



School of Computer Science and Artificial Intelligence

### Lab Assignment-9.1

Course Name : AI Assistant Coding  
Name of the Student : Shaik Naved Ahmed  
Registration No : 2303A54053  
Batch No : 47-B

#### 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.

#### SCREENSHOT OF GENERATED CODE:

The screenshot shows a code editor interface with the following details:

- EXPLORER View:** Shows a file tree with various files and folders related to the assignment, including `AIAC`, `lab.9.1.py`, and several PDF files.
- LAB.9.1.PY Content:**

```
#Docstring Documentation with complete code  
def find_max(numbers):  
    """  
        This function takes a list of numbers as input and returns the maximum value from the list.  
    Parameters:  
        numbers (list): A list of numerical values.  
    Returns:  
        The maximum value from the input list.  
    """  
    # Check if the input list is empty  
    if not numbers:  
        raise ValueError("The input list cannot be empty.")  
    # Use the built-in max function to find the maximum value in the list  
    return max(numbers)  
#Example usage:  
numbers = [3, 1, 4, 1, 5, 9]  
max_value = find_max(numbers)  
print(f"The maximum value in the list is: {max_value}")
```

**Inline Documentation with complete code**

```
# Inline Documentation with complete code  
def find_max(numbers):  
    """  
        This function takes a list of numbers as input and returns the maximum value from the list.  
    Parameters:  
        numbers (list): A list of numerical values.  
    Returns:  
        The maximum value from the input list.  
    """  
    # Check if the input list is empty  
    if not numbers:  
        raise ValueError("The input list cannot be empty.")  
    # Use the built-in max function to find the maximum value in the list  
    return max(numbers)  
#Example usage:  
numbers = [3, 1, 4, 1, 5, 9]  
max_value = find_max(numbers)  
print(f"The maximum value in the list is: {max_value}")
```

**Google-style Documentation with complete code**

```
#Google-style Documentation with complete code  
def find_max(numbers):  
    """  
        Finds the maximum value in a list of numbers.  
    Args:  
        numbers (list): A list of numerical values. The list should not be empty.  
    Returns:  
        The maximum value from the input list.  
    Raises:  
        ValueError: If the input list is empty.  
    """  
    if not numbers:  
        raise ValueError("The input list cannot be empty.")  
    return max(numbers)  
#Example usage:  
numbers = [3, 1, 4, 1, 5, 9]  
max_value = find_max(numbers)  
print(f"The maximum value in the list is: {max_value}")
```
- TERMINAL View:** Shows the command-line output of running the script, indicating the use of Python 3.8 and the execution of the `lab.9.1.py` file.

## EXPLANATION:

The function `find\_max` takes a list of numbers as input and returns the maximum value from that list. It first checks if the input list is empty and raises a `ValueError` if it is. If the list is not empty, it uses the built-in `max` function to find and return the maximum value. The function is documented using three different styles: docstring documentation, inline documentation, and Google-style documentation, providing clear explanations of the function's purpose, parameters, return value, and potential exceptions.

Style	Advantages	Disadvantages	Suitable Use Cases
<b>Docstring (standard)</b>	Simple, readable, built into Python help tools	Less structured than formal styles	Small scripts, educational code
<b>Inline comments</b>	Explain logic step-by-step	Can clutter code, not accessible via help()	Complex algorithms needing explanation
<b>Google-style</b>	Highly structured, professional, IDE-friendly, good for teams	Slightly longer to write	Libraries, APIs, production code

