Lab Assignment 9.1

NAME: K. SRUTHAKEERTHI
HALLTICKETNO: 2303A51022
BATCH-01

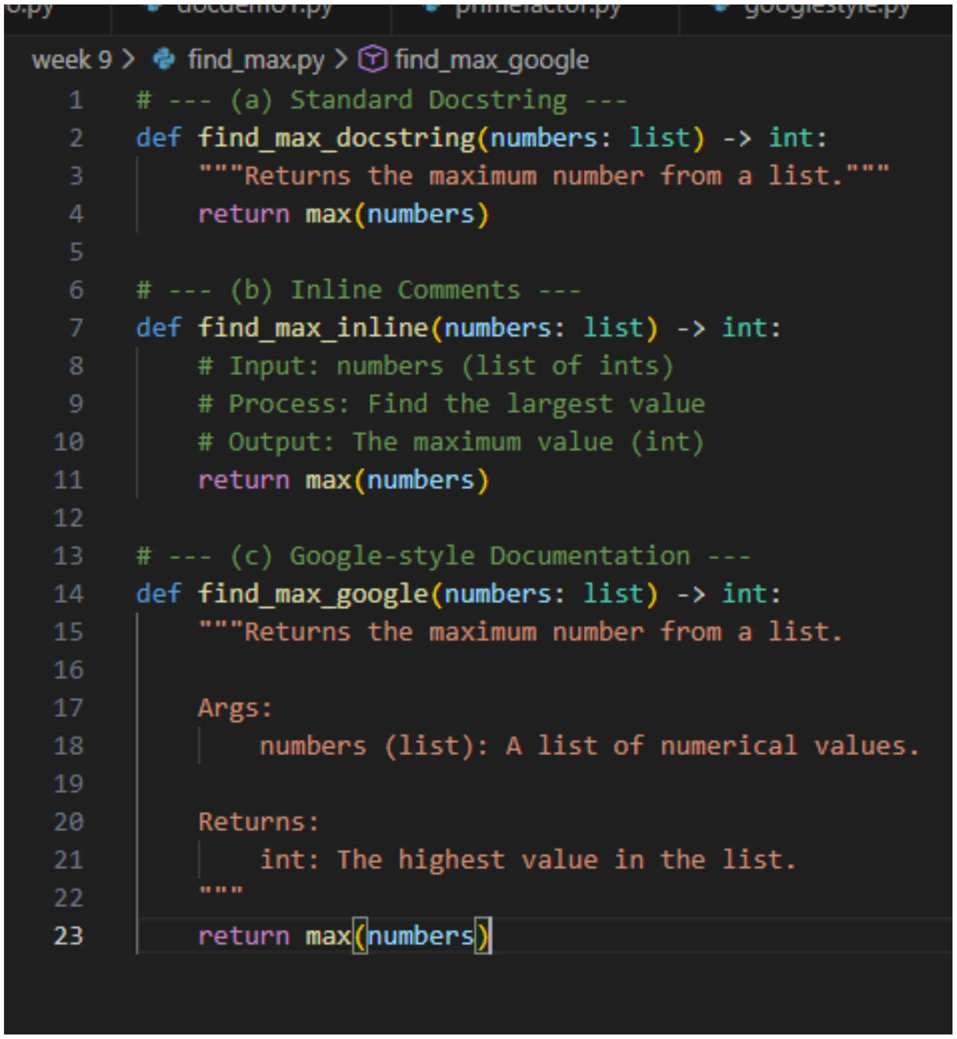
Problem 1:

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.



week 9 > find_max.py > find_max_google

```python
# --- (a) Standard Docstring ---
def find_max_docstring(numbers: list) -> int:
    """Returns the maximum number from a list."""
    return max(numbers)

# --- (b) Inline Comments ---
def find_max_inline(numbers: list) -> int:
    # Input: numbers (list of ints)
    # Process: Find the largest value
    # Output: The maximum value (int)
    return max(numbers)

# --- (c) Google-style Documentation ---
def find_max_google(numbers: list) -> int:
    """Returns the maximum number from a list.

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

    Returns:
        int: The highest value in the list.
    """
    return max(numbers)
```

Style	Advantages	Disadvantages	Use Case
Docstring	Easy to read, built-in Python support	Less structured	Small to medium projects
Inline Comments	Very simple	Not suitable for functions	Explaining logic
Google-Style	Professional, standardized	Slightly longer	Libraries & APIs

Google-style documentation is most effective for a **mathematical utilities library** because it is clear, structured, and widely used in professional projects

