

# Lab Experiment: Documentation Generation-Automatic documentation and code comments

**2303a51834**

**Batch :- 27**

**Assignment-9.1**

## **Problem 1**

Given Function

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

### **(a)Docstring Style**

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

### **(b) Inline Comments**

```
def find_max(numbers):  
    # This function returns the largest number  
    # from the given list of numbers  
    return max(numbers)
```

### **(c) Google-Style Documentation**

```
def find_max(numbers):  
    """  
  
    Finds the maximum number in a list.  
  
    Args:  
        numbers (list): List of numeric values.  
  
    Returns:  
        int or float: Maximum value in the list.  
    """  
    return max(numbers)
```

### **Critical Comparison**

<b>Style</b>	<b>Advantages</b>	<b>Disadvantages</b>	<b>Use Case</b>
<b>Docstring</b>	<b>Standard Python documentation</b>	<b>Slightly lengthy</b>	<b>General Python projects</b>
<b>Inline Comments</b>	<b>Easy to understand quickly</b>	<b>Not included in documentation tools</b>	<b>Small scripts</b>
<b>Google Style</b>	<b>Structured &amp; professional</b>	<b>Requires formatting knowledge</b>	<b>Large team projects</b>

## **Recommended Style (Mathematical Utility Library)**

Google-Style Documentation

Easy to read

Compatible with documentation tools

Standard in professional development

## **Problem 2**

Given Function

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

### **(a) Docstring Style**

```
def login(user, password, credentials):
```

```
    """
```

```
    Validates user login credentials.
```

```
    Parameters:
```

```
        user (str): Username
```

```
        password (str): Password entered by user
```

```
        credentials (dict): Stored username-password pairs
```

```
    Returns:
```

```
        bool: True if login successful, otherwise False
```

```
    """
```

```
    return credentials.get(user) == password
```

### **(b) Inline Comments**

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

### **(c) Google Style Documentation**

```
def login(user, password, credentials):
    """
    Authenticates a user.

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

    Returns:
        bool: Authentication result
    """
    return credentials.get(user) == password
```

### **Comparison**

<b>Style</b>	<b>Strength</b>
<b>Inline</b>	<b>Quick understanding</b>
<b>Docstring</b>	<b>Standard &amp; simple</b>
<b>Google Style Best readability &amp; structure</b>	

## **Recommended Style (For New Developers)**

### **Google Style**

**Very clear structure**

**Easy onboarding**

**Professional readability**

Problem 3 – Calculator Module

calculator.py

Calculator Module

Provides basic arithmetic operations.

```
def add(a, b):
```

```
    Returns sum of two numbers.
```

```
    return a + b
```

```
def subtract(a, b):
```

```
    Returns difference of two numbers.
```

```
    return a - b
```

```
def multiply(a, b):
```

```
    Returns product of two numbers.
```

```
    return a * b
```

```
def divide(a, b):  
    Returns quotient of two numbers.  
    if b == 0:  
        raise ValueError("Cannot divide by zero")  
    return a / b
```

### **Display Documentation in Terminal**

```
python -m pydoc calculator
```

### **Generate HTML Documentation**

```
python -m pydoc -w calculator
```

**This creates:**

calculator.html

## **Problem 4 – Conversion Utilities Module**

**conversion.py**

### **Conversion Utility Module**

**Provides number conversion functions.**

```
def decimal_to_binary(n):  
    Converts decimal number to binary.  
    return bin(n)[2:]
```

```
def binary_to_decimal(b):
```

Converts binary number to decimal.

```
return int(b, 2)
```

```
def decimal_to_hexadecimal(n):
```

Converts decimal number to hexadecimal.

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

### **Terminal Documentation**

```
python -m pydoc conversion
```

### **Generate HTML**

```
python -m pydoc -w conversion
```

### **Problem 5 – Course Management Module**

**course.py**

"

**Course Management Module**

**Handles course operations.**

"

```
courses = {}
```

```
def add_course(course_id, name, credits):
```

Adds a course to the course list.

```
courses[course_id] = {"name": name, "credits": credits}
```

```
def remove_course(course_id):  
    Removes a course from the list.  
    courses.pop(course_id, None)
```

```
def get_course(course_id):  
    Returns course details.  
    return courses.get(course_id)
```

### **Terminal Documentation**

```
python -m pydoc course
```

### **Generate HTML**

```
python -m pydoc -w course
```