ASSIGNMENT - 9.1

Name : Aravind Reddy

Hall Ticket No : 2303a51027

Batch No : 01

Course Title : AI Assisted Coding

Instructor's Name : Mr. S Naresh Kumar

Problem 1 :

Welcome | Assignment - 9.1.py

Assignments > Assignment - 9.1.py > ...

```python
# Task 1:

# (a) Docstring
def find_max(numbers):
    """
    Find the maximum number in a list of numbers.

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

    Returns:
    The maximum value from the list.
    """
    return max(numbers)

# (b) Inline comments
def find_max(numbers):
    # Use the built-in max function to find the maximum value in the list
    return max(numbers)

# (c) Google-style documentation
def find_max(numbers):
    """
    Find the maximum number in a list of numbers.

    Args:
        numbers (list): A list of numerical values.
    Returns:
        The maximum value from the list.
    """
    return max(numbers)

# output
numbers = [3, 1, 4, 1, 5, 9]
print(find_max(numbers))  # Output: 9
```

Output :

```
Output :  9
```

Problem 2:

```python
# Task 2:

# (a) Docstring
def login(user, password, credentials):
    """
    Authenticate a user based on provided credentials.

    Parameters:
    user (str): The username of the user trying to log in.
    password (str): The password provided by the user.
    credentials (dict): A dictionary containing valid username-password pairs.

    Returns:
    bool: True if authentication is successful, False otherwise.
    """
    return credentials.get(user) == password
# (b) Inline comments

def login(user, password, credentials):
    # Retrieve the stored password for the given user from the credentials dictionary
    stored_password = credentials.get(user)

    # Compare the provided password with the stored password and return the result
    return stored_password == password

# (c) Google-style documentation
def login(user, password, credentials):
    """
    Authenticate a user based on provided credentials.

    Args:
        user (str): The username of the user trying to log in.
        password (str): The password provided by the user.
        credentials (dict): A dictionary containing valid username-password pairs.

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

```python
# output
credentials = {'alice': 'password123', 'bob': 'securepass'}
print('Output : ', login('alice', 'password123', credentials))  # Output: True
print('Output : ', login('bob', 'wrongpass', credentials))    # Output: False
```

Output :

```
Output :  True
Output :  False
```

Problem 3:

```python
# calculator.py
def add(a, b):
    """
    Add two numbers.

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

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

def subtract(a, b):
    """
    Subtract one number from another.

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

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

```python
def multiply(a, b):
    """
    Multiply two numbers.

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

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

def divide(a, b):
    """
    Divide one number by another.

    Parameters:
    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

# To display the module documentation in the terminal, you can use the following code:
if __name__ == "__main__":
    import pydoc
    print(pydoc.render_doc(add))
    print(pydoc.render_doc(subtract))
    print(pydoc.render_doc(multiply))
    print(pydoc.render_doc(divide))
```

Terminal Output :

```
add(a, b)
    Add two numbers.

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

    Returns:
    float: The sum of a and b.

Python Library Documentation: function subtract in module __main__

subtract(a, b)
    Subtract one number from another.

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

    Returns:
    float: The difference of a and b.

Python Library Documentation: function multiply in module __main__

multiply(a, b)
    Multiply two numbers.

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

    Returns:
    float: The product of a and b.

Python Library Documentation: function divide in module __main__

divide(a, b)
    Divide one number by another.

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

    Returns:
    float: The quotient of a and b.

    Raises:
    ValueError: If b is zero.

(base) →  AI Assissted Coding
```

Web site Output :

127.0.0.1:5500/Assignments/calculator.html

index
calculator /Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/calculator.py

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.

Functions

add(a, b)
Add two numbers.

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

Returns:
float: The sum of a and b.

divide(a, b)
Divide one number by another.

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

Returns:
float: The quotient of a and b.

Raises:
ValueError: If b is zero.

multiply(a, b)
Multiply two numbers.