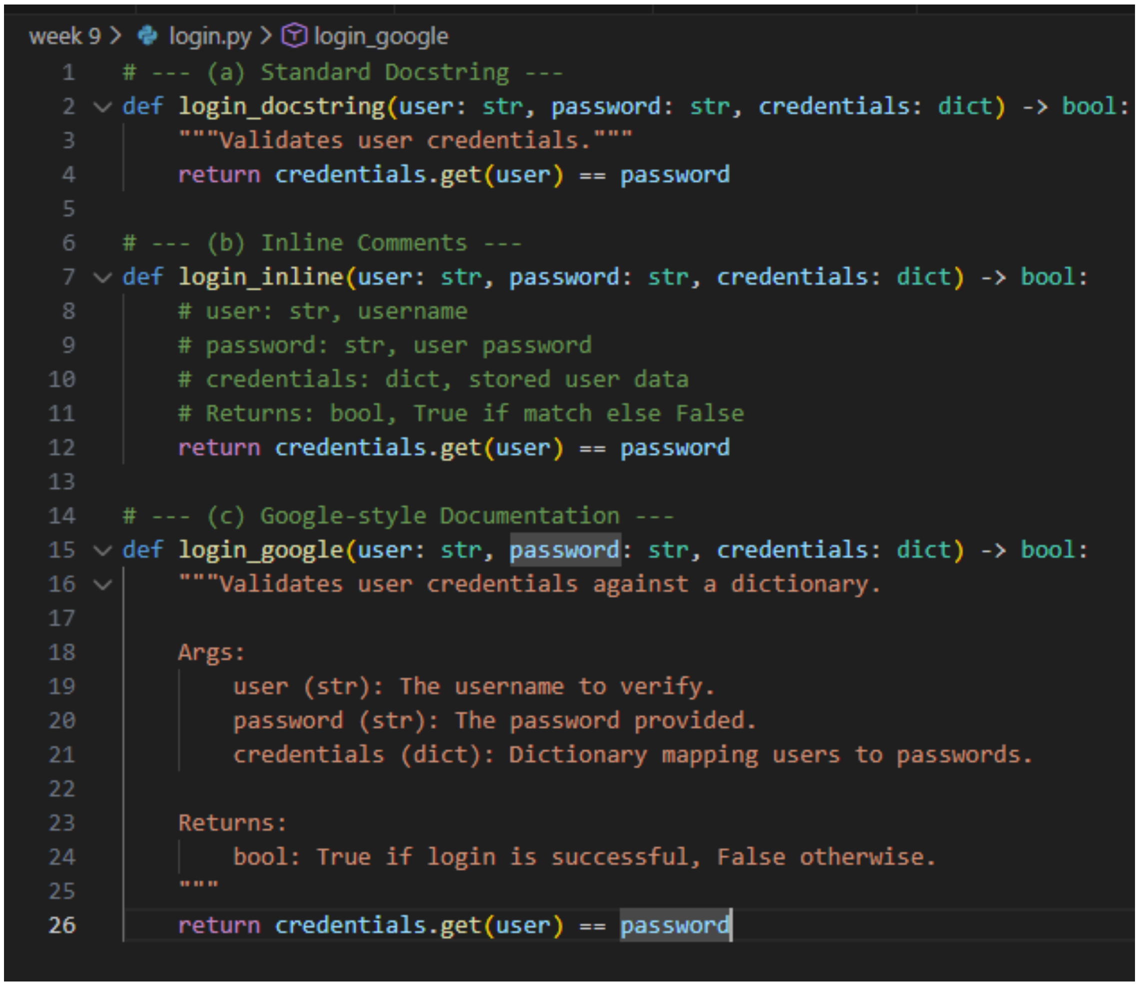
Problem 2: Consider the following Python function:

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



week 9 > login.py > login_google

```python
# --- (a) Standard Docstring ---
def login_docstring(user: str, password: str, credentials: dict) -> bool:
    """Validates user credentials."""
    return credentials.get(user) == password

# --- (b) Inline Comments ---
def login_inline(user: str, password: str, credentials: dict) -> bool:
    # user: str, username
    # password: str, user password
    # credentials: dict, stored user data
    # Returns: bool, True if match else False
    return credentials.get(user) == password

# --- (c) Google-style Documentation ---
def login_google(user: str, password: str, credentials: dict) -> bool:
    """Validates user credentials against a dictionary.

    Args:
        user (str): The username to verify.
        password (str): The password provided.
        credentials (dict): Dictionary mapping users to passwords.

    Returns:
        bool: True if login is successful, False otherwise.
    """
    return credentials.get(user) == password
```

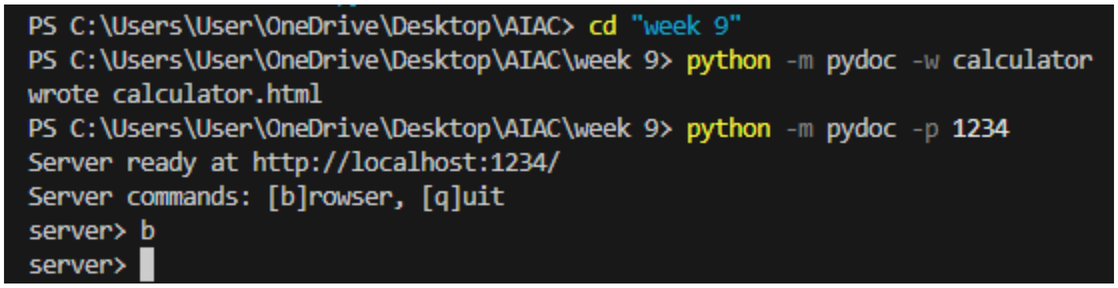
Google-style documentation is best for **new developers onboarding** because it clearly explain parameters, return values, and usage.

Problem 3: Calculator (Automatic Documentation Generation)

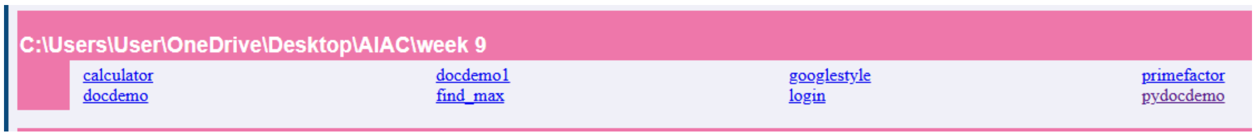
Task: Design a Python module named calculator.py and demonstrate automatic documentation generation.

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
   - subtract(a, b) – returns the difference of two numbers
   - multiply(a, b) – returns the product of two numbers
   - 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.

