Assignment-9.1

Name: K. Nithin Kumar

Hall Ticket: 2303A51630

Batch -23

Problem 1:

Consider the following Python function: def

ind_max(numbers): return max(numbers)

Task:

• Write documentation for the function in all three formats:

(a) **Docstring**

(b) **Inline comments**

(c) **Google-style documentation**

- **Critically compare the three approaches. Discuss the advantages, disadvantages, and suitable use cases of each style.**
- **Recommend which documentation style is most e ective for a mathematical utilities library and justify your answer.**

```python
#doc string
def find_maximum(a, b):
    """Returns the maximum of two numbers.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The maximum of a and b.
    """
    return max(a, b)
print(find_maximum.__doc__)
#inline comment
def find_maximum(a, b):
    # Use the built-in max function to find the maximum of a and b
    return max(a, b)
print(find_maximum.__doc__)
#google style documentation
def find_maximum(a, b):
    """Returns the maximum of two numbers.

    Args:
        a (int): The first number.
        b (int): The second number.

    Returns:
        int: The maximum of a and b.
    """
    return max(a, b)
print(find_maximum.__doc__)
```

```
a (int): The first number.
b (int): The second number.


Returns:
int: The maximum of a and b.

None
Returns the maximum of two numbers.
None
Returns the maximum of two numbers.
Returns the maximum of two numbers.

Args:
    a (int): The first number.
    b (int): The second number.

Returns:
    int: The maximum of a and b.
```

Problem 2: Consider the following Python function: def login(user, password, credentials): return credentials.get(user) == password Task:

1. **Write documentation in all three formats.**
2. **Critically compare the approaches.**
3. **Recommend which style would be most helpful for new developers onboarding a project, and justify your choice.**

```python
#doc string
def login(username,password,credentials):
    """Checks if the provided username and password match the credentials.

    Parameters:
    username (str): The username to check.
    password (str): The password to check.
    credentials (dict): A dictionary containing valid username-password pairs.

    Returns:
    bool: True if the credentials are valid, False otherwise.
    """
    return credentials.get(username) == password
print(login.__doc__)
#inline comment
def login(username,password,credentials):
    # Check if the provided username and password match the credentials
    return credentials.get(username) == password
print(login.__doc__)
#google style documentation
def login(username,password,credentials):
    """Checks if the provided username and password match the credentials.

    Args:
        username (str): The username to check.
        password (str): The password to check.
        credentials (dict): A dictionary containing valid username-password pairs.
    Returns:
        bool: True if the credentials are valid, False otherwise.
    """
    return credentials.get(username) == password
print(login.__doc__)
```

Output:

```
Checks if the provided username and password match the credentials.

Parameters:
username (str): The username to check.
password (str): The password to check.
credentials (dict): A dictionary containing valid username-password pairs.

Returns:
bool: True if the credentials are valid, False otherwise.

None
Checks if the provided username and password match the credentials.

Args:
    username (str): The username to check.
    password (str): The password to check.
    credentials (dict): A dictionary containing valid username-password pairs.
Returns:
    bool: True if the credentials are valid, False otherwise.
```

Problem 3: Calculator (Automatic Documentation Generation) Task: Design a Python module named calculator.py and demonstrate automatic documentation generation.

Instructions:

1. Create a Python module calculator.py that includes the following functions, each written with appropriate docstrings:

add(a, b) – returns the sum of two numbers o subtract(a, b) – returns the difference of two numbers o multiply(a, b) – returns the product of two numbers o divide(a, b) – returns the quotient of two numbers 2. Display the module documentation in the terminal using Python's documentation tools.

3. Generate and export the module documentation in HTML format using the pydoc utility, and open the generated HTML file in a web browser to verify the output.

ulations.py > subtract

```python
def add(a, b):

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The sum of a and b.
    """
    return a + b
print(add.__doc__)
def subtract(a, b):
    """Returns the difference of a and b.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The difference of a and b.
    """
    return a - b
print(subtract.__doc__)
def multiply(a, b):
    """Returns the product of a and b.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The product of a and b.
    """
    return a * b
print(multiply.__doc__)
def divide(a, b):
    """Returns the quotient of a and b.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The quotient of a and b.

    Raises:
    ValueError: If b is zero.
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b
print(divide.__doc__)
```

#doc string

Functions

add(a, b)
Returns the sum of a and b.

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The sum of a and b.

divide(a, b)
Returns the quotient of a and b.

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The quotient of a and b.

Raises:
ValueError: If b is zero.

multiply(a, b)
Returns the product of a and b.

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The product of a and b.

subtract(a, b)
Returns the difference of a and b.

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The difference of a and b.

