

2303A52416

A.akshaya

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.

1

Code

The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main workspace has tabs for 'main.py' and 'calculator.py'. The code in 'main.py' imports 'calculator' and uses its functions to perform arithmetic operations. The code in 'calculator.py' defines four arithmetic functions: add, subtract, multiply, and divide. The 'calculator.py' code includes docstrings describing the purpose of each function. The bottom right corner shows a terminal window with the message "...Program finished with exit code 0 Press ENTER to exit console.".

```
1 """
2 calculator.py
3 |
4 A simple calculator module that provides basic arithmetic operations.
5
6 Functions:
7     add(a, b)
8     subtract(a, b)
9     multiply(a, b)
10    divide(a, b)
11 """
12
13
14 def add(a, b):
15     """Return the sum of two numbers."""
16     return a + b
17
18
19 def subtract(a, b):
20     """Return the difference of two numbers."""
21     return a - b
22
23
24 def multiply(a, b):
25     """Return the product of two numbers."""
26     return a * b
27
28 def divide(a, b):
29     """Return the quotient of two numbers."""
30     return a / b
```

The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with links for IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main area has tabs for Run, Debug, Stop, Share, Save, Beautify, Language (Python 3), and settings. A file named 'main.py' is open, containing the following Python code:

```
10     return a + b
11
12
13
14 def subtract(a, b):
15     """Return the difference of two numbers."""
16     return a - b
17
18
19
20 def multiply(a, b):
21     """Return the product of two numbers."""
22     return a * b
23
24
25 def divide(a, b):
26     """
27     Return the quotient of two numbers.
28
29     Raises:
30         ZeroDivisionError: If denominator is zero.
31     """
32     if b == 0:
33         raise ZeroDivisionError("Cannot divide by zero!")
34     return a / b
35
36
37
38
39
```

The output window below shows the program's execution:

```
...Program finished with exit code 0
Press ENTER to exit console.
```

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.

The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with links for IDE, My Projects, Classroom, Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main area has tabs for Run, Debug, Stop, Share, Save, and Beautify, with Language set to Python 3. The current file is 'main.py' containing the following code:

```
1 def login(user, password, credentials):
2     """
3         Checks whether the given username and password are valid.
4
5         This function verifies if the provided password matches
6         the stored password for the given user in the credentials dictionary.
7
8         Parameters:
9             user (str): The username entered by the user.
10            password (str): The password entered by the user.
11            credentials (dict): A dictionary containing usernames as keys
12                                and passwords as values.
13
14        Returns:
15            bool: True if login is successful, False otherwise.
16        """
17    return credentials.get(user) == password
```

Below the code editor is a terminal window showing the output: "...Program finished with exit code 0 Press ENTER to exit console." The bottom of the page includes a footer with links for About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, Privacy, and a copyright notice for 2016-2026 GDB Online.

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.

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with links for IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main area has tabs for Run, Debug, Stop, Share, Save, and Beautify, with Language set to Python 3. The code editor contains a file named 'main.py' with the following content:

```
1 """
2 calculator.py
3
4 A simple Calculator Module.
5
6 This module provides basic arithmetic operations such as:
7
8     - Addition
9     - Subtraction
10    - Multiplication
11    - Division
12
13 Each function includes proper docstrings so that documentation
14 can be generated automatically using Python tools like pydoc.
15 """
16
17
18 def add(a, b):
19     """
20         Add two numbers.
21
22     Args:
23         a (int/float): First number
24         b (int/float): Second number
25
26     Returns:
27         int/float: Sum of a and b
28
29     Examples:
30         add(1, 2) == 3
31         add(2.5, 4.5) == 7.0
32
33     """
34     return a + b
```

The output console at the bottom shows the program finished with exit code 0.