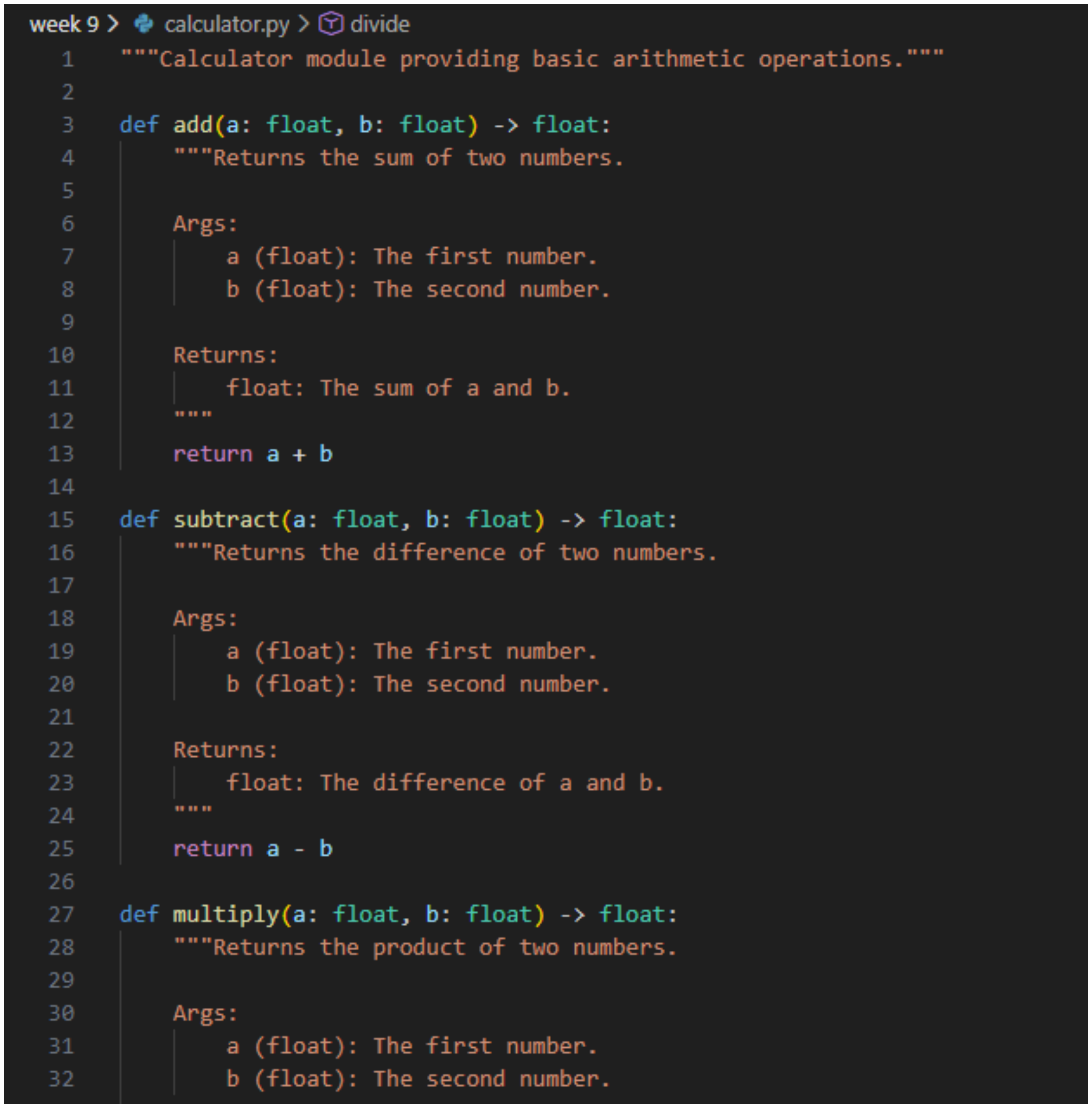


```
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -w calculator
wrote calculator.html
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
server>
```

