

Assignment – 9.1

Documentation Generation – Automatic Documentation and Code Comments

Week: 5 (Monday)

Name: K.MANOJ KUMAR

HT NO: 2303A52201

Batch: 35

Problem 1

Given Python Function

```
def find_max(numbers):  
    return max(numbers) (a)
```

Docstring Documentation

```
def find_max(numbers):  
    """  
    Returns the maximum value from a list of numbers.  
  
    Parameters:  
    numbers (list): A list of numeric values.  
  
    Returns:  
    int or float: The maximum value in the list.  
    """  
    return max(numbers) (b)
```

Inline Comments

```
def find_max(numbers):  
    # Find and return the maximum value from the list  
    return max(numbers) (c) Google-Style Documentation
```

```
def find_max(numbers):  
    """  
    Finds the maximum value in a list of numbers.
```

```
    Args:
```

```

    numbers (list): A list containing numeric values.

Returns:
    int or float: The largest number in the list.
    """
    return
max(numbers)

```

Critical Comparison

- **Docstrings** provide structured internal documentation and are accessible using `help()` and `pydoc`.
- **Inline comments** are simple but limited and unsuitable for detailed explanations.
- **Google-style documentation** is highly readable, standardized, and ideal for large projects.

Recommendation

For a mathematical utilities library, **Google-style documentation** is most effective due to its clarity, consistency, and compatibility with documentation tools.

Problem 2

Given Python Function

```

def login(user, password, credentials):
    return credentials.get(user) == password

```

(a) Docstring Documentation

```

def login(user, password, credentials):
    """
    Verifies user login credentials.

    Parameters:
        user (str): Username
        password (str): User password
        credentials (dict): Dictionary of stored credentials

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

```

(b) Inline Comments

```

def login(user, password, credentials):
    # Check if the entered password matches stored credentials
    return credentials.get(user) == password

```

(c) Google-Style Documentation

```
def login(user, password, credentials):  
    """  
        Authenticates a user using provided credentials.  
    Args:  
        user (str): Username of the user.        password (str):  
        Password entered by the user.        credentials (dict):  
        Dictionary mapping users to passwords.  
  
    Returns:  
        bool: True if authentication succeeds, otherwise False.  
    """  
    return credentials.get(user) ==  
password
```

Comparison and Recommendation

Google-style documentation is most helpful for **new developers onboarding a project** because it clearly explains parameters, return values, and intent in a standardized format.

Problem 3 – Calculator Module

calculator.py

```
def add(a, b):  
    """Returns the sum of two numbers."""  
    return a + b  
    def subtract(a,  
b):  
    """Returns the difference of two numbers."""  
    return a - b  
    def multiply(a,  
b):  
    """Returns the product of two numbers."""  
    return a * b  
    def divide(a,  
b):  
    """Returns the quotient of two numbers."""  
    return a / b
```

Documentation Generation

□ Terminal documentation: `help(calculator)` □

HTML documentation generation: `pydoc -w
calculator`

The generated calculator.html file is opened in a web browser to verify documentation.

Problem 4 – Conversion Utilities Module

conversion.py

```
def decimal_to_binary(n):
    """Converts a decimal number to binary."""
    return bin(n)[2:]
    def
binary_to_decimal(b):
    """Converts a binary number to decimal."""
    return int(b, 2)
    def
decimal_to_hexadecimal(n):
    """Converts a decimal number to hexadecimal."""

return hex(n)[2:]
```

Documentation Generation

- Terminal: `help(conversion)` □ HTML export using: `pydoc -w conversion`

Problem 5 – Course Management Module

course.py

```
courses = {}
def add_course(course_id, name,
credits):
    """Adds a new course to the course list."""
    courses[course_id] = {'name': name, 'credits': credits}
    def
remove_course(course_id):
    """Removes a course using course ID."""
    courses.pop(course_id,
None)
    def
get_course(course_id):
    """Retrieves course details by course ID."""
return courses.get(course_id)
```

Documentation Generation

- Terminal documentation using `help(course)`
- HTML documentation exported using:

`pydoc -w course`

The generated HTML file is opened in a browser to verify correctness.

Conclusion

This lab demonstrates the importance of proper documentation in software development. Automatic documentation generation improves maintainability, onboarding efficiency, and overall code quality. Google-style docstrings are recommended for professional and collaborative projects.