Assignment -9.1

Hallticket no:2303A510A6

Batch no:02

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.

Code:

C: > Users > shyam > ai assistes code > ass16ai.py

```python
def find_max(numbers):
    """
    Returns the maximum value from a list of numbers.

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

    Returns:
        int or float: The largest value in the list.

    Raises:
        ValueError: If the list is empty."""
def find_max(numbers):
    # Check if the list is empty
    if not numbers:
        # Raise an error if empty
        raise ValueError("The input list cannot be empty.")

    # Return the maximum value using Python's built-in max()
    return max(numbers)

def find_max(numbers):
    """Returns the maximum value from a list of numbers.

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

    Returns:
        int or float: The largest value in the list.

    Raises:
        ValueError: If the list is empty.
    """
    if not numbers:
        raise ValueError("The list cannot be empty.")
    return max(numbers)
```

1) Docstring

Advantages:

- Explains what the function does.
- Can be viewed using help() in Python.
- Easy to write and understand.

Disadvantages:

- Not always structured.
- Can be inconsistent if different people write it differently.
- **2) Inline Comments**
- **Advantages:**
- Explain how the code works step by step.
- Helpful for understanding complex logic.
- **Disadvantages:**

- Cannot be accessed using documentation tools.
- Can make code messy if overused.

3) Google-Style Documentation

Advantages:

- Clear and well-structured format.
- Separates Args, Returns, and Raises.
- Good for large projects.

Disadvantages:

- Slightly longer to write.
- May be unnecessary for very small functions.

or a mathematical utilities library, **Google-style documentation is the most effective** because it clearly explains inputs, outputs, and errors in a structured way. Mathematical functions require precision, so a clear and consistent format is important.

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.

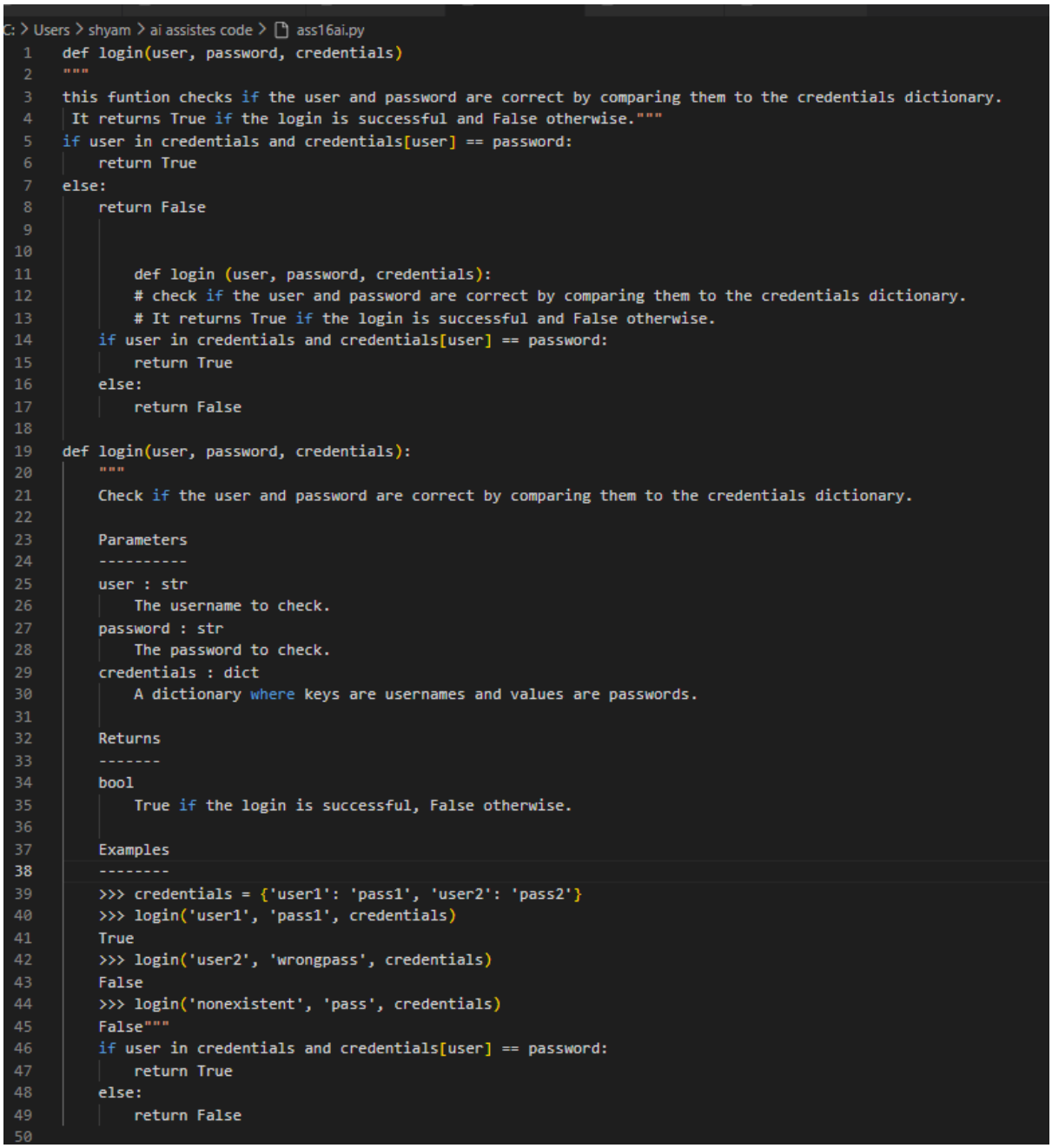
2. Critically compare the approaches.

