AI ASSISTED CODING

ASSIGNMENT -9.1

Name: Harshitha Guda

H.T.No: 2303A51102

Problem 1:

Consider the following Python function:

def find_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 effective for a mathematical utilities library and justify your answer.

```python
def find_max(numbers):
    """
    This function takes a list of numbers as input and returns the maximum number in the list.
    Parameters:
    numbers (list): A list of numerical values.
    Returns:
    The maximum number in the list. If the list is empty, it returns None.
    """
    if not numbers:
        return None  # Return None for an empty list

    max_number = numbers[0]  # Initialize max_number to the first element

    for num in numbers:
        if num > max_number:
            max_number = num  # Update max_number if a larger number is found

    return max_number
# Example usage
numbers = [3, 5, 7, 2, 8]
max_value = find_max(numbers)  # Storing the maximum value in a variable
print(f"The maximum number in the list is: {max_value}")  # Printing the maximum value found in the list
print(find_max.__doc__)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Returns:
The maximum number in the list. If the list is empty, it returns None.

The maximum number in the list is: 8
The maximum number in the list is: 8
PS C:\Users\gudah\OneDrive\Documents\AIAC>
```

```python
def find_max(numbers):
    if not numbers:
        return None  # Return None for an empty list

    max_number = numbers[0]  # Initialize max_number to the first element

    for num in numbers:
        if num > max_number:
            max_number = num  # Update max_number if a larger number is found

    return max_number
# Example usage
numbers = [3, 5, 7, 2, 8]
max_value = find_max(numbers)#storing the maximum value in a variable
print(f"The maximum number in the list is: {max_value}")#printing the maximum value found in the list

```

```python
#Generate python code for returning the maximum value from a list of numbers and print the result and give google style docstrings for the function
def find_max(numbers:list)->int:
    """
    Returns the maximum value from a list of numbers.

    Args:
        numbers (list): A list of numerical values.
    Returns:
        int: The maximum number in the list. If the list is empty, it returns None.
    """
    if not numbers:
        return None  # Return None for an empty list

    max_number = numbers[0]  # Initialize max_number to the first element

    for num in numbers:
        if num > max_number:
            max_number = num  # Update max_number if a larger number is found

    return max_number
# Example usage
numbers = [3, 5, 7, 2, 8]
max_value = find_max(numbers)  # Storing the maximum value in a variable
print(f"The maximum number in the list is: {max_value}")  # Printing the maximum value found in the list
```

Docstring: Written inside triple quotes below the function, it explains what the function does, its inputs and outputs, and can be viewed using help(), making it simple and easy to write.

Inline Comments: Written using #, they explain the logic step by step inside the code and help developers understand how each part works clearly.

Google-Style Documentation: A structured form of docstring that uses sections like Args and Returns, looks professional, and is suitable for large projects.

Best Style: Google-style documentation is best for a mathematical utilities library because it clearly explains inputs, outputs, and formulas. It keeps documentation organized and easy to understand for other developers.

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
def login(user,password,credentials):
    """
    This function checks if the provided username and password match any entry in the credentials dictionary.

    Args:
        user (str): The username to be checked.
        password (str): The password to be checked.
        credentials (dict): A dictionary where keys are usernames and values are passwords.

    Returns:
        bool: True if the username and password match an entry in the credentials, False otherwise.
        """
    if user in credentials and credentials[user] == password:
        return True
    else:
        return False
# Example usage
credentials = {"admin": "admin123", "user1": "password1", "user2": "password2"}
print(login("admin", "admin123", credentials))  # Output: True
print(login("user1", "wrongpassword", credentials))  # Output: False
print(login.__doc__)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\gudah\OneDrive\Documents\AIAC> & C:\Python314\python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_9.1.py
True
False

This function checks if the provided username and password match any entry in the credentials dictionary.

Args:
    user (str): The username to be checked.
    password (str): The password to be checked.
    credentials (dict): A dictionary where keys are usernames and values are passwords.

