AI ASSISTED CODING

LAB-9.1

Jadala Varshini

2303A51758

Batch-11

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

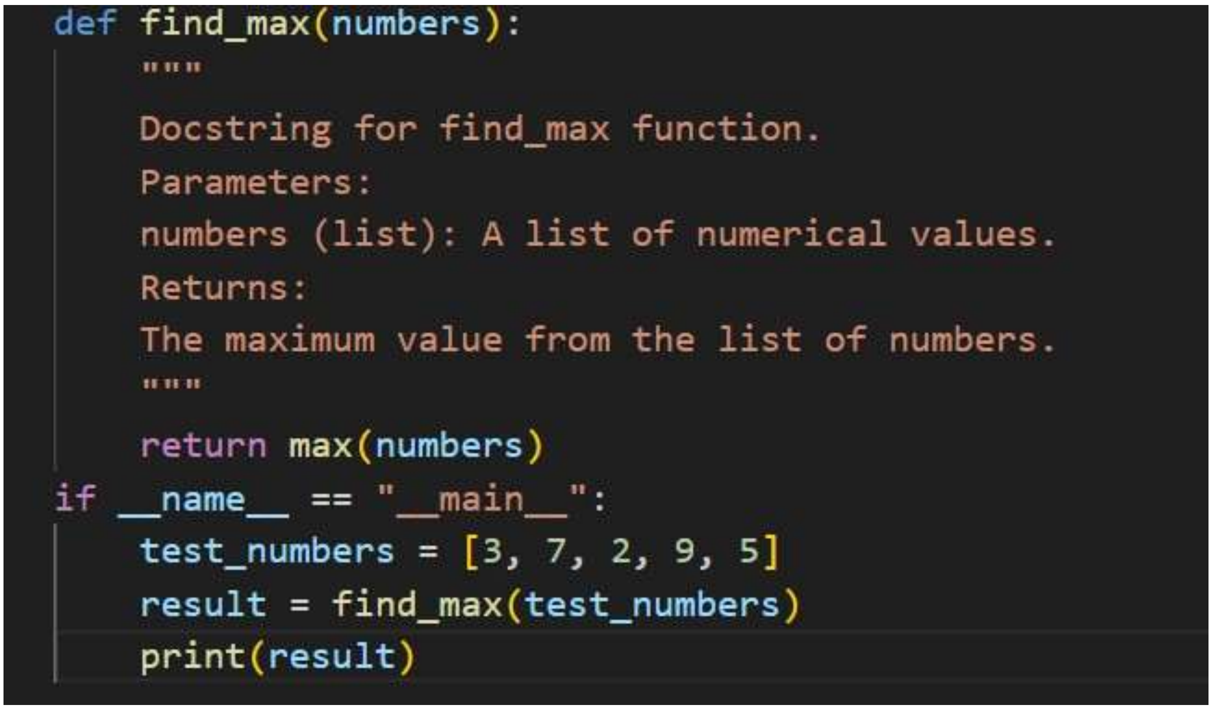
(b) Inline comments

(c) Google-style documentation

• Critically compare the three approaches. Discuss the advantages, disadvantages, and suitable use cases of each style.

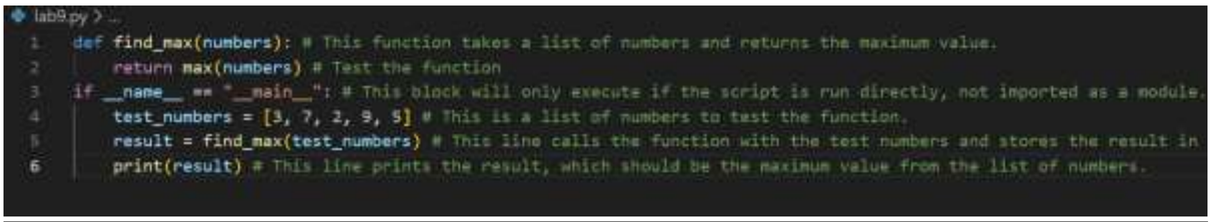
• Recommend which documentation style is most effective for a mathematical utilities library and justify your answer.

(a) Docstring



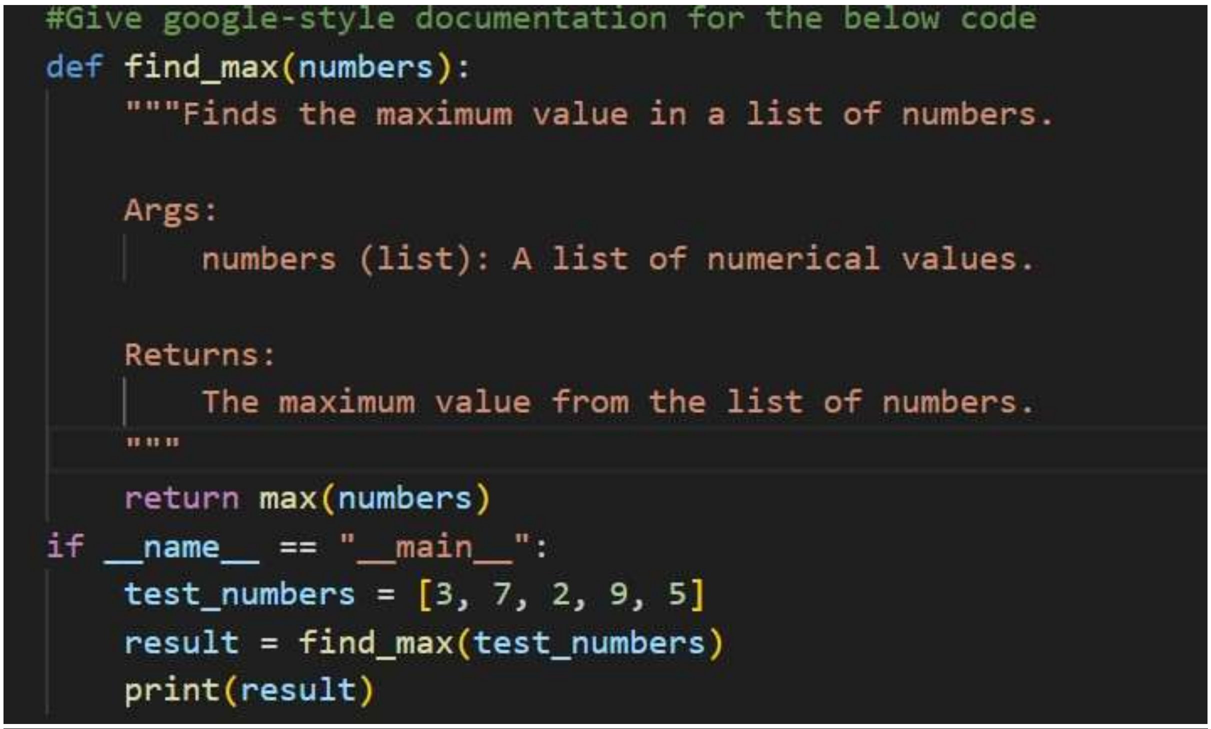
```python
def find_max(numbers):
    """
    Docstring for find_max function.
    Parameters:
    numbers (list): A list of numerical values.
    Returns:
    The maximum value from the list of numbers.
    """
    return max(numbers)
if __name__ == "__main__":
    test_numbers = [3, 7, 2, 9, 5]
    result = find_max(test_numbers)
    print(result)
```