3. Recommend which style would be most helpful for new

developers onboarding a project, and justify your choice.

Code:

For new developers onboarding a project, Google-style documentation is the most helpful.



```
C: > Users > shyam > ai assistes code > ass16ai.py
def login(user, password, credentials)
"""
this funtion checks if the user and password are correct by comparing them to the credentials dictionary.
 It returns True if the login is successful and False otherwise."""
if user in credentials and credentials[user] == password:
    return True
else:
    return False



    def login (user, password, credentials):
    # check if the user and password are correct by comparing them to the credentials dictionary.
    # It returns True if the login is successful and False otherwise.
    if user in credentials and credentials[user] == password:
        return True
    else:
        return False

def login(user, password, credentials):
    """
    Check if the user and password are correct by comparing them to the credentials dictionary.

    Parameters
    ----------
    user : str
        The username to check.
    password : str
        The password to check.
    credentials : dict
        A dictionary where keys are usernames and values are passwords.

    Returns
    -------
    bool
        True if the login is successful, False otherwise.

    Examples
    --------
    >>> credentials = {'user1': 'pass1', 'user2': 'pass2'}
    >>> login('user1', 'pass1', credentials)
    True
    >>> login('user2', 'wrongpass', credentials)
    False
    >>> login('nonexistent', 'pass', credentials)
    False"""
    if user in credentials and credentials[user] == password:
        return True
    else:
        return False

```

For new developers onboarding a project, Google-style documentation is the most helpful.

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.

Code:

> Users > shyam > ai assistes code > ass1bai.html

```
"""
calculator.py
A simple calculator module that provides basic arithmetic operations:
addition, subtraction, multiplication, and division."""

def add(a, b):
    """Return the sum of two numbers."""
    return a + b


def subtract(a, b):
    """Return the difference of two numbers."""
    return a - b


def multiply(a, b):
    """Return the product of two numbers."""
    return a * b


def divide(a, b):
    """
    Return the quotient of two numbers.

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

Display the module documentation in the terminal using

Python’s documentation tools.

```
PS C:\Users\shyam\ai assistes code> python -m pydoc .\ass16ai.html
problem in .\ass16ai.html - SyntaxError: invalid syntax (ass16ai.html, line 1)
PS C:\Users\shyam\ai assistes code> python -m pydoc .\ass16ai.html
Help on module ass16ai:

NAME
    ass16ai

DESCRIPTION
    calculator.py
    A simple calculator module that provides basic arithmetic operations:
    addition, subtraction, multiplication, and division.

FUNCTIONS
    add(a, b)
        Return the sum of two numbers.

    divide(a, b)
        Return the quotient of two numbers.

        Raises:
            ZeroDivisionError: If b is zero.

    multiply(a, b)
        Return the product of two numbers.

    subtract(a, b)
        Return the difference of two numbers.