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

Returns:
float: The product of a and b

Problem 4:

```python
# conversion.py
def decimal_to_binary(n):
    """
    Convert a decimal number to its binary representation.

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

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

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

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

    Returns:
    int: The decimal representation of the binary number.
    """
    return int(b, 2)
```

```python
def decimal_to_hexadecimal(n):
    """
    Convert a decimal number to its hexadecimal representation.

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

    Returns:
    str: The hexadecimal representation of the decimal number.
    """
    return hex(n)[2:].upper()

# To display the module documentation in the terminal, you can use the following code:
if __name__ == "__main__":
    import pydoc
    print(pydoc.render_doc(decimal_to_binary))
    print(pydoc.render_doc(binary_to_decimal))
    print(pydoc.render_doc(decimal_to_hexadecimal))

# To generate and export the module documentation in HTML format, you can run the following command in the terminal:
# pydoc -w conversion
# This will create a file named conversion.html, which you can open in a web browser to verify the output.
```

Terminal Output :

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

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

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

Python Library Documentation: function binary_to_decimal in module __main__

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

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

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

Python Library Documentation: function decimal_to_hexadecimal in module __main__

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

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

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

Web Site Output :

index

conversion /Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/conversion.py

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.

Functions

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

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

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

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

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

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

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

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

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

Problem 5:

```python
# course.py
class Course:
    """
    A class to represent a course in a course management system.
    """

    def __init__(self):
        self.courses = {}

    def add_course(self, course_id, name, credits):
        """
        Add a course to the course management 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:
        None
        """
        self.courses[course_id] = {'name': name, 'credits': credits}

    def remove_course(self, course_id):
        """
        Remove a course from the course management system.

        Parameters:
        course_id (str): The unique identifier for the course to be removed.

        Returns:
        None
        """
        if course_id in self.courses:
            del self.courses[course_id]
```

```
    def get_course(self, course_id):
        """
        Retrieve information about a specific course.

        Parameters:
        course_id (str): The unique identifier for the course to retrieve.

        Returns:
        dict: A dictionary containing the name and credits of the course, or None if the course does not exist.
        """
        return self.courses.get(course_id)

# To display the module documentation in the terminal, you can use the following code:
if __name__ == "__main__":
    import pydoc
    course_manager = Course()
    print(pydoc.render_doc(course_manager.add_course))
    print(pydoc.render_doc(course_manager.remove_course))
    print(pydoc.render_doc(course_manager.get_course))
# To generate and export the module documentation in HTML format, you can run the following command in the terminal:
# pydoc -w course
# This will create a file named course.html, which you can open in a web browser to
# verify the output.
```

Terminal Output:

```
add_course(course_id, name, credits) method of __main__.Course instance
    Add a course to the course management 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:
    None

Python Library Documentation: method remove_course in module __main__

remove_course(course_id) method of __main__.Course instance
    Remove a course from the course management system.

    Parameters:
    course_id (str): The unique identifier for the course to be removed.

    Returns:
    None

Python Library Documentation: method get_course in module __main__

get_course(course_id) method of __main__.Course instance
    Retrieve information about a specific course.

    Parameters:
    course_id (str): The unique identifier for the course to retrieve.

    Returns:
    dict: A dictionary containing the name and credits of the course, or None if the course does not exist.
```

Web site Output:

index
course /Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/course.py

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.

Classes

builtins.object
Course

class **Course**(builtins.object)
A class to represent a course in a course management system.

Methods defined here:

__init__(self)
Initialize self. See help(type(self)) for accurate signature.

add_course(self, course_id, name, credits)
Add a course to the course management 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:
None

get_course(self, course_id)
Retrieve information about a specific course.

Parameters:
course_id (str): The unique identifier for the course to retrieve.

Returns:
dict: A dictionary containing the name and credits of the course, or None if the course does not exist.

remove_course(self, course_id)
Remove a course from the course management system.

Parameters:
course_id (str): The unique identifier for the course to be removed.

Returns:
None

Data descriptors defined here:

__dict__
dictionary for instance variables

__weakref__
list of weak references to the object