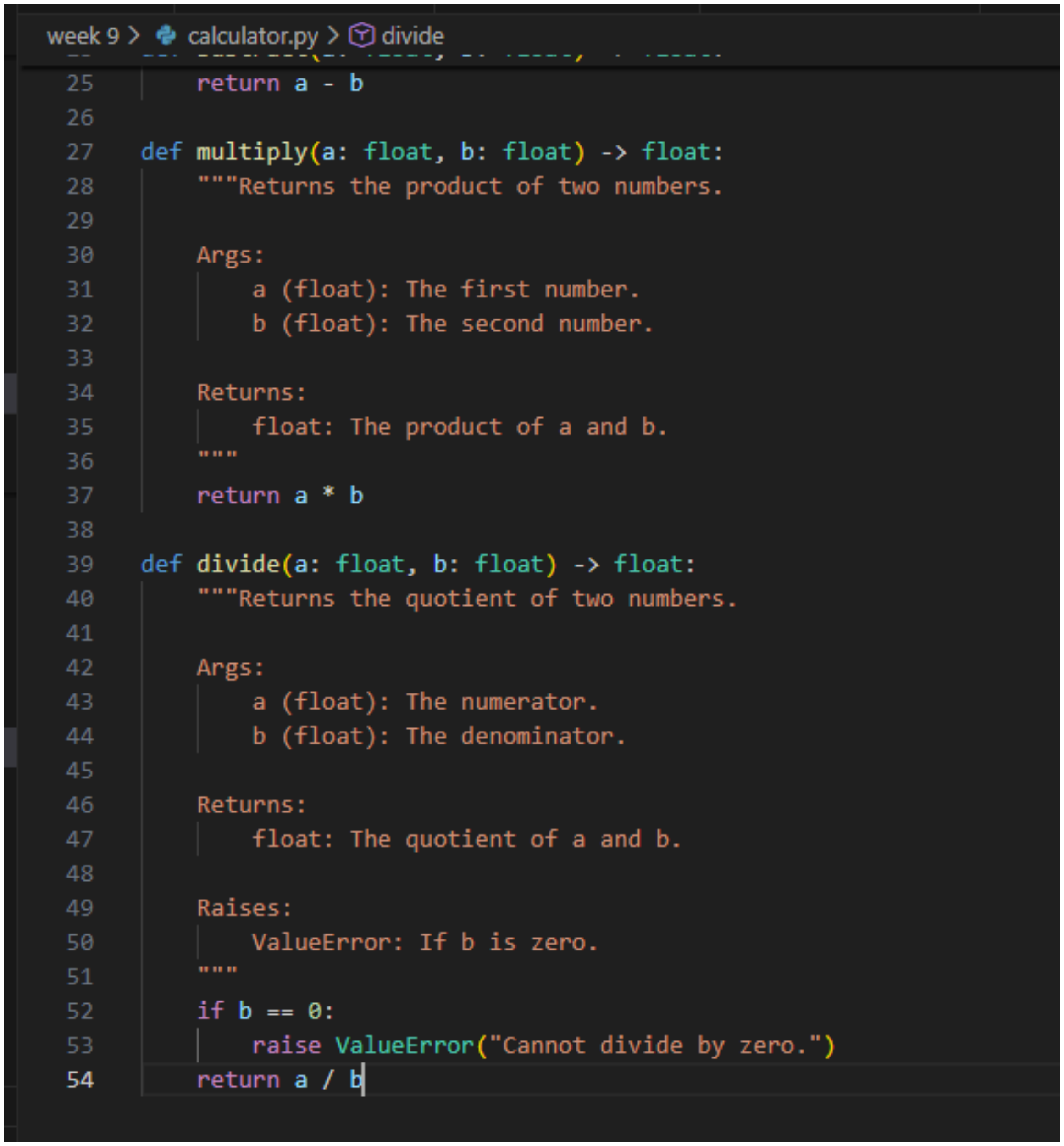


C:\Users\User\OneDrive\Desktop\AIAC\week 9

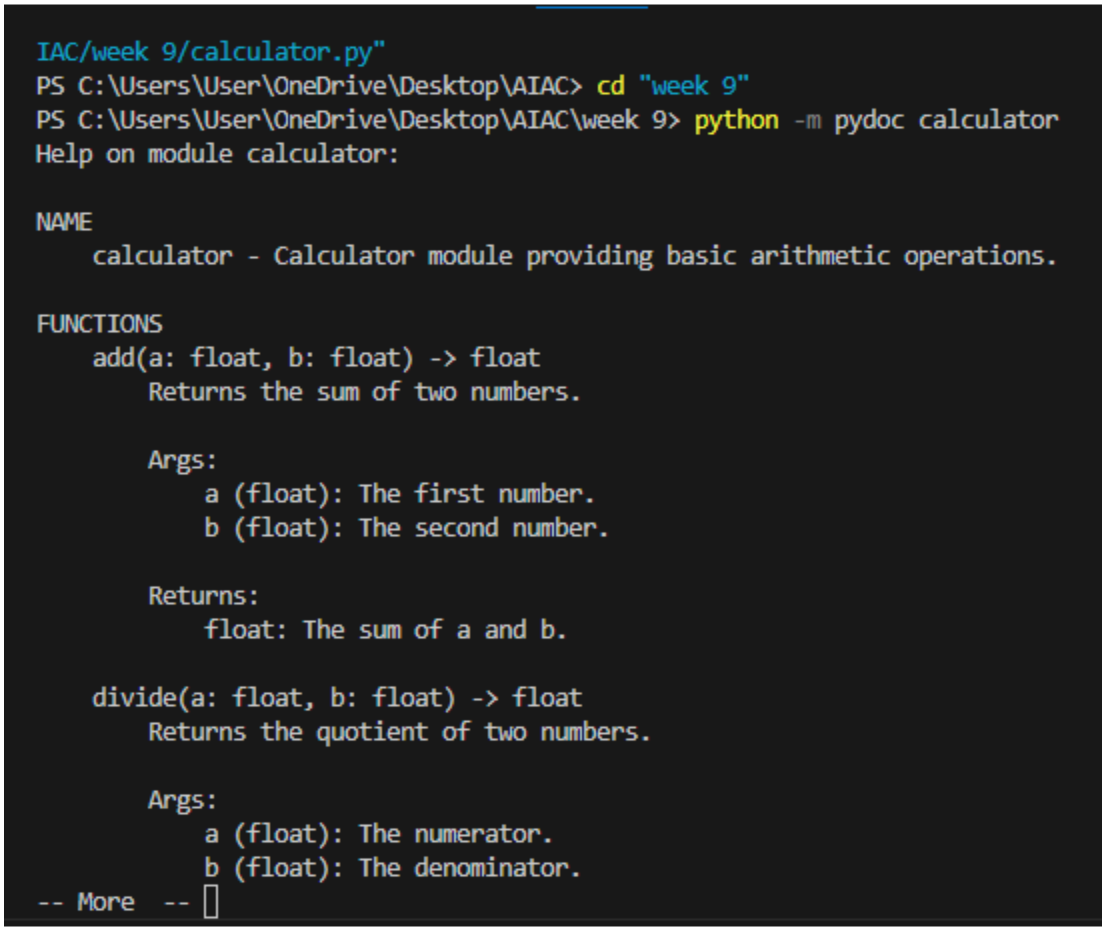
calculator	docdemo1	googlestyle	primefactor
docdemo	find_max	login	pydocdemo


week 9 > calculator.py > divide

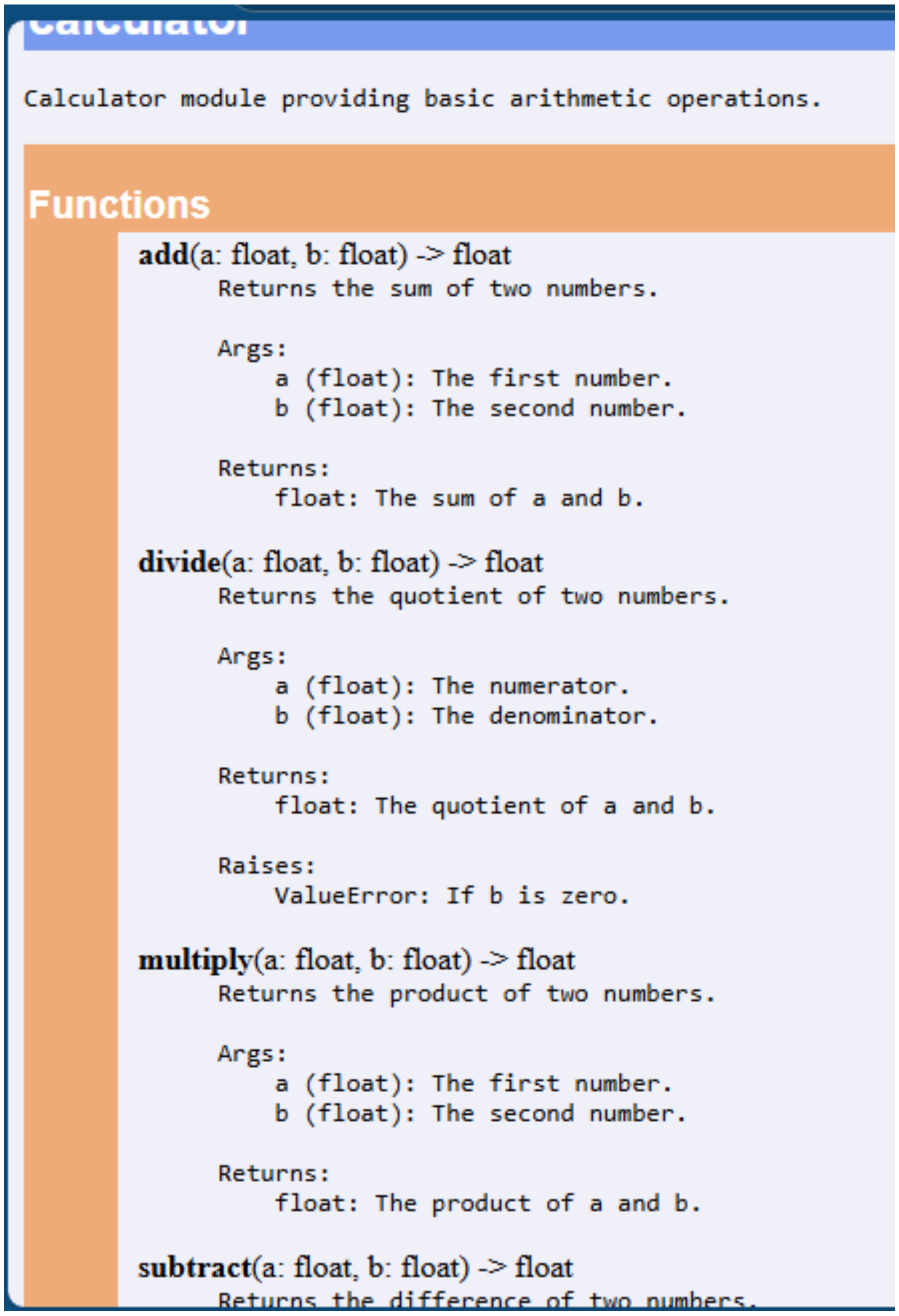
```python
"""Calculator module providing basic arithmetic operations."""

def add(a: float, b: float) -> float:
    """Returns the sum of two numbers.

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

    Returns:
        float: The sum of a and b.
    """
    return a + b

def subtract(a: float, b: float) -> float:
    """Returns the difference of two numbers.

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

    Returns:
        float: The difference of a and b.
    """
    return a - b

def multiply(a: float, b: float) -> float:
    """Returns the product of two numbers.

    Args:
        a (float): The first number.
        b (float): The second number.
```



```python
    return a - b

def multiply(a: float, b: float) -> float:
    """Returns the product of two numbers.

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

    Returns:
        float: The product of a and b.
    """
    return a * b

def divide(a: float, b: float) -> float:
    """Returns the quotient of two numbers.

    Args:
        a (float): The numerator.
        b (float): The denominator.

    Returns:
        float: The quotient of a and b.

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