FILE
    c:\users\shyam\ai assistes code\ass16ai.html
```

Generate and export the module documentation in HTML

format using the pydoc utility, a

```
PS C:\Users\shyam\ai assistes code> python -m pydoc -w .\ass16ai.html
wrote ass16ai.html
```

index
ass16ai c:\users\shyam\ai assistes code\ass16ai.html

calculator.py
A simple calculator module that provides basic arithmetic operations:
addition, subtraction, multiplication, and division.

Functions

add(a, b)
Return the sum of two numbers.

divide(a, b)
Return the quotient of two numbers.

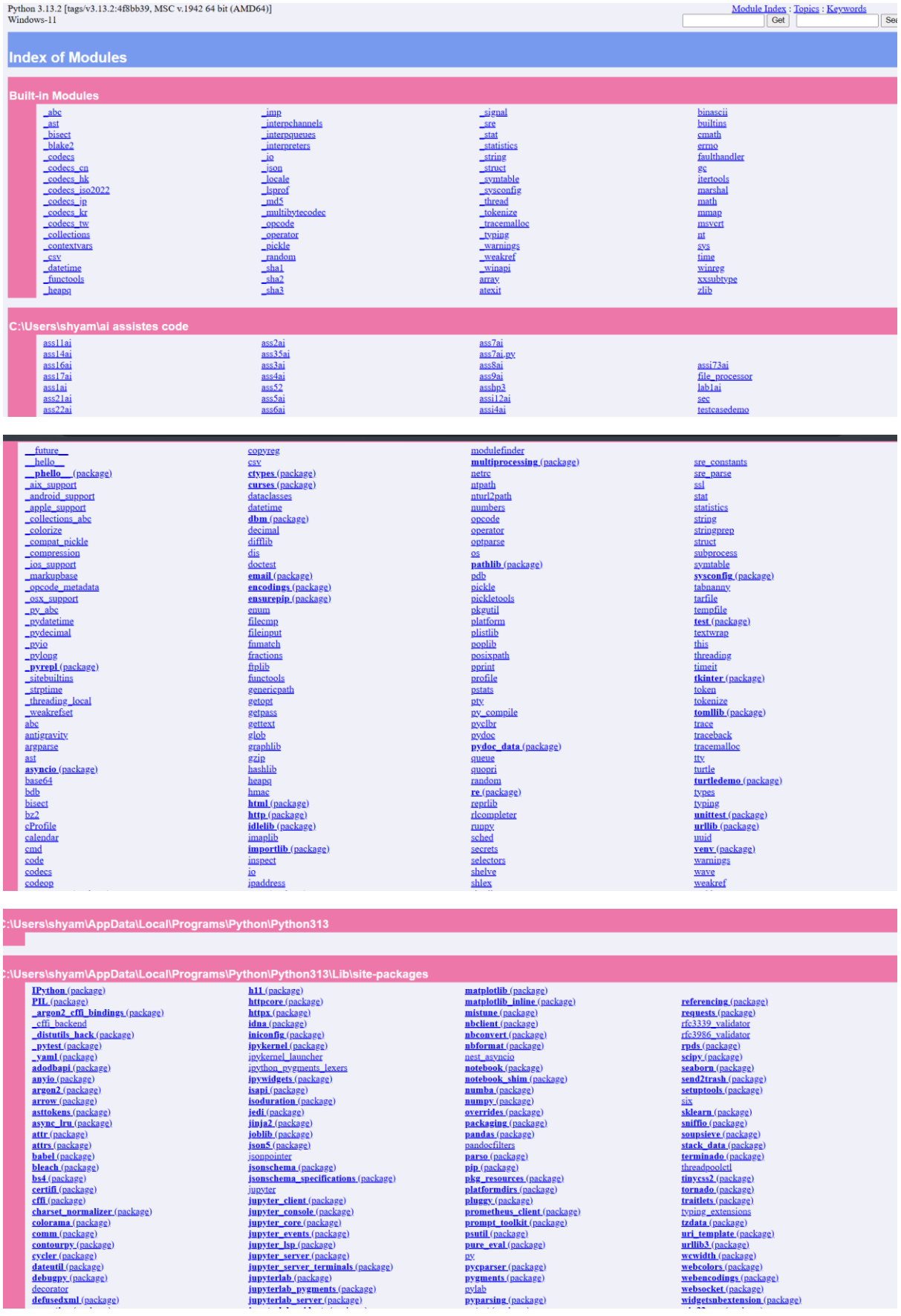
Raises:
ZeroDivisionError: If b is zero.

multiply(a, b)
Return the product of two numbers.

subtract(a, b)
Return the difference of two numbers.

Webserver:

```
S C:\Users\shyam\ai assistes code> python -m pydoc -p1234
erver ready at http://localhost:1234/
erver commands: [b]rowser, [q]uit
erver> b
erver>
```



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 Browse

Code:

terminal

```
ncher' '49771' '--' 'C:\Users\shyam\ai assistes code\ass16ai.py'
PS C:\Users\shyam\ai assistes code> python -m pydoc .\ass16ai.py
Help on module ass16ai:

NAME
    ass16ai

DESCRIPTION
    conversion.py
    A module that provides number conversion utilities.