(b) Inline comments



```python
def find_max(numbers): # This function takes a list of numbers and returns the maximum value.
    return max(numbers) # Test the function
if __name__ == "__main__": # This block will only execute if the script is run directly, not imported as a module.
    test_numbers = [3, 7, 2, 9, 5] # This is a list of numbers to test the function.
    result = find_max(test_numbers) # This line calls the function with the test numbers and stores the result in
    print(result) # This line prints the result, which should be the maximum value from the list of numbers.
```

(c) Google-style documentation



```python
#Give google-style documentation for the below code
def find_max(numbers):
    """Finds the maximum value in a list of numbers.

    Args:
        numbers (list): A list of numerical values.

    Returns:
        The maximum value from the list of numbers.
    """
    return max(numbers)
if __name__ == "__main__":
    test_numbers = [3, 7, 2, 9, 5]
    result = find_max(test_numbers)
    print(result)
```

Problem 2:

Consider the following Python function: def

login(user, password, credentials): return

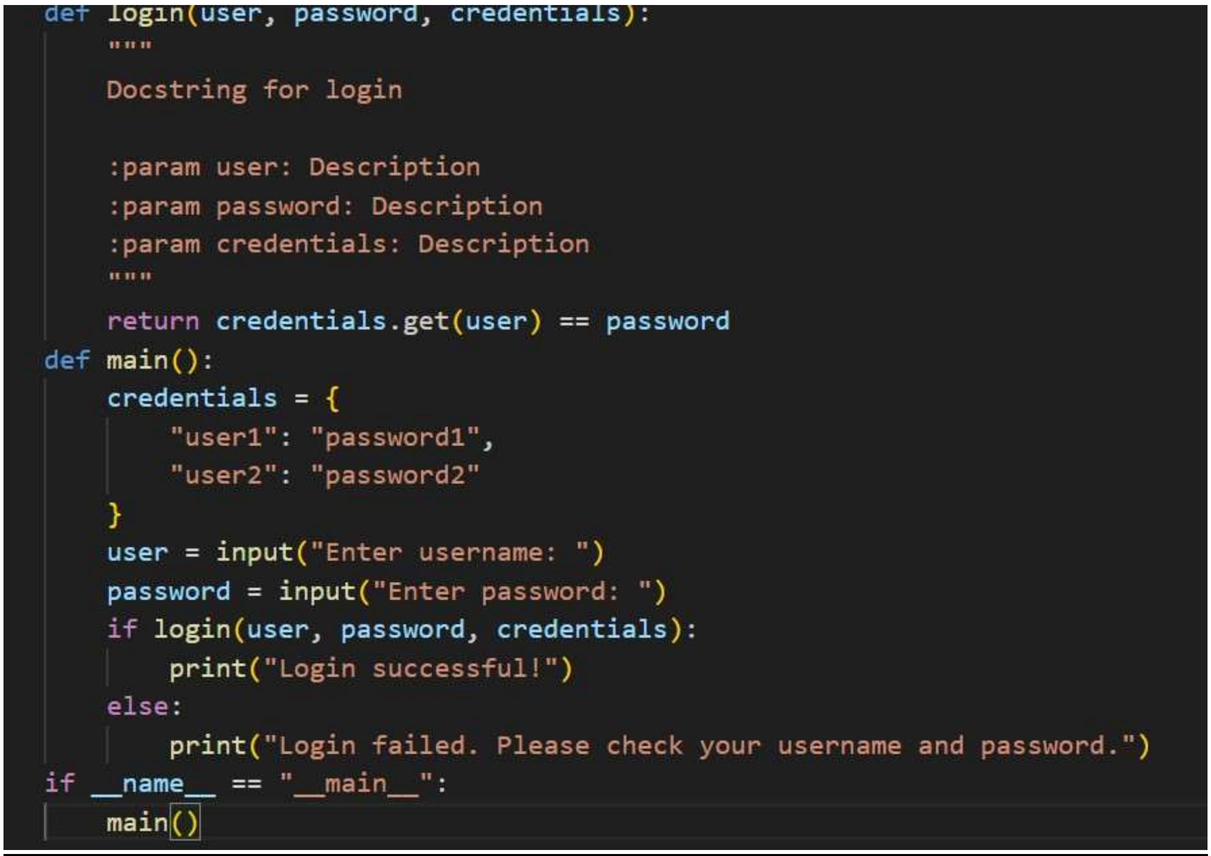
credentials.get(user) == password Task:

1. Write documentation in all three formats.

2. Critically compare the approaches.

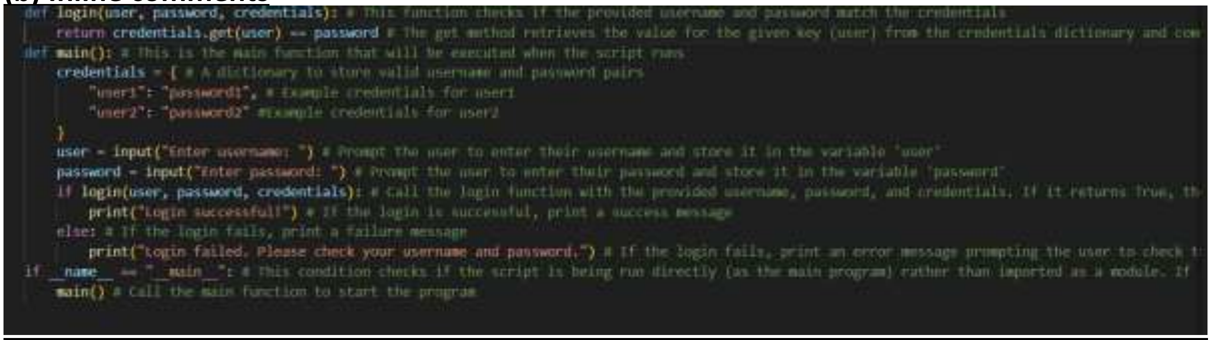
3. Recommend which style would be most helpful for new developers onboarding a project, and justify your choice.

(a) Docstring



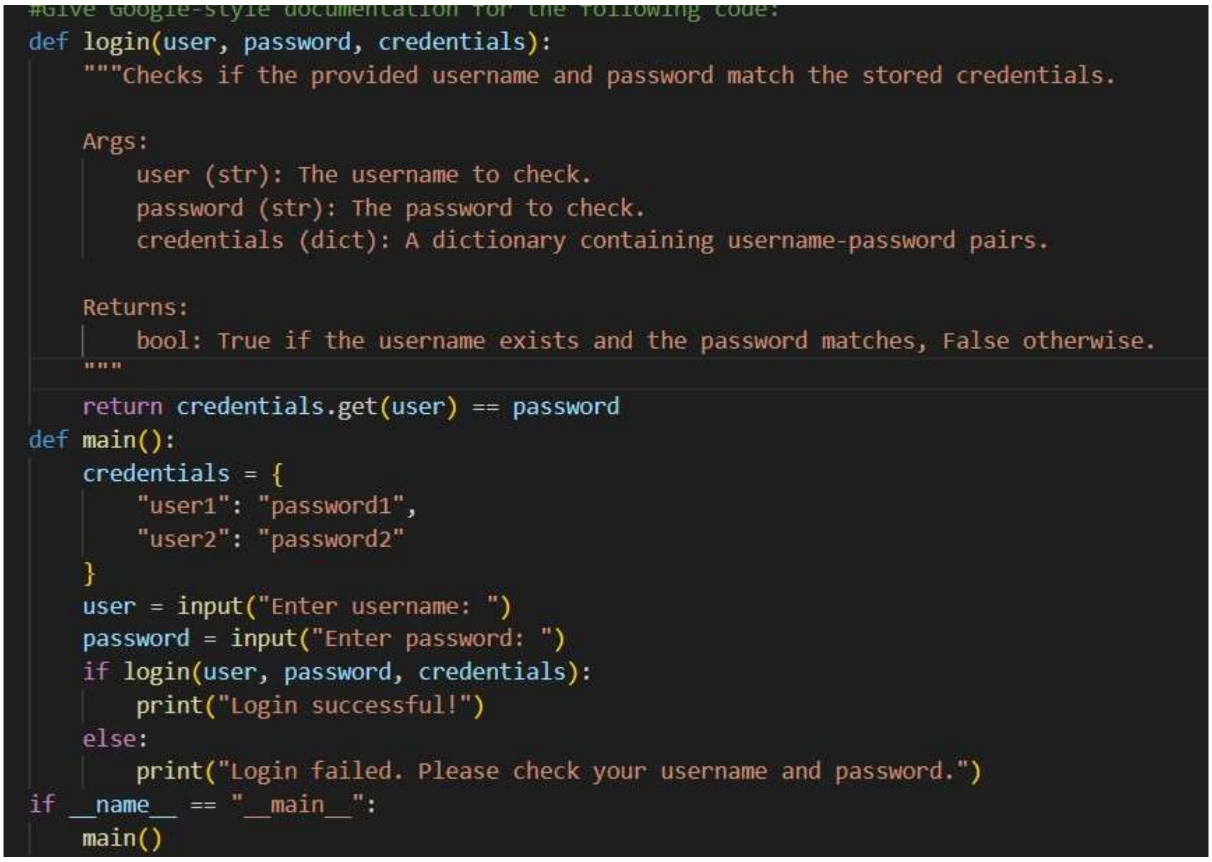
```
def login(user, password, credentials):
    """
    Docstring for login

    :param user: Description
    :param password: Description
    :param credentials: Description
    """
    return credentials.get(user) == password
def main():
    credentials = {
        "user1": "password1",
        "user2": "password2"
    }
    user = input("Enter username: ")
    password = input("Enter password: ")
    if login(user, password, credentials):
        print("Login successful!")
    else:
        print("Login failed. Please check your username and password.")
if __name__ == "__main__":
    main()
```

(b) Inline comments



```
def login(user, password, credentials): # This function checks if the provided username and password match the credentials
    return credentials.get(user) == password # The get method retrieves the value for the given key (user) from the credentials dictionary and com
def main(): # This is the main function that will be executed when the script runs
    credentials = { # A dictionary to store valid username and password pairs
        "user1": "password1", # Example credentials for user1
        "user2": "password2" #Example credentials for user2
    }
    user = input("Enter username: ") # Prompt the user to enter their username and store it in the variable 'user'
    password = input("Enter password: ") # Prompt the user to enter their password and store it in the variable 'password'
    if login(user, password, credentials): # Call the login function with the provided username, password, and credentials. If it returns True, th
        print("Login successful!") # If the login is successful, print a success message
    else: # If the login fails, print a failure message
        print("Login failed. Please check your username and password.") # If the login fails, print an error message prompting the user to check t
if __name__ == "__main__": # This condition checks if the script is being run directly (as the main program) rather than imported as a module. If
    main() # Call the main function to start the program
```

(c) Google-style documentation



```
#Give Google-style documentation for the following code:
def login(user, password, credentials):
    """Checks if the provided username and password match the stored credentials.

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

    Returns:
        bool: True if the username exists and the password matches, False otherwise.
    """
    return credentials.get(user) == password
def main():
    credentials = {
        "user1": "password1",
        "user2": "password2"
    }
    user = input("Enter username: ")
    password = input("Enter password: ")
    if login(user, password, credentials):
        print("Login successful!")
    else:
        print("Login failed. Please check your username and password.")
if __name__ == "__main__":
    main()
```