4. Conversion Utilities Module Task:

1. Write a module named conversion.py with functions:

o **decimal_to_binary(n)**

o **binary_to_decimal(b) o decimal_to_hexadecimal(n)**

2. **Use Copilot for auto-generating docstrings.**

3. **Generate documentation in the terminal.**

4. **Export the documentation in HTML format and open it in a browser.**

```
Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The difference of a and b.

Returns the product of a and b.

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The product of a and b.

Returns the quotient of a and b.

Parameters:

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The product of a and b.

Returns the quotient of a and b.

Parameters:

Returns the quotient of a and b.

Parameters:

Parameters:
a (int): The first number.
b (int): The second number.

Returns:
int: The quotient of a and b.

Raises:
ValueError: If b is zero.

wrote calculations.html
```

```
def decimal_to_binary(n):
        n (int): The decimal number to convert.

    Returns:
        str: The binary representation of the decimal number.
    """
    if n == 0:
        return "0"

    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n //= 2

    return binary
print(decimal_to_binary.__doc__)
def binary_to_decimal(b):
    """Converts a binary number to decimal.

    Args:
        b (str): The binary number to convert.

    Returns:
        int: The decimal representation of the binary number.
    """
    decimal = 0
    for index, digit in enumerate(reversed(b)):
        decimal += int(digit) * (2 ** index)

    return decimal
print(binary_to_decimal.__doc__)
def decimal_to_hexadecimal(n):
    """Converts a decimal number to hexadecimal.

    Args:
        n (int): The decimal number to convert.

    Returns:
        str: The hexadecimal representation of the decimal number.
    """
    if n == 0:
        return "0"

    hexadecimal = ""
    hex_digits = "0123456789ABCDEF"

    while n > 0:
        hexadecimal = hex_digits[n % 16] + hexadecimal
        n //= 16

    return hexadecimal
print(decimal_to_hexadecimal.__doc__)
```

```
Server ready at http://localhost:8080/
Server commands: [b]rowser, [q]uit
server> b
server>
```

conversion

#doc_string

Functions

binary_to_decimal(b)

```
Converts a binary number to decimal.

Args:
    b (str): The binary number to convert.

Returns:
    int: The decimal representation of the binary number.
```

decimal_to_binary(n)

```
Converts a decimal number to binary.

Args:
    n (int): The decimal number to convert.

Returns:
    str: The binary representation of the decimal number.
```

decimal_to_hexadecimal(n)

```
Converts a decimal number to hexadecimal.

Args:
    n (int): The decimal number to convert.

Returns:
    str: The hexadecimal representation of the decimal number.
```

Problem 5 – Course Management Module Task:

1. Create a module course.py with functions:

o add_course(course_id, name, credits) o

remove_course(course_id) o get_course(course_id)

2. **Add docstrings with Copilot.**

3. **Generate documentation in the terminal.**

4. **Export the documentation in HTML format and open it in a Browser**

```python
#doc string
def add_course(course_id,name,credits):
    """Adds a course to the system.
    Parameters:
    course_id (str): The unique identifier for the course.
    name (str): The name of the course.
    credits (int): The number of credits for the course.
    Returns:
    str: A message indicating the success or failure of the operation.
    """
    # Code to add course to database
    return f"Course {name} with ID {course_id} and {credits} credits added successfully."
print(add_course.__doc__)
def remove_course(course_id):
    """Removes a course from the system.
    Parameters:
    course_id (str): The unique identifier for the course to remove.
    Returns:
    str: A message indicating the success or failure of the operation.
    """
    # Code to remove course from database
    return f"Course with ID {course_id} removed successfully."
print(remove_course.__doc__)
def get_course_info(course_id):
    """Retrieves information about a course.
    Parameters:
    course_id (str): The unique identifier for the course.
    Returns:
    str: A message indicating the success or failure of the operation.
    """
    # Code to get course information from database
    return f"Information for course with ID {course_id} retrieved successfully."
```

```
Adds a course to the system.
Parameters:
course_id (str): The unique identifier for the course.
name (str): The name of the course.
credits (int): The number of credits for the course.
Returns:
str: A message indicating the success or failure of the operation.

Removes a course from the system.
Parameters:
course_id (str): The unique identifier for the course to remove.

Removes a course from the system.
Parameters:
course_id (str): The unique identifier for the course to remove.
course_id (str): The unique identifier for the course to remove.
Returns:
```

```
Server ready at http://localhost:8080/
Server commands: [b]rowser, [q]uit
server> b
server>
```

#doc string

Functions

add_course(course_id, name, credits)
Adds a course to the system.
Parameters:
course_id (str): The unique identifier for the course.
name (str): The name of the course.
credits (int): The number of credits for the course.
Returns:
str: A message indicating the success or failure of the operation.

get_course_info(course_id)
Retrieves information about a course.
Parameters:
course_id (str): The unique identifier for the course.
Returns:
str: A message indicating the success or failure of the operation.

remove_course(course_id)
Removes a course from the system.
Parameters:
course_id (str): The unique identifier for the course to remove.
Returns:
str: A message indicating the success or failure of the operation.

Python 3.13.5 [tags/v3.13.5:6cb20a2, MSC v.1943 64 bit (AMD64)]
Windows-11

course

#doc string

Functions

add_course(course_id, name, credits)

```
Adds a course to the system.
Parameters:
course_id (str): The unique identifier for the course.
name (str): The name of the course.
credits (int): The number of credits for the course.
Returns:
str: A message indicating the success or failure of the operation.
```

get_course_info(course_id)

```
Retrieves information about a course.
Parameters:
course_id (str): The unique identifier for the course.
Returns:
str: A message indicating the success or failure of the operation.
```

remove_course(course_id)

```
Removes a course from the system.
Parameters:
course_id (str): The unique identifier for the course to remove.
Returns:
str: A message indicating the success or failure of the operation.
```