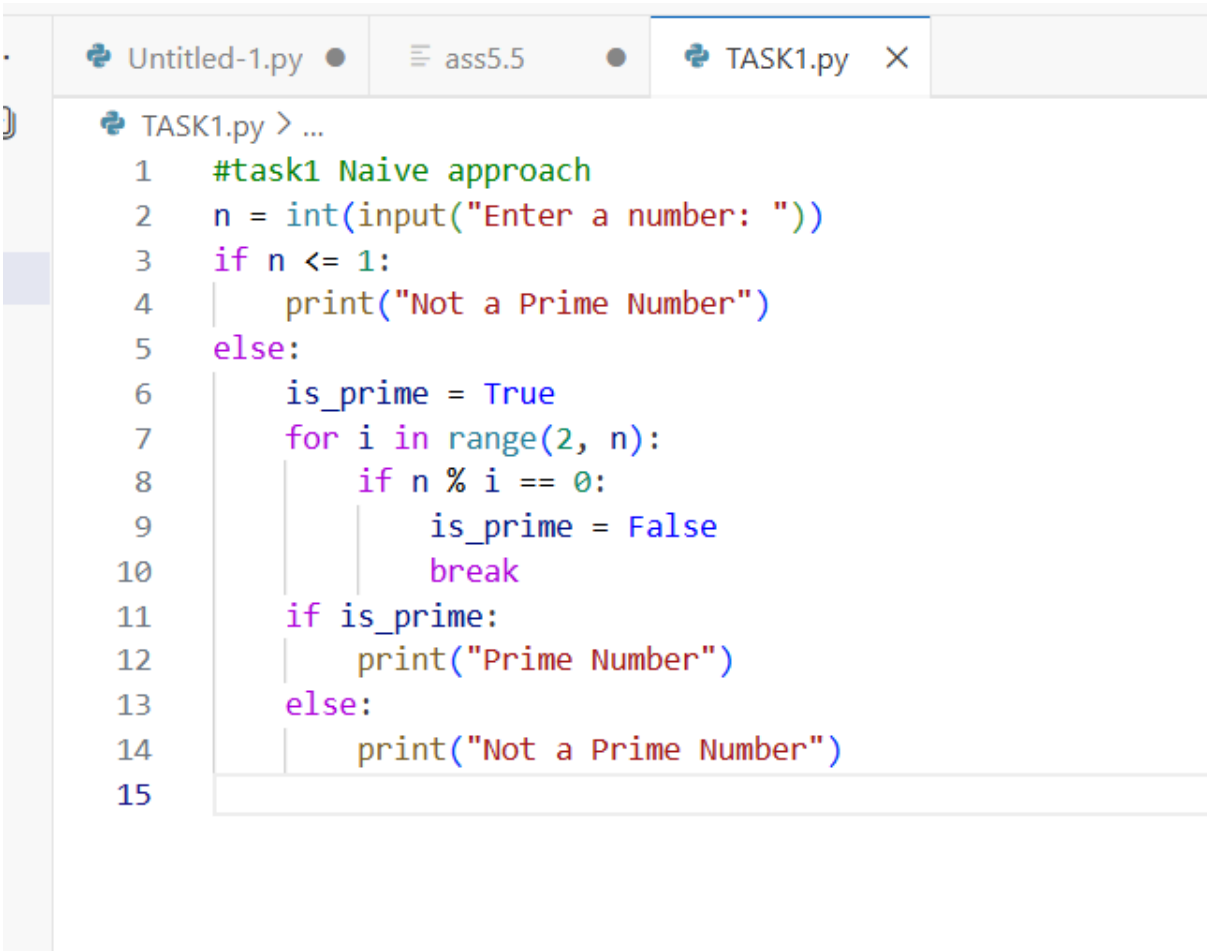
HT.NO: 2303A510I4

BATCH: 30

TASK 1

Transparency in Algorithm Optimization Prompt: Generate Python code for two prime-checking methods and explain how the optimized version improves performance.