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

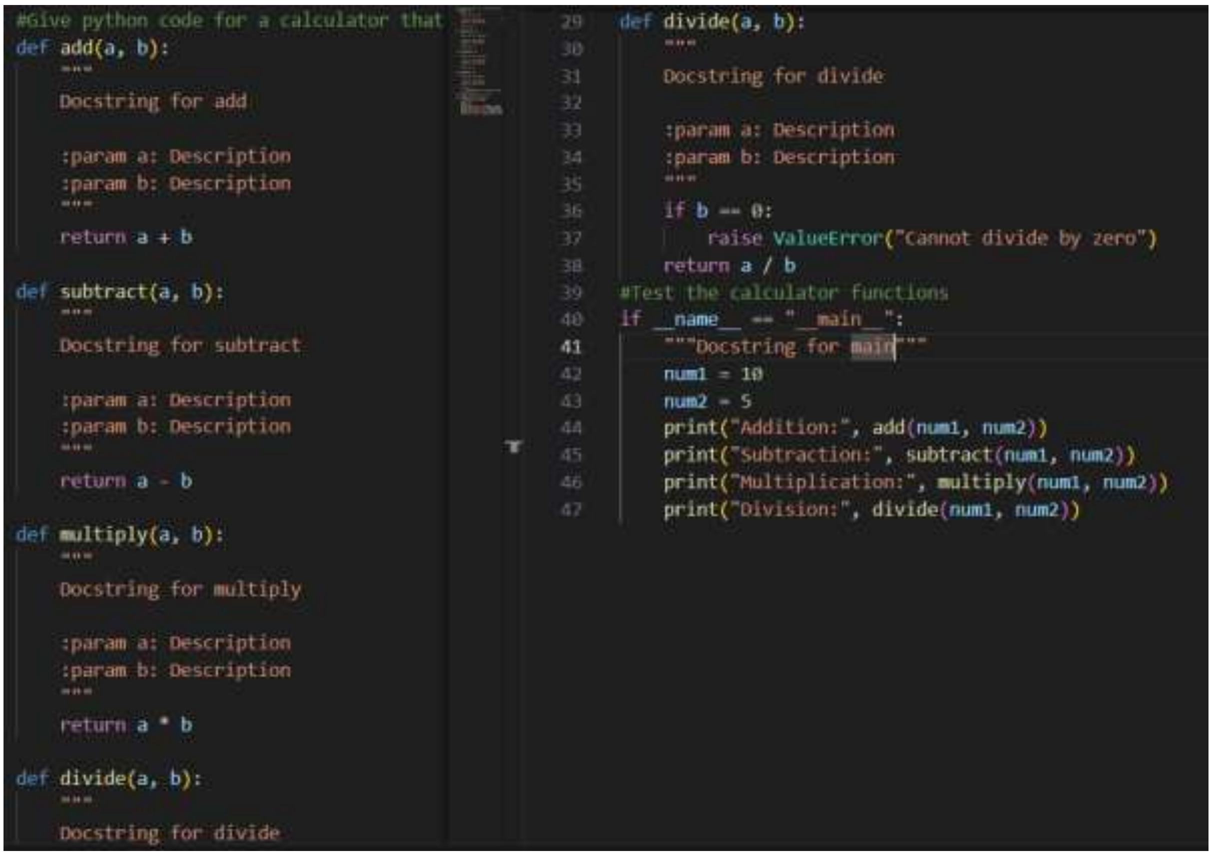
2. Display the module documentation in the terminal using Python's documentation tools.

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.

Prompt:

#Give python code for a calculator that has functions of addition, subtraction, multiplication and division

Docstring

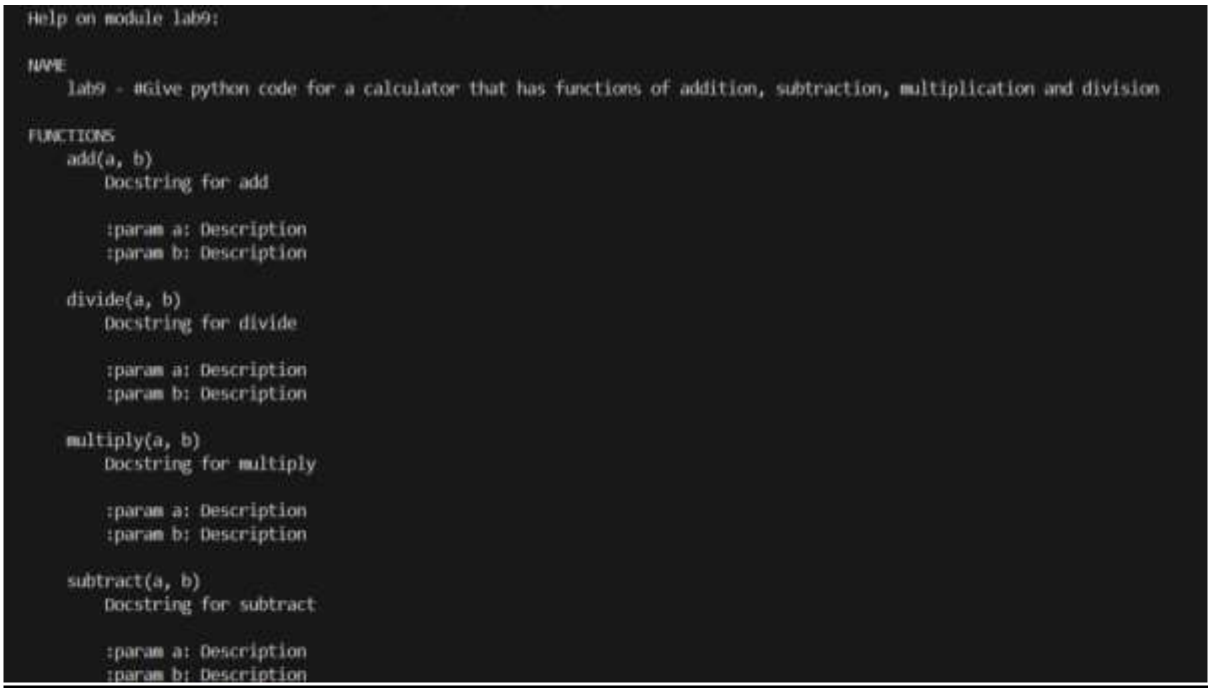


```python
#Give python code for a calculator that
def add(a, b):
    """
    Docstring for add

    :param a: Description
    :param b: Description
    """
    return a + b

def subtract(a, b):
    """
    Docstring for subtract

    :param a: Description
    :param b: Description
    """
    return a - b

def multiply(a, b):
    """
    Docstring for multiply

    :param a: Description
    :param b: Description
    """
    return a * b

def divide(a, b):
    """
    Docstring for divide
```

```python
def divide(a, b):
    """
    Docstring for divide

    :param a: Description
    :param b: Description
    """
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
#Test the calculator functions
if __name__ == "__main__":
    """Docstring for main"""
    num1 = 10
    num2 = 5
    print("Addition:", add(num1, num2))
    print("Subtraction:", subtract(num1, num2))
    print("Multiplication:", multiply(num1, num2))
    print("Division:", divide(num1, num2))
```