The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: OnlineGDB (with a logo), online compiler and debugger for c/c++, code, compile, run, debug, share, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main workspace has tabs for main.py and input. The code in main.py defines three functions: add, subtract, and multiply, each with detailed docstrings specifying arguments and return types. The output console at the bottom shows the program finished with exit code 0 and prompts the user to press ENTER to exit.

```
main.py
24     b (int/float): Second number
25
26     Returns:
27     int/float: Sum of a and b
28     """
29     return a + b
30
31
32     def subtract(a, b):
33     """
34     Subtract two numbers.
35
36     Args:
37         a (int/float): First number
38         b (int/float): Second number
39
40     Returns:
41         int/float: Difference of a and b
42     """
43     return a - b
44
45
46     def multiply(a, b):
47     """
...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows the OnlineGDB IDE interface. The top bar includes icons for file operations (New, Open, Save, etc.), run/debug/stop, share, beautify, and language selection (Python 3). The left sidebar has links for IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main workspace shows a Python script named 'main.py' with code for multiplication and division. The bottom console output shows the program finished with exit code 0.

```
main.py
47     """
48     Multiply two numbers.
49
50     Args:
51         a (int/float): First number
52         b (int/float): Second number
53
54     Returns:
55         int/float: Product of a and b
56     """
57     return a * b
58
59
60 def divide(a, b):
61     """
62     Divide two numbers.
63
64     Args:
65         a (int/float): Numerator
66         b (int/float): Denominator
67
68     Returns:
69         float: Quotient of a and b
70
71
72
73
74
75
76
77
78
79
79
```

...Program finished with exit code 0
Press ENTER to exit console.

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. The main area has tabs for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and language selection ('Python 3'). The code editor window contains a Python script named 'main.py' with the following content:

```
54     Returns:
55     int/float: Product of a and b
56     """
57     return a * b
58
59
60 def divide(a, b):
61     """
62     Divide two numbers.
63
64     Args:
65         a (int/float): Numerator
66         b (int/float): Denominator
67
68     Returns:
69         float: Quotient of a and b
70
71     Raises:
72         ZeroDivisionError: If b is zero
73     """
74     if b == 0:
75         raise ZeroDivisionError("Division by zero is not allowed!")
76     return a / b
77
```

The terminal window below shows the execution results:

```
...Program finished with exit code 0
Press ENTER to exit console.
```

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 browser.

OnlineGDB
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Opportunity for Intern

Sign Up

Login

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

```
main.py
1 """
2 conversion.py
3
4 A Conversion Utilities Module.
5
6 This module provides functions to convert numbers between
7 different number systems:
8
9     - Decimal to Binary
10    - Binary to Decimal
11    - Decimal to Hexadecimal
12
13 Docstrings are included so that documentation can be
14 generated automatically using pydoc.
15 """
16
17
18 def decimal_to_binary(n):
19     """
20         Convert a decimal integer to its binary representation.
21
22     Args:
23         n (int): A decimal number
24
25     Returns:
26         str: Binary string equivalent of the decimal number
27
28     Example:
29         decimal_to_binary(10) -> '1010'
30     """
31     return bin(n)[2:]
32
33
34 def binary_to_decimal(b):
35     """
36         Convert a binary string to its decimal representation.
37
38     Args:
39         b (str): A binary number as a string
40
41     Returns:
42         int: Decimal equivalent of the binary number
43
44     Example:
45         binary_to_decimal("1010") -> 10
46     """
47     return int(b, 2)

```

input

```
...Program finished with exit code 0
Press ENTER to exit console.
```

OnlineGDB
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Opportunity for Intern

Sign Up

Login

About • FAQ • Blog • Terms of Use •
Contact Us • GDB Tutorial • Credits •
Privacy
© 2016 - 2026 GDB Online

```
main.py
25
26     Returns:
27         str: Binary string equivalent of the decimal number
28
29     Example:
30         decimal_to_binary(10) -> '1010'
31     """
32     return bin(n)[2:]
33
34 def binary_to_decimal(b):
35     """
36         Convert a binary string to its decimal representation.
37
38     Args:
39         b (str): A binary number as a string
40
41     Returns:
42         int: Decimal equivalent of the binary number
43
44     Example:
45         binary_to_decimal("1010") -> 10
46     """
47     return int(b, 2)

```

input