Returns:
    bool: True if the username and password match an entry in the credentials, False otherwise.
```

```python
def login(user,password,credentials):
    if user in credentials and credentials[user] == password:#checking if the user exists in the credentials and if the password matches
        return True#if the user exists and the password matches, return True
    else:#if the user does not exist or the password does not match, return False
        return False
# Example usage
credentials = {"admin": "admin123", "user1": "password1", "user2": "password2"}#a dictionary containing usernames and their corresponding passwords
print(login("admin", "admin123", credentials))  # Output: True
print(login("user1", "wrongpassword", credentials))  # Output: False


#Generate python code for a login function that checks if the provided username and password match any entry in a credentials dictionary and print the result and give goo
def login(user:str,password:str,credentials:dict)->bool:
    """
    This function checks if the provided username and password match any entry in the credentials dictionary.

    Args:
        user (str): The username to be checked.
        password (str): The password to be checked.
        credentials (dict): A dictionary where keys are usernames and values are passwords.

    Returns:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS C:\Users\gudah\OneDrive\Documents\AIAC> & C:\Python314\python.exe c:/Users/gudah/OneDrive/Documents/AIAC/Lab_assignment_9.1.py
Returns:
    bool: True if the username and password match an entry in the credentials, False otherwise.

True
False
True
False
PS C:\Users\gudah\OneDrive\Documents\AIAC>
```

```python
#Generate python code for a login function that checks if the provided username and password match any entry in a credentials dictionary and print the result and give
def login(user:str,password:str,credentials:dict)->bool:
    """
    This function checks if the provided username and password match any entry in the credentials dictionary.

    Args:
        user (str): The username to be checked.
        password (str): The password to be checked.
        credentials (dict): A dictionary where keys are usernames and values are passwords.

    Returns:
        bool: True if the username and password match an entry in the credentials, False otherwise.
    """
    if user in credentials and credentials[user] == password:
        return True
    else:
        return False
# Example usage
credentials = {"admin": "admin123", "user1": "password1", "user2": "password2"}#a dictionary containing usernames and their corresponding passwords
print(login("admin", "admin123", credentials))  # Output: True
print(login("user1", "wrongpassword", credentials))  # Output: False
```

Google-style documentation is most helpful for new developers. It clearly explains inputs, outputs, and purpose in a structured way. New developers can quickly understand how to use the function. It keeps documentation neat and professional. It is best for onboarding in real projects.

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:

o 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.

calculator.py > ...

```python

This module provides basic arithmetic operations: addition, subtraction, multiplication, and division.
Each function takes two numbers as input and returns the result of the respective operation.
The functions also include Google-style docstrings that describe their purpose, arguments, and return values.
The module can be used as a simple calculator for performing basic mathematical operations.
Example usage of each function is provided,
along with the ability to print the documentation for each function using the __doc__ attribute.

"""

def add(a,b):
    """
    This function takes two numbers as input and returns their sum.

    Args:
        a (int): The first number.
        b (int): The second number.
    Returns:
        int: The sum of the two numbers.
    """
    return a + b
print("The sum of 5 and 3 is:", add(5, 3))  # Example usage of the add function and printing the result
print(add.__doc__)

def subtract(a,b):
    """
    This function takes two numbers as input and returns their difference.

    Args:
        a (int): The first number.
        b (int): The second number.
    Returns:
        int: The difference of the two numbers.
    """
    return a - b
print("The difference between 5 and 3 is:", subtract(5, 3))  # Example usage of the subtract function and printing the result
print(subtract.__doc__)
```

```python
def multiply(a,b):
    """
    This function takes two numbers as input and returns their product.

    Args:
        a (int): The first number.
        b (int): The second number.
    Returns:
        int: The product of the two numbers.
        """
    return a * b
print("The product of 5 and 3 is:", multiply(5, 3))  # Example usage of the multiply function and printing the result
print(multiply.__doc__)

def divide(a,b):
    """
    This function takes two numbers as input and returns their quotient.