Using pydoc

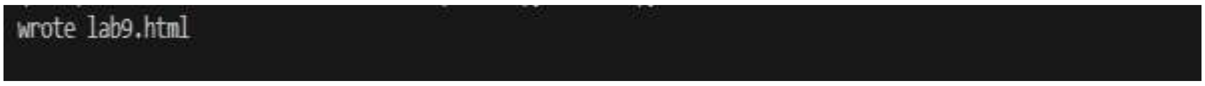
1) python -m pydoc filename



```
Help on module lab9:

NAME
    lab9 - #Give python code for a calculator that has functions of addition, subtraction, multiplication and division

FUNCTIONS
    add(a, b)
        Docstring for add

        :param a: Description
        :param b: Description

    divide(a, b)
        Docstring for divide

        :param a: Description
        :param b: Description

    multiply(a, b)
        Docstring for multiply

        :param a: Description
        :param b: Description

    subtract(a, b)
        Docstring for subtract

        :param a: Description
        :param b: Description
```

<u>2) python -m pydoc -w filename</u>

<u>3) python -m pydoc -p 8080</u>



```
wrote lab9.html
```

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

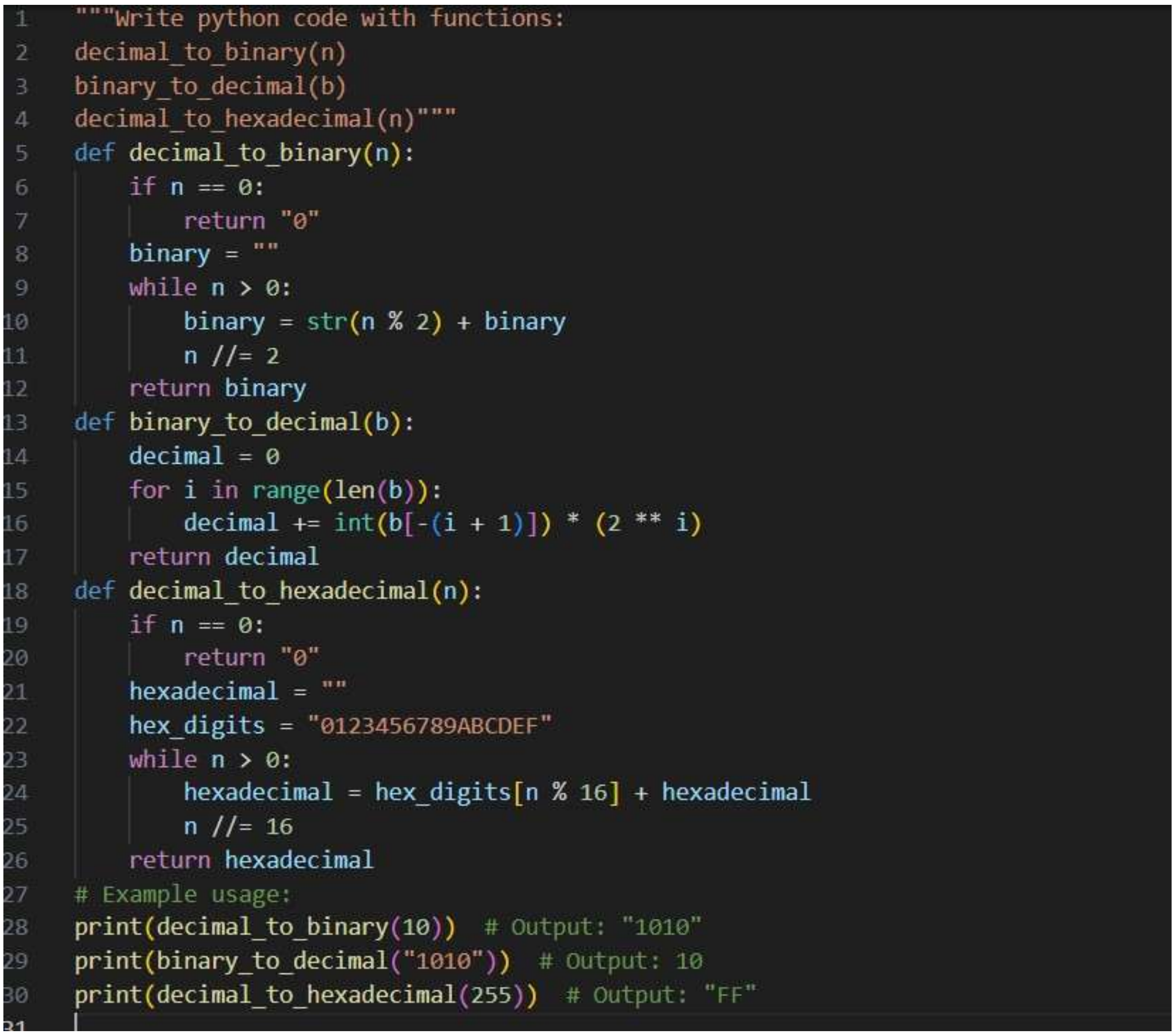
decimal_to_hexadecimal(n)