```
IAC/week 9/calculator.py"
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc calculator
Help on module calculator:

NAME
    calculator - Calculator module providing basic arithmetic operations.

FUNCTIONS
    add(a: float, b: float) -> float
        Returns the sum of two numbers.

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

        Returns:
            float: The sum of a and b.

    divide(a: float, b: float) -> float
        Returns the quotient of two numbers.

        Args:
            a (float): The numerator.
            b (float): The denominator.
-- More  --
```



Calculator module providing basic arithmetic operations.

Functions

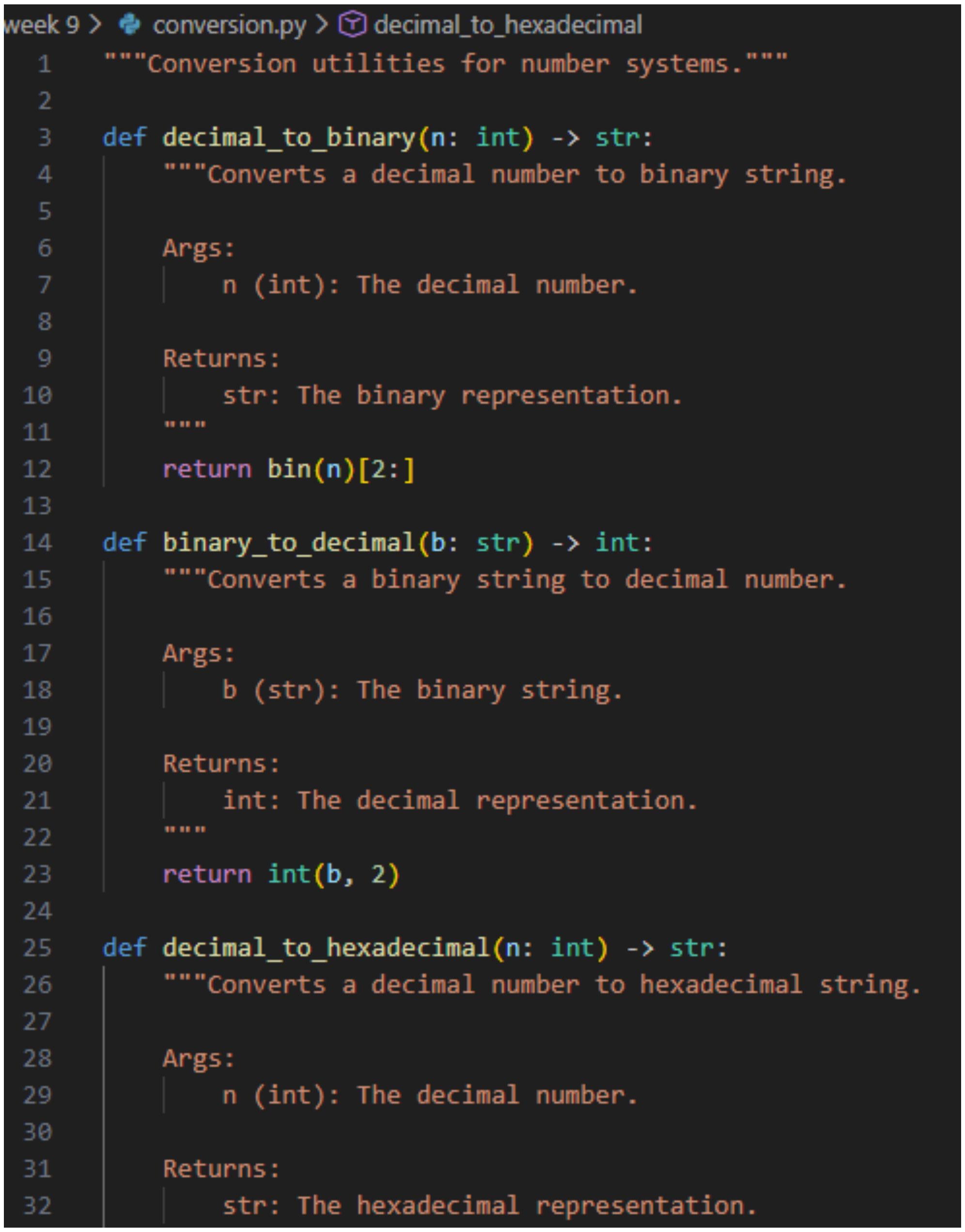
```
add(a: float, b: float) -> float
    Returns the sum of two numbers.