FUNCTIONS
    binary_to_decimal(b)
        Convert a binary number to its decimal representation.

        Parameters:
            b (str): The binary number to be converted.

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

html:

```
PS C:\Users\shyam\ai assistes code>
PS C:\Users\shyam\ai assistes code> python -m pydoc -w .\ass16ai.py
wrote ass16ai.html
```

index
ass16ai c:\users\shyam\ai assistes code\ass16ai.py

conversion.py
A module that provides number conversion utilities.

Functions

binary_to_decimal(b)
Convert a binary number to its decimal representation.

Parameters:
b (str): The binary number to be converted.

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

Example:
>>> binary_to_decimal('1010')
10

decimal_to_binary(n)
Convert a decimal number to its binary representation.

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

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

Example:
>>> decimal_to_binary(10)
'1010'

decimal_to_hexadecimal(n)
Convert a decimal number to its hexadecimal representation.

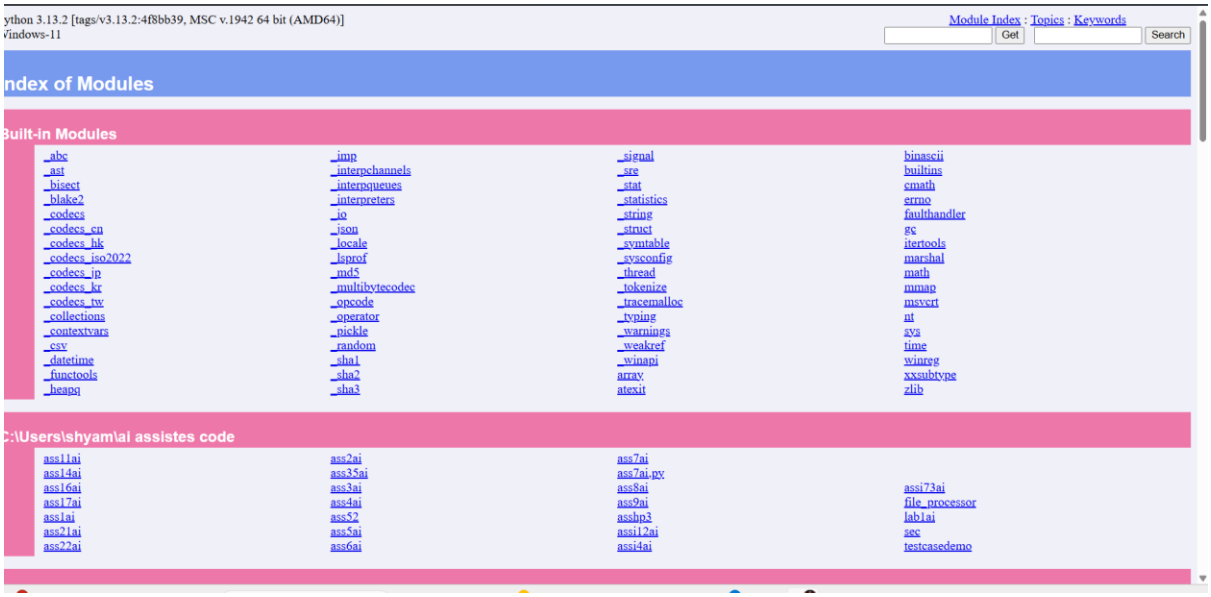
Parameters:
n (int): The decimal number to be converted.

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

Example:
>>> decimal_to_hexadecimal(255)
'ff'

Webbrowser:

```
S C:\Users\shyam\ai assistes code> python -m pydoc -p1234
erver ready at http://localhost:1234/
erver commands: [b]rowser, [q]uit
erver> b
erver>
```



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.