2. Use Copilot for auto-generating docstrings.

3. Generate documentation in the terminal.

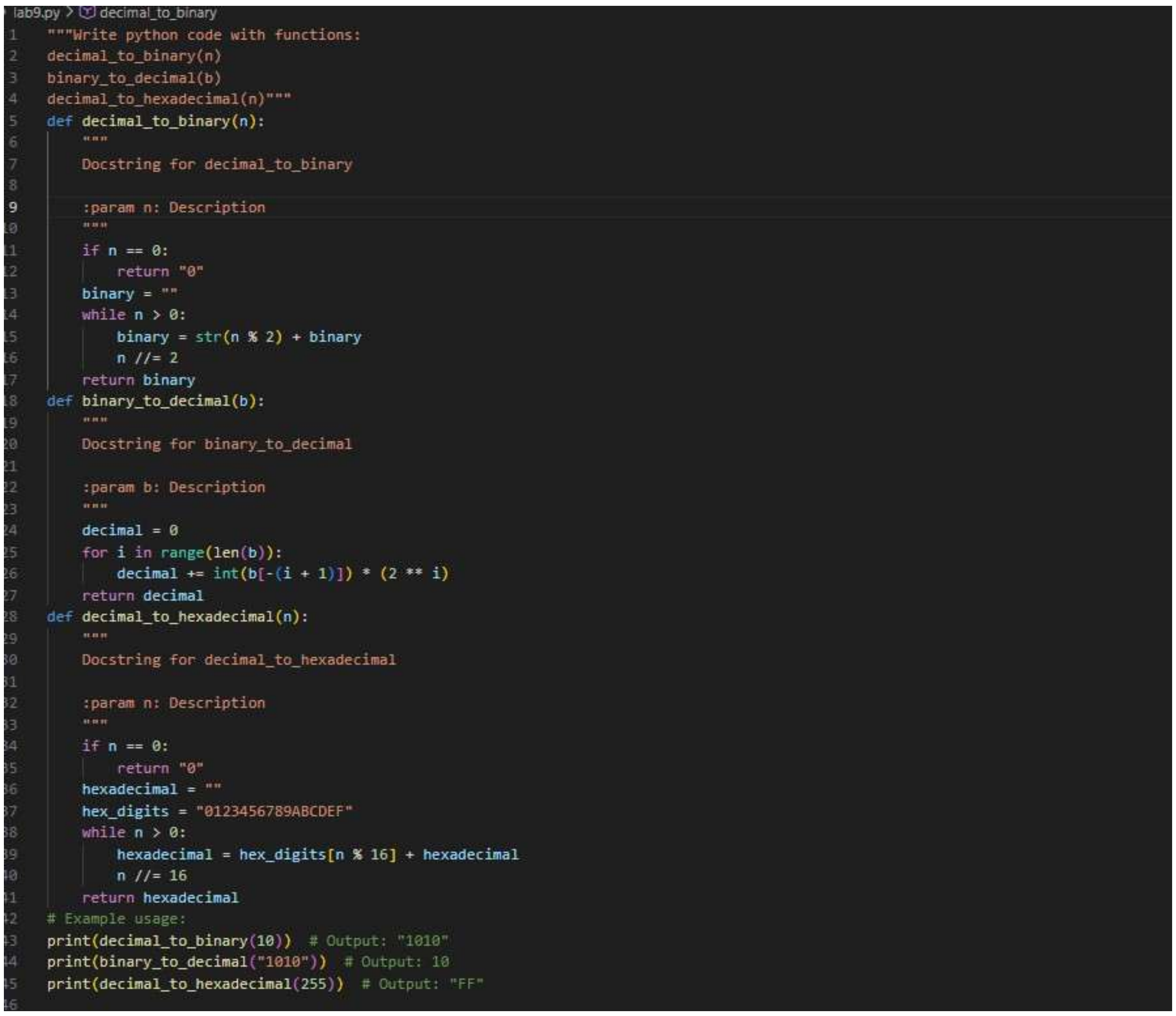
4. Export the documentation in HTML format and open it in a browser.

<u>Given Code:</u>



```python
"""Write python code with functions:
decimal_to_binary(n)
binary_to_decimal(b)
decimal_to_hexadecimal(n)"""
def decimal_to_binary(n):
    if n == 0:
        return "0"
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n //= 2
    return binary
def binary_to_decimal(b):
    decimal = 0
    for i in range(len(b)):
        decimal += int(b[-(i + 1)]) * (2 ** i)
    return decimal
def decimal_to_hexadecimal(n):
    if n == 0:
        return "0"
    hexadecimal = ""
    hex_digits = "0123456789ABCDEF"
    while n > 0:
        hexadecimal = hex_digits[n % 16] + hexadecimal
        n //= 16
    return hexadecimal
# Example usage:
print(decimal_to_binary(10))  # Output: "1010"
print(binary_to_decimal("1010"))  # Output: 10
print(decimal_to_hexadecimal(255))  # Output: "FF"
```