    Args:
        a (int): The first number (dividend).
        b (int): The second number (divisor).
    Returns:
        float: The quotient of the two numbers. If the divisor is zero, it returns None.
     """
    if b == 0:
        return None  # Return None for division by zero
    return a / b
print("The quotient of 5 and 3 is:", divide(5, 3))  # Example usage of the divide function and printing the result
print("The quotient of 5 and 0 is:", divide(5, 0))

import calculator
print(calculator.add.__doc__)  # Printing the documentation for the add function

```

```
PS C:\Users\gudah\OneDrive\Documents\AIAC> python -m pydoc calculator

This function takes two numbers as input and returns their sum.

Args:
    a (int): The first number.
    b (int): The second number.
Returns:
    int: The sum of the two numbers.

Help on module calculator:

NAME
    calculator

DESCRIPTION
    This module provides basic arithmetic operations: addition, subtraction, multiplication, and division.
    Each function takes two numbers as input and returns the result of the respective operation.
    The functions also include Google-style docstrings that describe their purpose, arguments, and return values.
    The module can be used as a simple calculator for performing basic mathematical operations.
    Example usage of each function is provided,
    along with the ability to print the documentation for each function using the __doc__ attribute.

FUNCTIONS
    add(a, b)
        This function takes two numbers as input and returns their sum.

        Args:
            a (int): The first number.
            b (int): The second number.
        Returns:
            int: The sum of the two numbers.

    divide(a, b)
        This function takes two numbers as input and returns their quotient.

        Args:
            a (int): The first number (dividend).
            b (int): The second number (divisor).
        Returns:
```

calculator c:\users\gudah\onedrive\documents\aiac\calculator.py

This module provides basic arithmetic operations: addition, subtraction, multiplication, and division.
Each function takes two numbers as input and returns the result of the respective operation.
The functions also include Google-style docstrings that describe their purpose, arguments, and return values.
The module can be used as a simple calculator for performing basic mathematical operations.
Example usage of each function is provided,
along with the ability to print the documentation for each function using the __doc__ attribute.

Modules

calculator

Functions

add(a, b)
This function takes two numbers as input and returns their sum.

Args:
a (int): The first number.
b (int): The second number.
Returns:
int: The sum of the two numbers.

divide(a, b)
This function takes two numbers as input and returns their quotient.

Args:
a (int): The first number (dividend).
b (int): The second number (divisor).
Returns:
float: The quotient of the two numbers. If the divisor is zero, it returns None.

multiply(a, b)
This function takes two numbers as input and returns their product.

Problem 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.

conversions.py > binary_to_decimal

```python
"""
This module provides functions for converting between different number systems: decimal to binary, binary to decimal, and decimal to hexadecimal.
Each function includes a Google-style docstring that describes its purpose, arguments, and return value.
The module can be used to perform various conversions between these number systems,
and example usage of each function is provided, along with the ability to print the
documentation for each function using the __doc__ attribute.

"""

def decimal_to_binaary(n):
    """
    This function converts a decimal number to its binary representation.

    Args:
        n (int): The decimal number to be converted.

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

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

    return binary
print("The binary representation of 10 is:", decimal_to_binaary(10))  # Example usage of the decimal_to_binary function and printing the result
print(decimal_to_binaary.__doc__)

def binary_to_decimal(b):
    """
    This function converts a binary number (given as a string) to its decimal representation.

    Args:
        b (str): The binary number to be converted, represented as a string of '0's and '1's.

    Returns:
        int: The decimal representation of the input binary number.
    """
    decimal = 0
    power = 0

    for digit in reversed(b):
        if digit == '1':
            decimal += 2 ** power
        power += 1

    return decimal
print("The decimal representation of '1010' is:", binary_to_decimal('1010'))  # Example usage of the binary_to_decimal function and printing the result
print(binary_to_decimal.__doc__)

def decimal_to_hexadecimal(n):
    """
    This function converts a decimal number to its hexadecimal representation.

    Args:
        n (int): The decimal number to be converted.