CODE: Naive approach(basic)



OUTPUT:

CODE:

Optimized approach

task1.2.py > ...

```python

#optimised approach
n = int(input("Enter a number: "))
if n <= 1:
    print("Not a Prime Number")
else:
    is_prime = True
    for i in range(2, n):
        if n % i == 0:
            is_prime = False
            break
    if is_prime:
        print("Prime Number")
    else:
        print("Not a Prime Number")

```

OUTPUT:

```
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.exe c:/Users/ANUSHA/OneDrive/D
esktop/AI-CODING/task1.2.py
Enter a number: 63
Not a Prime Number
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

Naive Method This method checks almost every number to see if it divides the given value, so it takes more time as the number increases.

Optimized Method This method checks only the required divisors up to the square root, so it gives the result much faster.

TASK 2:

Transparency in Recursive Algorithms

Prompt

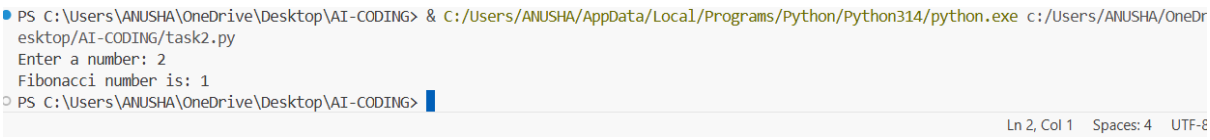
Generate a recursive Python function to calculate Fibonacci numbers. Add clear comments explaining how recursion works, including base cases and recursive calls.

Code:

task2.py > ...

```
1
2
3   def fibonacci(n):
4       if n == 0:
5           return 0
6       elif n == 1:
7           return 1
8       else:
9           return fibonacci(n-1) + fibonacci(n-2)
10  num = int(input("Enter a number: "))
11  print("Fibonacci number is:", fibonacci(num))
12
```

OUTPUT:



Observation:

The recursive function breaks the problem into smaller parts and solves them step by step until it reaches a stopping point, after which the final result is obtained.

TASK 3

Transparency in Error Handling Prompt: Generate code with proper error handling and clear explanations for each exception.

Code:

task3.py > ...

```python
#task3
filename = input("Enter file name: ")
try:
    file = open(filename, "r")
    data = file.read()
    print("File Content:")
    print(data)
    file.close()
except FileNotFoundError:
    print("Error: The file does not exist.")
except PermissionError:
    print("Error: You do not have permission to read this file.")
except Exception as e:
    print("An unexpected error occurred:", e)

```

OUTPUT:

```
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.exe
NUSHA/OneDrive/Desktop/AI-CODING/task3.py
Enter file name: task3
Error: The file does not exist.
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

The program safely reads the file and clearly reports errors when the f ile is missing or inaccessible, instead of stopping unexpectedly.

TASK 4

Security in User Authentication

Prompt: Generate a simple Python-based login system.

Code:

task4.py > ...

```python

#task 4
#1.Insecure login code
username = input("Username: ")
password = input("Password: ")
if password == "admin123":
    print("Login successful")
else:
    print("Login failed")

#2.Secure login code with hashed passwords
import hashlib
def hash_password(password):
    return
    hashlib.sha256(password.encode()).hexdigest()
stored_hashed_password = hash_password("admin123")
username = input("Username: ")
password = input("Password: ")
if hash_password(password) == stored_hashed_password:
    print("Login successful")
else:
    print("Login failed")

```

OUTPUT:

```
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Python/Python314/python.
esktop/AI-CODING/task4.py
Username: anusha
Password: admin123
Login successful
Username: anusha
Password: admin123
Login successful
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

The first code is insecure because it uses plain-text passwords, while the second code improves security by using password hashing.

TASK 5

Privacy in Data Logging

Prompt: Generate a Python script that logs user activity including username, IP address, and timestamp.

Code:

task5.py > ...

```python

#task5
#Generate a Python script that logs user activity including username, IP address, and timestamp.
import logging
from datetime import datetime
# Configure logging
logging.basicConfig(filename='user_activity.log', level=logging.INFO, format='%(asctime)s - %(message)s')
def log_user_activity(username, ip_address):
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    logging.info(f'Username: {username}, IP Address: {ip_address}, Timestamp: {timestamp}')
# Example usage
username = input("Enter your username: ")
ip_address = input("Enter your IP address: ")
log_user_activity(username, ip_address)
print("User activity logged successfully.")
```

OUTPUT:

```
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING> & C:/Users/ANUSHA/AppData/Local/Programs/Py
esktop/AI-CODING/task5.py
Enter your username: task5
Enter your IP address: 45678
User activity logged successfully.
PS C:\Users\ANUSHA\OneDrive\Desktop\AI-CODING>
```

Observation:

The program logs complete user details such as username and IP address, which may expose sensitive information and create privacy risks if the log file is accessed without proper security.