Code:

C: > Users > shyam > ai assistes code > ass16ai.py

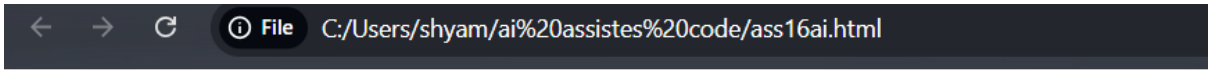
```python
"""
course.py
A simple course management module that allows adding,
removing, and retrieving course information.
"""
# Dictionary to store courses
courses = {}

def add_course(course_id, name, credits):
    """
    Add a new course to the course dictionary.

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

    Returns:
        bool: True if course is added successfully.
    """
    courses[course_id] = {
        "name": name,
        "credits": credits
    }
    return True

def remove_course(course_id):
    """
    Remove a course from the course dictionary.

    Args:
        course_id (str): Unique course identifier.

    Returns:
        bool: True if course is removed, False if not found.
    """
    if course_id in courses:
        del courses[course_id]
        return True
    return False

def get_course(course_id):
    """
    Retrieve course details.

    Args:
        course_id (str): Unique course identifier.

    Returns:
        dict or None: Course details if found, otherwise None.
    """
    return courses.get(course_id)

```

Terminal

```
PS C:\Users\shyam\ai assistes code> python -m pydoc .\ass16ai.py
Help on module ass16ai:

NAME
    ass16ai

DESCRIPTION
    course.py
    A simple course management module that allows adding,
    removing, and retrieving course information.

FUNCTIONS
    add_course(course_id, name, credits)
        Add a new course to the course dictionary.

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

        Returns:
            bool: True if course is added successfully.

    get_course(course_id)
        Retrieve course details.

        Args:
            course_id (str): Unique course identifier.

        Returns:
            dict or None: Course details if found, otherwise None.

    remove_course(course_id)
        Remove a course from the course dictionary.

        Args:
            course_id (str): Unique course identifier.

        Returns:
            bool: True if course is removed, False if not found.
```

Html:

```
PS C:\Users\shyam\ai assistes code>
PS C:\Users\shyam\ai assistes code> python -m pydoc -w .\ass16ai.py
wrote ass16ai.html
PS C:\Users\shyam\ai assistes code>
```



index
ass16ai c:\users\shyam\ai assistes code\ass16ai.py

course.py
A simple course management module that allows adding,
removing, and retrieving course information.

Functions

add_course(course_id, name, credits)
Add a new course to the course dictionary.

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

Returns:
bool: True if course is added successfully.

get_course(course_id)
Retrieve course details.

Args:
course_id (str): Unique course identifier.

Returns:
dict or None: Course details if found, otherwise None.

remove_course(course_id)
Remove a course from the course dictionary.

Args:
course_id (str): Unique course identifier.

Returns:
bool: True if course is removed, False if not found.

Data

courses = {}

Web browser:

```
PS C:\Users\shyam\ai assistes code> python -m pydoc -p1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
server>
```

Python 3.13.2 [tags/v3.13.2:4f8bb39, MSC v.1942 64 bit (AMD64)]
Windows-11

Module Index : Topics : Keywords
Get Search

Index of Modules

Built-in Modules

_abc	_imp	_signal	binascii
_ast	_interpchannels	_sre	builtins
_bisect	_interpqueues	_stat	cmath
_blake2	_interpreters	_statistics	errno
_codecs	_io	_string	faulthandler
_codecs_cn	_json	_struct	gc
_codecs_hk	_locale	_symtable	itertools
_codecs_iso2022	_lsprof	_sysconfig	marshal
_codecs_jp	_md5	_thread	math
_codecs_kr	_multibytecodec	_tokenize	mmap
_codecs_tw	_opcode	_tracemalloc	msvcrt
_collections	_operator	_typing	nt
_contextvars	_pickle	_warnings	sys
_csv	_random	_weakref	time
_datetime	_sha1	_winapi	winreg
_functools	_sha2	array	xxsubtype
_heapq	_sha3	atexit	zlib

C:\Users\shyam\ai assistes code

ass11ai	ass2ai	ass7ai	
ass14ai	ass35ai	ass7ai.py	
ass16ai	ass3ai	ass8ai	assi73ai
ass17ai	ass4ai	ass9ai	file_processor
ass1ai	ass52	asshp3	lab1ai
ass21ai	ass5ai	assi12ai	sec
ass22ai	ass6ai	assi4ai	testcasedemo