    Returns:
        str: The hexadecimal representation of the input 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("The hexadecimal representation of 255 is:", decimal_to_hexadecimal(255))  # Example usage of the decimal_to_hexadecimal function and printing the result
print(decimal_to_hexadecimal.__doc__)

import conversions
print(conversions.__doc__)
```

```
PS C:\Users\gudah\OneDrive\Documents\AIAC> & C:\Python314\python.exe c:/Users/gudah/OneDrive/Documents/AIAC/conversions.py
The binary representation of 10 is: 1010

This function converts a decimal number to its binary representation.

Args:
    n (int): The decimal number to be converted.

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

The decimal representation of '1010' is: 10

This function converts a binary number (given as a string) to its decimal representation.

Args:
    b (str): The binary number to be converted, represented as a string of '0's and '1's.

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

The hexadecimal representation of 255 is: FF

This function converts a decimal number to its hexadecimal representation.

Args:
    n (int): The decimal number to be converted.

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

The binary representation of 10 is: 1010

This function converts a decimal number to its binary representation.

Args:
    n (int): The decimal number to be converted.

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

```
The decimal representation of '1010' is: 10

This function converts a binary number (given as a string) to its decimal representation.

Args:
    b (str): The binary number to be converted, represented as a string of '0's and '1's.

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

The hexadecimal representation of 255 is: FF

This function converts a decimal number to its hexadecimal representation.

Args:
    n (int): The decimal number to be converted.

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


This module provides functions for converting between different number systems: decimal to binary, binary to decimal, and decimal to hexadecimal.
Each function includes a Google-style docstring that describes its purpose, arguments, and return value.
The module can be used to perform various conversions between these number systems,
and example usage of each function is provided, along with the ability to print the
documentation for each function using the __doc__ attribute.



This module provides functions for converting between different number systems: decimal to binary, binary to decimal, and decimal to hexadecimal.
Each function includes a Google-style docstring that describes its purpose, arguments, and return value.
The module can be used to perform various conversions between these number systems,
and example usage of each function is provided, along with the ability to print the
documentation for each function using the __doc__ attribute.
The module can be used to perform various conversions between these number systems,
and example usage of each function is provided, along with the ability to print the
documentation for each function using the __doc__ attribute.
```

index

conversions c:\users\gudah\onedrive\documents\aiac\conversions.py

This module provides functions for converting between different number systems: decimal to binary, binary to decimal, and decimal to hexadecimal.
Each function includes a Google-style docstring that describes its purpose, arguments, and return value.
The module can be used to perform various conversions between these number systems,
and example usage of each function is provided, along with the ability to print the
documentation for each function using the __doc__ attribute.

Modules

conversions

Functions

binary_to_decimal(b)
This function converts a binary number (given as a string) to its decimal representation.

Args:
b (str): The binary number to be converted, represented as a string of '0's and '1's.

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

decimal_to_binaary(n)
This function converts a decimal number to its binary representation.

Args:
n (int): The decimal number to be converted.

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

decimal_to_hexadecimal(n)
This function converts a decimal number to its hexadecimal representation.

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
"""
This module defines functions for managing courses in a course catalog.
It includes functions to add a course, remove a course, and retrieve course information.
Each function is documented with Google style docstrings to explain its purpose, parameters, and return values.
The module can be used to simulate a simple course management system, allowing users to add, remove, and retrieve course information.
Example usage of each function is provided, along with the ability to print the documentation for each function using the __doc__ attribute.
"""
def add_course(course_id,name,credits):
    """
    This function takes a course ID, name, and credits as input and returns a dictionary representing the course.
    Args:
        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:
    dict: A dictionary containing the course information with keys 'course_id', 'name', and 'credits'.
    """
    return {
        "course_id": course_id,
        "name": name,
        "credits": credits
    }
# Example usage of the add_course function and printing the result
course_info = add_course("CS101", "Introduction to Computer Science", 4)
print(course_info)
print(add_course.__doc__)

```