```
...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows the OnlineGDB IDE interface. On the left, a sidebar menu includes links for IDE, My Projects, Classroom, Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main workspace displays a Python script named 'main.py' with the following content:

```
    Returns:
        int: Decimal equivalent of the binary number

    Example:
        binary_to_decimal("1010") -> 10
    """
    return int(b, 2)

def decimal_to_hexadecimal(n):
    """
    Convert a decimal integer to its hexadecimal representation.

    Args:
        n (int): A decimal number

    Returns:
        str: Hexadecimal string equivalent of the decimal number

    Example:
        decimal_to_hexadecimal(255) -> 'FF'
    """
    return hex(n)[2:].upper()
```

The terminal window at the bottom shows the output of running the program:

```
...Program finished with exit code 0
Press ENTER to exit console.
```

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.

The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main workspace has tabs for 'main.py' and 'input'. The code in 'main.py' is as follows:

```
1 """
2 course.py
3
4 Course Management Module.
5
6 This module provides basic functions to manage courses
7 in a simple course database.
8
9 Functions included:
10
11     - add_course(course_id, name, credits)
12     - remove_course(course_id)
13     - get_course(course_id)
14
15 Docstrings are added so that documentation can be
16 automatically generated using Python tools like pydoc.
17 """
18
19 # Dictionary to store course details
20 courses = {}
21
22
23 def add_course(course_id, name, credits):
24     """
...Program finished with exit code 0
Press ENTER to exit console.
```

The status bar at the bottom indicates 'Language Python 3' and includes icons for file operations, run, stop, share, save, and beautify.

The screenshot shows the OnlineGDB IDE interface. The top navigation bar includes icons for file operations (New, Open, Save, etc.) and language selection (Language: Python 3). Below the navigation bar is a toolbar with buttons for Run, Stop, Share, Save, and Beautify.

The left sidebar contains links for the IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login.

The main workspace displays Python code in a file named `main.py`. The code defines two functions: `add_course` and `remove_course`.

```
22
23 - def add_course(course_id, name, credits):
24     """
25         Add a new course into the course database.
26
27 -     Args:
28         course_id (str): Unique identifier for the course
29         name (str): Name of the course
30         credits (int): Number of credits assigned to the course
31
32 -     Returns:
33         str: Confirmation message after adding the course
34     """
35 -     courses[course_id] = {
36         "name": name,
37         "credits": credits
38     }
39     return f"Course {course_id} added successfully!"
40
41
42 - def remove_course(course_id):
43     """
44         Remove an existing course from the course database.
45     """
...Program finished with exit code 0
Press ENTER to exit console.
```

The bottom status bar indicates the program finished with exit code 0 and prompts the user to press Enter to exit the console.

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. The main area has tabs for Run, Debug, Stop, Share, Save, Beautify, Language (Python 3), and settings. The code editor window is titled 'main.py' and contains the following Python code:

```
46 -     Args:
47 -         course_id (str): Unique identifier of the course to remove
48 -
49 -     Returns:
50 -         str: Confirmation message if removed, otherwise error message
51 -         """
52 -     if course_id in courses:
53 -         del courses[course_id]
54 -         return f"Course {course_id} removed successfully!"
55 -     return "Course not found!"
56 -
57 - def get_course(course_id):
58 -     """
59 -     Retrieve details of a specific course.
60 -
61 -     Args:
62 -         course_id (str): Unique identifier of the course
63 -
64 -     Returns:
65 -         dict: Course information if found
66 -         str: Error message if course does not exist
67 -         """
68 -
```

The output console shows the program's execution results:

```
...Program finished with exit code 0
Press ENTER to exit console.
```