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

    Returns:
        float: The sum of a and b.

divide(a: float, b: float) -> float
    Returns the quotient of two numbers.

    Args:
        a (float): The numerator.
        b (float): The denominator.

    Returns:
        float: The quotient of a and b.

    Raises:
        ValueError: If b is zero.

multiply(a: float, b: float) -> float
    Returns the product of two numbers.

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

    Returns:
        float: The product of a and b.

subtract(a: float, b: float) -> float
    Returns the difference of two numbers.
```

Problem 4: Conversion Utilities Module

Task:

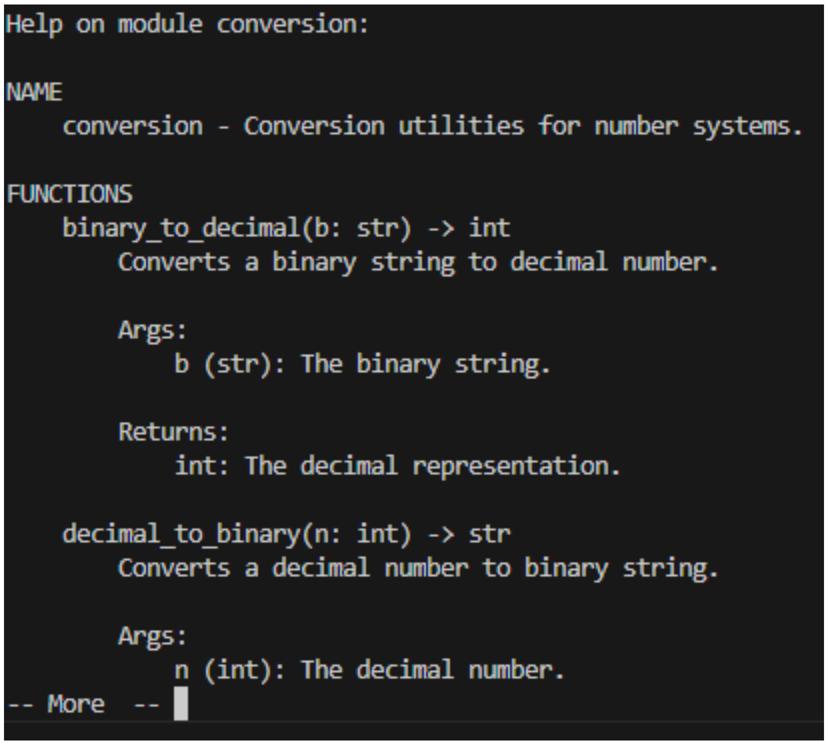
1. Write a module named conversion.py with functions:
    - decimal_to_binary(n)
    - binary_to_decimal(b)
    - decimal_to_hexadecimal(n)

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.

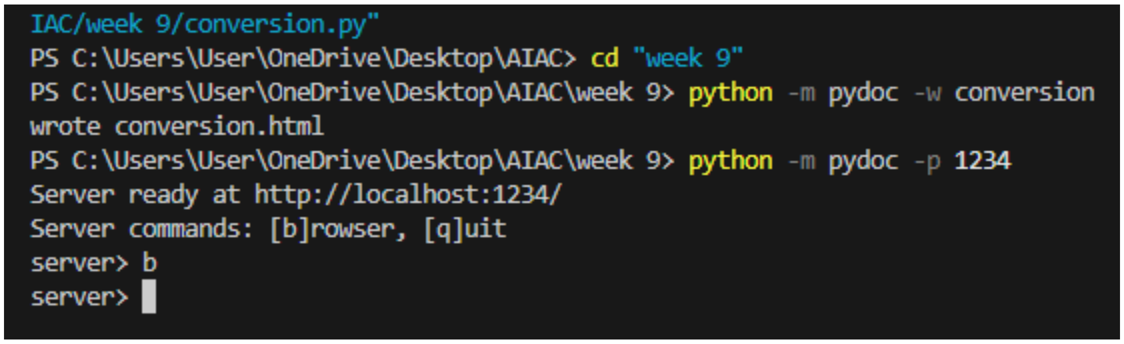


week 9 > conversion.py > decimal_to_hexadecimal

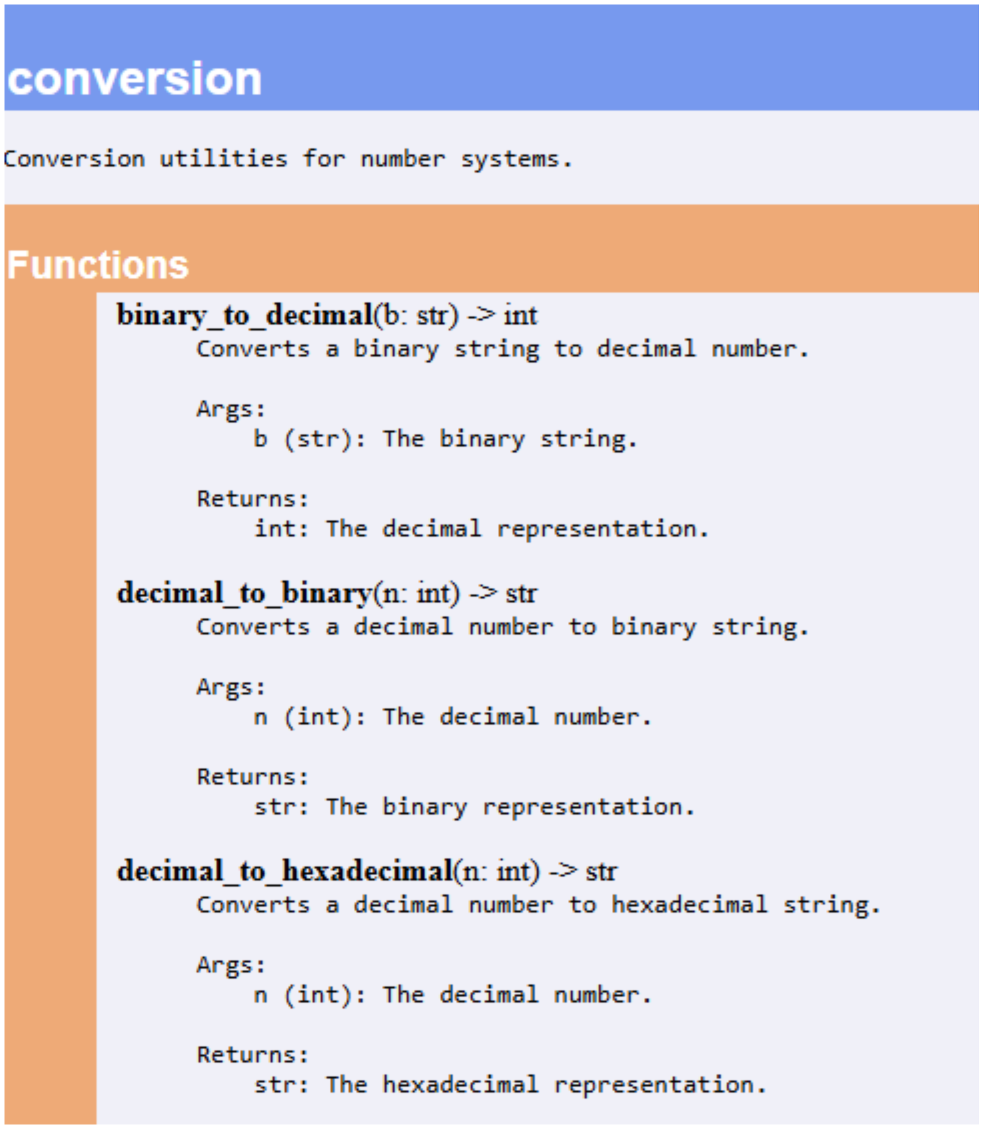
```python
"""Conversion utilities for number systems."""