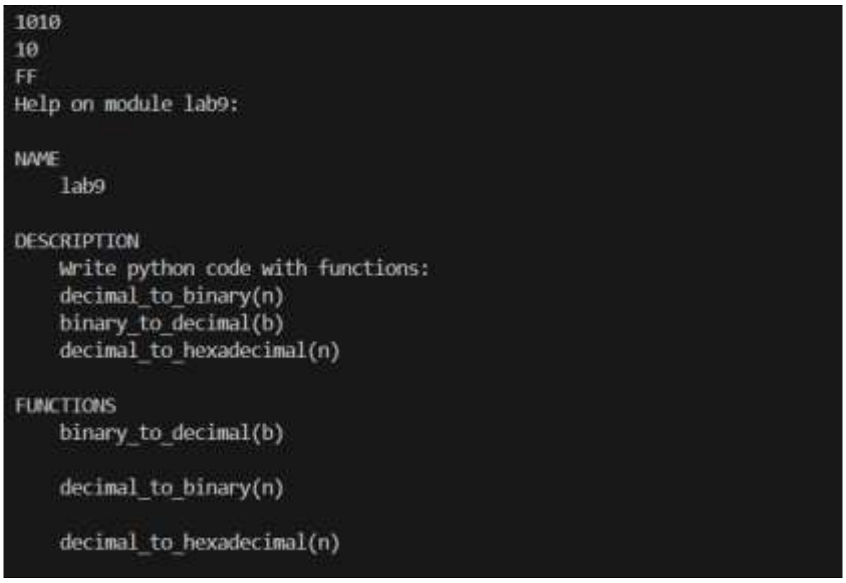
Docstring:



```
lab9.py > decimal_to_binary
"""Write python code with functions:
decimal_to_binary(n)
binary_to_decimal(b)
decimal_to_hexadecimal(n)"""
def decimal_to_binary(n):
    """
    Docstring for decimal_to_binary

    :param n: Description
    """
    if n == 0:
        return "0"
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n //= 2
    return binary
def binary_to_decimal(b):
    """
    Docstring for binary_to_decimal

    :param b: Description
    """
    decimal = 0
    for i in range(len(b)):
        decimal += int(b[-(i + 1)]) * (2 ** i)
    return decimal
def decimal_to_hexadecimal(n):
    """
    Docstring for decimal_to_hexadecimal

    :param n: Description
    """
    if n == 0:
        return "0"
    hexadecimal = ""
    hex_digits = "0123456789ABCDEF"
    while n > 0:
        hexadecimal = hex_digits[n % 16] + hexadecimal
        n //= 16
    return hexadecimal
# Example usage:
print(decimal_to_binary(10))  # Output: "1010"
print(binary_to_decimal("1010"))  # Output: 10
print(decimal_to_hexadecimal(255))  # Output: "FF"
```

Using pydoc

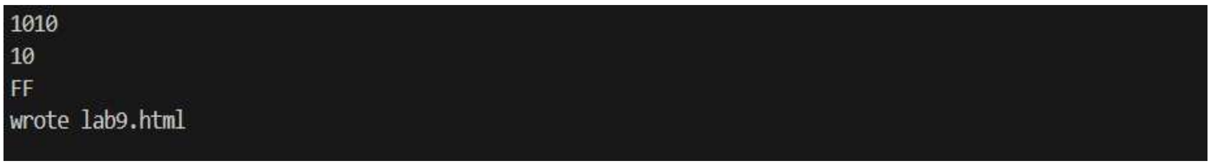
1) python -m pydoc filename



```
1010
10
FF
Help on module lab9:

NAME
    lab9

DESCRIPTION
    Write python code with functions:
    decimal_to_binary(n)
    binary_to_decimal(b)
    decimal_to_hexadecimal(n)

FUNCTIONS
    binary_to_decimal(b)

    decimal_to_binary(n)

    decimal_to_hexadecimal(n)
```

2)python -m pydoc -w filename

3)python -m pydoc -p 8080



```
1010
10
FF
wrote lab9.html
```

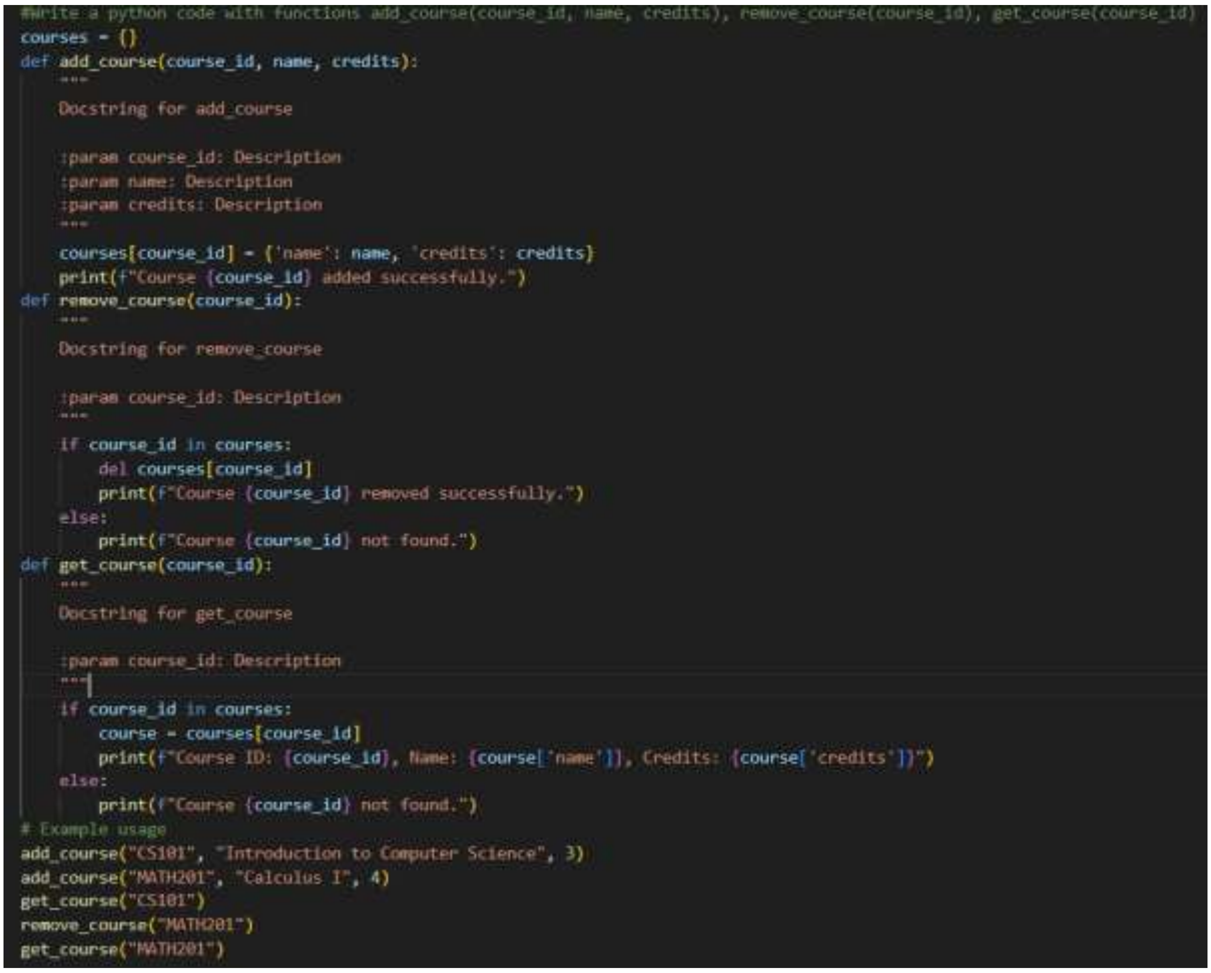
Problem 5 – Course Management Module

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**

Given Code:

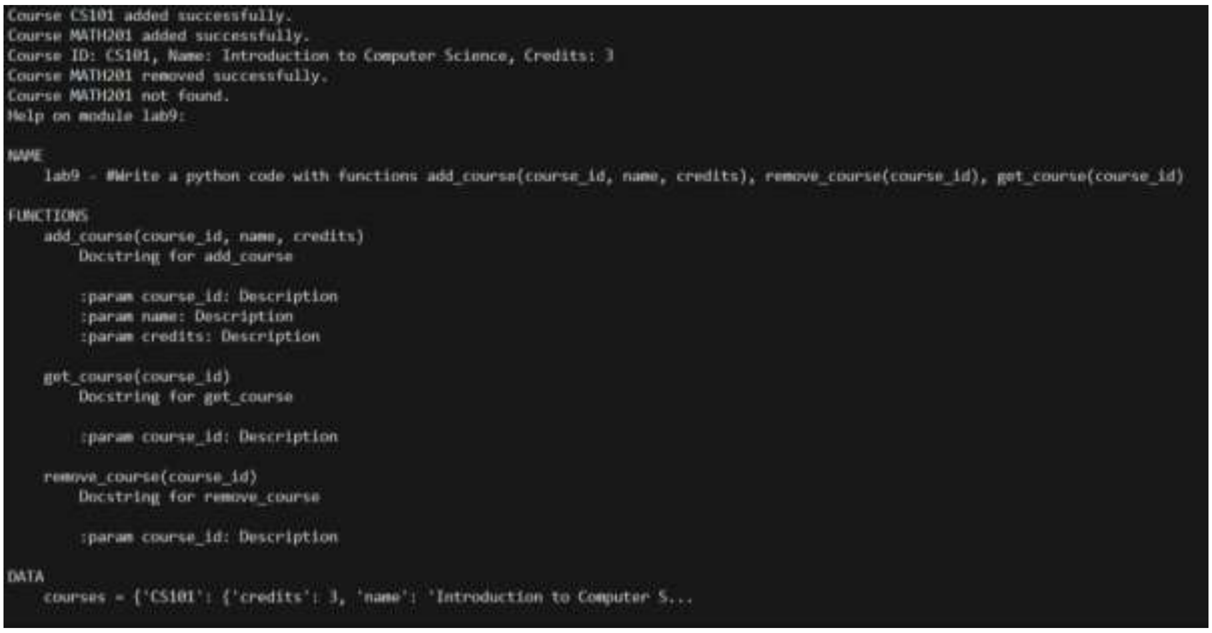
Docstrings:



```python
#Write a python code with functions add_course(course_id, name, credits), remove_course(course_id), get_course(course_id)
courses = {}
def add_course(course_id, name, credits):
    """
    Docstring for add_course

    :param course_id: Description
    :param name: Description
    :param credits: Description
    """
    courses[course_id] = {'name': name, 'credits': credits}
    print(f"Course {course_id} added successfully.")
def remove_course(course_id):
    """
    Docstring for remove_course

    :param course_id: Description
    """
    if course_id in courses:
        del courses[course_id]
        print(f"Course {course_id} removed successfully.")
    else:
        print(f"Course {course_id} not found.")
def get_course(course_id):
    """
    Docstring for get_course

    :param course_id: Description
    """
    if course_id in courses:
        course = courses[course_id]
        print(f"Course ID: {course_id}, Name: {course['name']}, Credits: {course['credits']}")
    else:
        print(f"Course {course_id} not found.")
# Example usage
add_course("CS101", "Introduction to Computer Science", 3)
add_course("MATH201", "Calculus I", 4)
get_course("CS101")
remove_course("MATH201")
get_course("MATH201")
```

Using pydoc

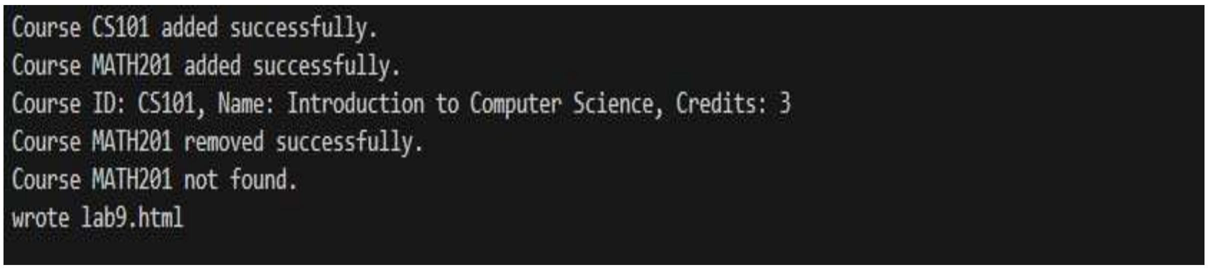
1) python -m pydoc filename



```
Course CS101 added successfully.
Course MATH201 added successfully.
Course ID: CS101, Name: Introduction to Computer Science, Credits: 3
Course MATH201 removed successfully.
Course MATH201 not found.
Help on module lab9:

NAME
    lab9 - #Write a python code with functions add_course(course_id, name, credits), remove_course(course_id), get_course(course_id)

FUNCTIONS
    add_course(course_id, name, credits)
        Docstring for add_course

        :param course_id: Description
        :param name: Description
        :param credits: Description

    get_course(course_id)
        Docstring for get_course

        :param course_id: Description

    remove_course(course_id)
        Docstring for remove_course

        :param course_id: Description

DATA
    courses = {'CS101': {'credits': 3, 'name': 'Introduction to Computer S...
```

2)python -m pydoc -w filename

3)python -m pydoc -p 8080



```
Course CS101 added successfully.
Course MATH201 added successfully.
Course ID: CS101, Name: Introduction to Computer Science, Credits: 3
Course MATH201 removed successfully.
Course MATH201 not found.
wrote lab9.html
```