```python
def remove_course(course_id):
    """
    This function takes a course ID as input and simulates the removal of a course from a course catalog.
    Args:course_id (str): The unique identifier for the course to be removed.
    Returns:
    str: A message indicating that the course has been removed.
    """
    return f"Course with ID {course_id} has been removed from the catalog."
# Example usage of the remove_course function and printing the result
print(remove_course("CS101"))
print(remove_course.__doc__)

def get_course(course_id):
    """
    This function takes a course ID as input and simulates retrieving course information from a course catalog.
    Args:course_id (str): The unique identifier for the course to be retrieved.
    Returns:
    dict: A dictionary containing the course information if found, or a message indicating that the course was not found.
    """
    # Simulating a course catalog with a dictionary
    course_catalog = {
        "CS101": {"name": "Introduction to Computer Science", "credits": 4},
        "MATH201": {"name": "Calculus I", "credits": 3},
        "ENG301": {"name": "English Literature", "credits": 2}
    }
    if course_id in course_catalog:
        return course_catalog[course_id]
    else:
        return f"Course with ID {course_id} not found in the catalog."
# Example usage of the get_course function and printing the result
print(get_course("CS101"))
print(get_course("HIST101"))
print(get_course.__doc__)

import course
print(course.__doc__)
```

```
PS C:\Users\gudah\OneDrive\Documents\AIAC> & C:\Python314\python.exe c:/Users/gudah/OneDrive/Documents/AIAC/course.py
{'course_id': 'CS101', 'name': 'Introduction to Computer Science', 'credits': 4}

This function takes a course ID, name, and credits as input and returns a dictionary representing the course.
Args:
    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:
dict: A dictionary containing the course information with keys 'course_id', 'name', and 'credits'.

Course with ID CS101 has been removed from the catalog.

This function takes a course ID as input and simulates the removal of a course from a course catalog.
Args:course_id (str): The unique identifier for the course to be removed.
Returns:
str: A message indicating that the course has been removed.

{'name': 'Introduction to Computer Science', 'credits': 4}
Course with ID HIST101 not found in the catalog.

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

{'course_id': 'CS101', 'name': 'Introduction to Computer Science', 'credits': 4}

This function takes a course ID, name, and credits as input and returns a dictionary representing the course.
Args:
    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:
dict: A dictionary containing the course information with keys 'course_id', 'name', and 'credits'.

Course with ID CS101 has been removed from the catalog.
```

```
This function takes a course ID as input and simulates the removal of a course from a course catalog.
Args:course_id (str): The unique identifier for the course to be removed.
Returns:
str: A message indicating that the course has been removed.

{'name': 'Introduction to Computer Science', 'credits': 4}
Course with ID HIST101 not found in the catalog.

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.


This module defines functions for managing courses in a course catalog.
It includes functions to add a course, remove a course, and retrieve course information.
 Each function is documented with Google style docstrings to explain its purpose, parameters, and return values.
The module can be used to simulate a simple course management system, allowing users to add, remove, and retrieve course information.
Example usage of each function is provided, along with the ability to print the documentation for each function using the __doc__ attribute.
```

index

course c:\users\gudah\onedrive\documents\aiac\course.py

This module defines functions for managing courses in a course catalog.
It includes functions to add a course, remove a course, and retrieve course information.
Each function is documented with Google style docstrings to explain its purpose, parameters, and return values.
The module can be used to simulate a simple course management system, allowing users to add, remove, and retrieve course information.
Example usage of each function is provided, along with the ability to print the documentation for each function using the __doc__ attribute.

Modules

course

Functions

add_course(course_id, name, credits)
This function takes a course ID, name, and credits as input and returns a dictionary representing the course.
Args:
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:
dict: A dictionary containing the course information with keys 'course_id', 'name', and 'credits'.

get_course(course_id)
This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

remove_course(course_id)
This function takes a course ID as input and simulates the removal of a course from a course catalog.
Args:course_id (str): The unique identifier for the course to be removed.
Returns:
str: A message indicating that the course has been removed.