def decimal_to_binary(n: int) -> str:
    """Converts a decimal number to binary string.

    Args:
        n (int): The decimal number.

    Returns:
        str: The binary representation.
    """
    return bin(n)[2:]

def binary_to_decimal(b: str) -> int:
    """Converts a binary string to decimal number.

    Args:
        b (str): The binary string.

    Returns:
        int: The decimal representation.
    """
    return int(b, 2)

def decimal_to_hexadecimal(n: int) -> str:
    """Converts a decimal number to hexadecimal string.

    Args:
        n (int): The decimal number.

    Returns:
        str: The hexadecimal representation.
```



```
Help on module conversion:

NAME
    conversion - Conversion utilities for number systems.

FUNCTIONS
    binary_to_decimal(b: str) -> int
        Converts a binary string to decimal number.

        Args:
            b (str): The binary string.

        Returns:
            int: The decimal representation.

    decimal_to_binary(n: int) -> str
        Converts a decimal number to binary string.

        Args:
            n (int): The decimal number.
-- More  --
```



```
IAC/week 9/conversion.py"
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -w conversion
wrote conversion.html
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
server>
```


conversion

```
Conversion utilities for number systems.
```

Functions

binary_to_decimal(b: str) -> int
```
    Converts a binary string to decimal number.

    Args:
        b (str): The binary string.

    Returns:
        int: The decimal representation.
```

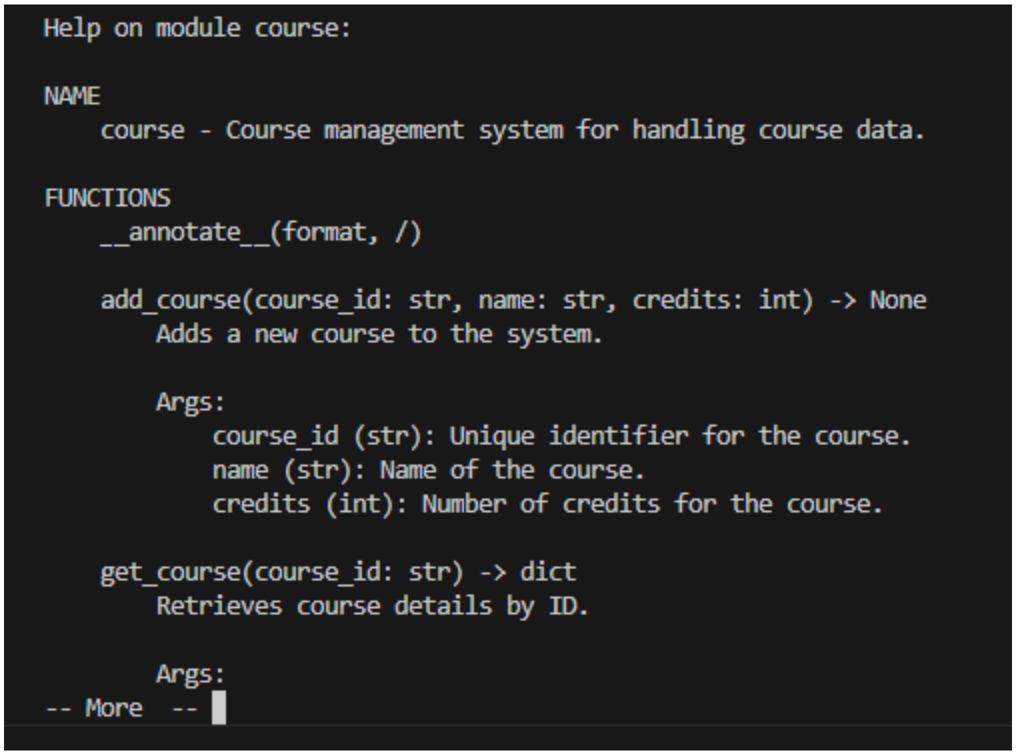
decimal_to_binary(n: int) -> str
```
    Converts a decimal number to binary string.

    Args:
        n (int): The decimal number.

    Returns:
        str: The binary representation.
```

decimal_to_hexadecimal(n: int) -> str
```
    Converts a decimal number to hexadecimal string.

    Args:
        n (int): The decimal number.

    Returns:
        str: The hexadecimal representation.
```

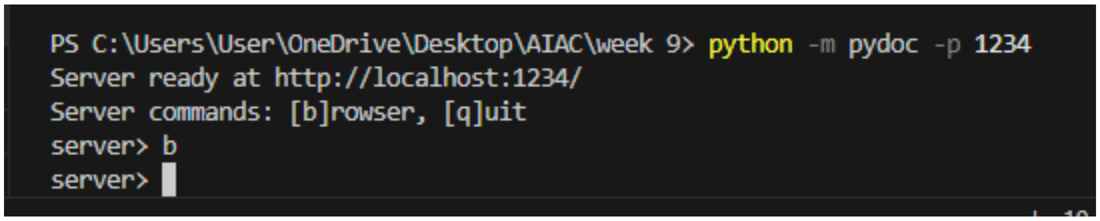
Problem 5 – Course Management Module

Task:

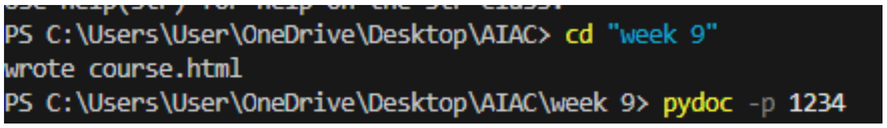
1. Create a module course.py with functions:
   - o add_course(course_id, name, credits)
   - o remove_course(course_id)
   - o get_course(course_id)
2. Add docstrings with Copilot.
3. Generate documentation in the terminal.
4. Export the documentation in HTML format and open it in a browser



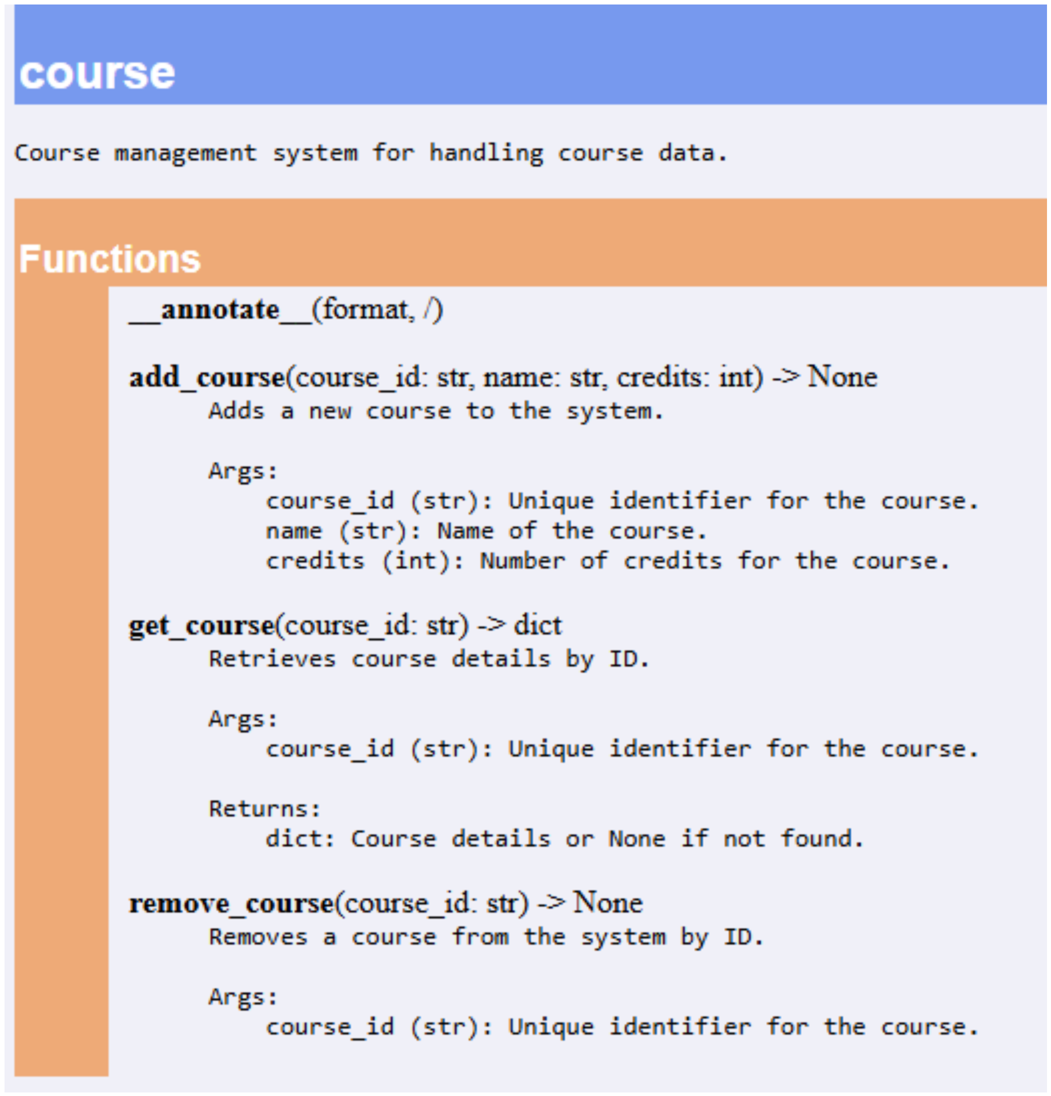
```
Help on module course:

NAME
    course - Course management system for handling course data.

FUNCTIONS
    __annotate__(format, /)

    add_course(course_id: str, name: str, credits: int) -> None
        Adds a new course to the system.

        Args:
            course_id (str): Unique identifier for the course.
            name (str): Name of the course.
            credits (int): Number of credits for the course.

    get_course(course_id: str) -> dict
        Retrieves course details by ID.

        Args:
-- More  --
```



```
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
server>
```



```
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
wrote course.html
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> pydoc -p 1234
```



course

Course management system for handling course data.

Functions

__annotate__(format, /)

add_course(course_id: str, name: str, credits: int) -> None

```
Adds a new course to the system.

Args:
    course_id (str): Unique identifier for the course.
    name (str): Name of the course.
    credits (int): Number of credits for the course.
```

get_course(course_id: str) -> dict

```
Retrieves course details by ID.

Args:
    course_id (str): Unique identifier for the course.

Returns:
    dict: Course details or None if not found.
```

remove_course(course_id: str) -> None

```
Removes a course from the system by ID.

Args:
    course_id (str): Unique identifier for the course.
```