## 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.

## SCREENSHOT OF GENERATED CODE:

```
lab.9.py
1 #!/usr/bin/python
2
3 def login(user, password, credentials):
4     return credentials.get(user) == password
5
6
7 #Docstring Documentation with complete code
8 def login(user, password, credentials):
9     """
10     This function checks if the provided username and password match the credentials stored in a dictionary.
11
12     Parameters:
13         user (str): The username to be checked.
14         password (str): The password to be checked.
15         credentials (dict): A dictionary containing username-password pairs.
16
17     Returns:
18         bool: True if the username and password match the credentials, False otherwise.
19
20     """
21     return credentials.get(user) == password
22
23
24 #Docstring Documentation with complete code
25 def login(user, password, credentials):
26     """
27     This function checks if the provided username and password match the credentials stored in a dictionary.
28
29     Parameters:
30         user (str): The username to be checked.
31         password (str): The password to be checked.
32         credentials (dict): A dictionary containing username-password pairs.
33
34     Returns:
35         bool: True if the username and password match the credentials, False otherwise.
36
37     """
38     return credentials.get(user) == password
39
40
41 #Google-Style Documentation with complete code
42 def login(user, password, credentials):
43     """
44     Checks if the provided username and password match the credentials stored in a dictionary.
45
46     Args:
47         user (str): The username to be checked.
48         password (str): The password to be checked.
49         credentials (dict): A dictionary containing username-password pairs. The keys are usernames and the values are passwords.
50
51     Returns:
52         bool: True if the username and password match the credentials, False otherwise.
53
54     """
55     return credentials.get(user) == password
56
57
58 #lab.9.1.py
59
60 #lab.9.2.py
61
62 #lab.9.3.py
63
64 #lab.9.4.py
65
66 #lab.9.5.py
67
68 #lab.9.6.py
69
70 #lab.9.7.py
71
72 #lab.9.8.py
73
74 #lab.9.9.py
75
76 #users.secure.txt
77
78 #users.txt
```

## EXPLANATION:

The function `login` takes three parameters: `user`, `password`, and `credentials`. It checks if the provided username and password match the credentials stored in a dictionary. The function uses the `get` method of the dictionary to retrieve the password associated with the given username and compares it with the provided password. If they match, it returns `True`; otherwise, it returns `False`. The function is documented using three different styles: docstring documentation, inline documentation, and Google-style documentation.

documentation, inline documentation, and Google-style documentation, providing clear explanations of the function's purpose, parameters, return value, and potential exceptions.

Style	Advantages	Disadvantages	Best Use Cases
<b>Docstring (standard)</b>	Simple, readable, built into Python help tools	Less structured for large projects	Small scripts, student projects
<b>Inline comments</b>	Explain logic directly near code	Not visible via help(), can clutter code	Complex internal logic explanation
<b>Google-style</b>	Structured, professional, IDE-friendly, great for teams	Slightly longer to write	Libraries, APIs, collaborative projects

### 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.

### SCREENSHOT OF GENERATED CODE:

 Below the code, the terminal shows the output of 'pydoc calculator.py' and the generated HTML documentation in the browser."/>

```

  1 # Lab 9.1
  2 # Problem-1
  3 """
  4     This module provides basic arithmetic operations.
  5 """
  6
  7 def add(a, b):
  8     """Return the sum of a and b."""
  9     return a + b
 10
 11 def subtract(a, b):
 12     """Return the difference of a and b."""
 13     return a - b
 14
 15 def multiply(a, b):
 16     """Return the product of a and b."""
 17     return a * b
 18
 19 def divide(a, b):
 20     """Return the quotient of a and b.
 21
 22     Raises:
 23         ValueError: If b is zero.
 24     """
 25     if b == 0:
 26         raise ValueError("Cannot divide by zero")
 27     return a / b
  
```

Help on module calculator:  
 NAME  
 calculate - calculate.py - Simple calculator module with basic operations.  
 FUNCTIONS  
 add(a, b)  
     Return the sum of a and b.  
     Raises:  
         ValueError: If b is zero.  
 multiply(a, b)  
     Return the product of a and b.  
 subtract(a, b)  
     Return the difference of a and b.

[index](#)  
**calculator** [/Users/navedahmedshaik/Documents/3-2/AIAC/calculator.py](#)

calculator.py - Simple calculator module with basic operations.

## Functions

### **add(a, b)**

Return the sum of a and b.

### **divide(a, b)**

Return the quotient of a and b.

Raises:

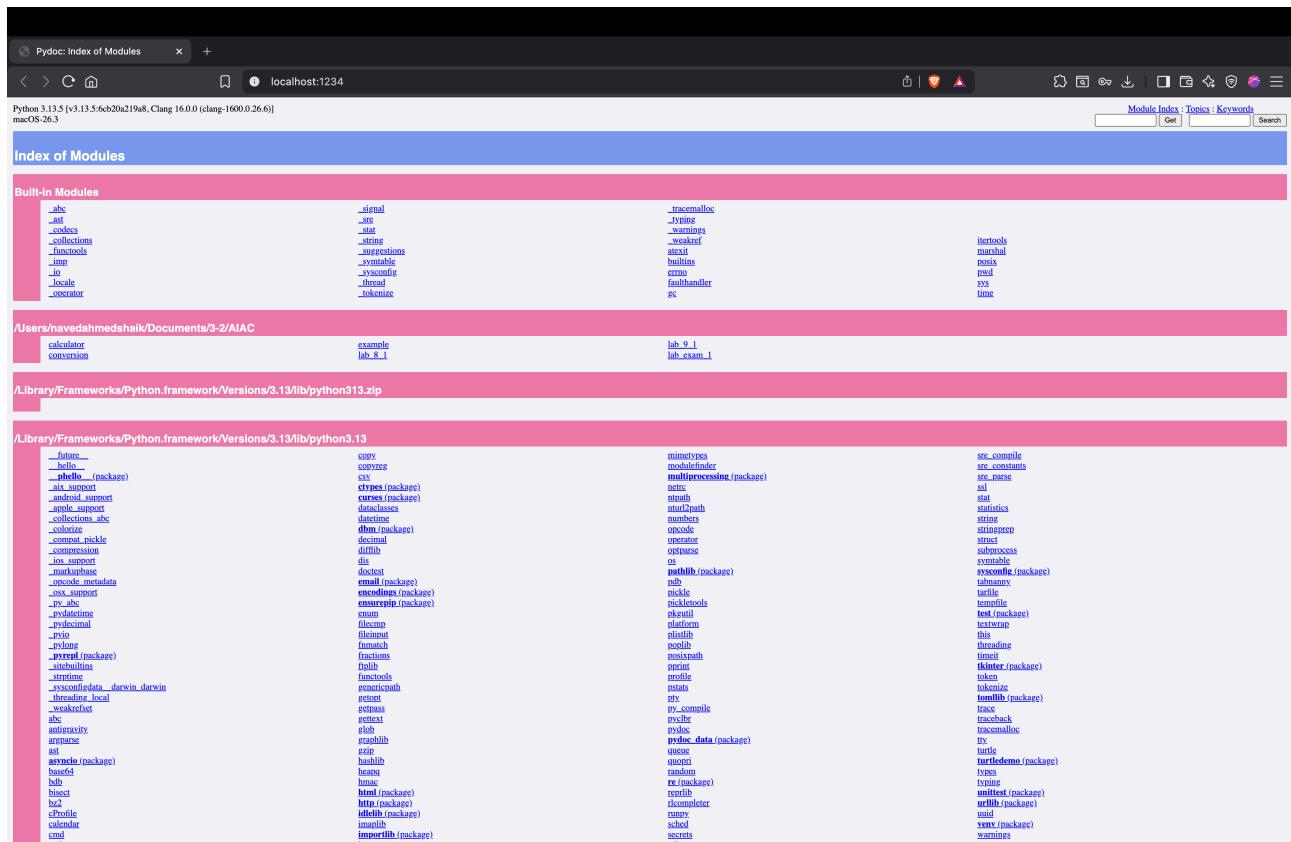
ValueError: If b is zero.

### **multiply(a, b)**

Return the product of a and b.

### **subtract(a, b)**

Return the difference of a and b.



## 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.

## SCREENSHOT OF GENERATED CODE:

```

conversion.py > ...
1  # lab_9_1
2  # Problem 4
3  """
4      conversion.py
5
6      Utility functions for number system conversions:
7
8          - Decimal to Binary
9          - Binary to Decimal
10         - Decimal to Hexadecimal
11
12     def decimal_to_binary(n):
13         """Convert a decimal integer to its binary string representation."""
14         if n < 0:
15             raise ValueError("Only non-negative integers allowed")
16         return bin(n)[2:]
17
18     def binary_to_decimal(b):
19         """Convert a binary string to its decimal integer representation."""
20         if not all(c in '01' for c in b):
21             raise ValueError("Input must be a binary string")
22         return int(b, 2)
23
24
25     def decimal_to_hexadecimal(n):
26         """Convert a decimal integer to its hexadecimal string representation."""
27         if n < 0:
28             raise ValueError("Only non-negative integers allowed")
29         return hex(n)[2:].upper()
30
31
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Help on module conversion:
NAME
    conversion -- conversion.py
DESCRIPTION
    Utility functions for number system conversions:
    - Decimal to Binary
    - Binary to Decimal
    - Decimal to Hexadecimal
FUNCTIONS
    decimal_to_binary(b)
        Convert a decimal string to its binary integer representation.

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

    decimal_to_hexadecimal(n)
        Convert a decimal integer to its hexadecimal string representation.
FILE
    /Users/navedahmedshaih/Documents/3-2/AIAC/conversion.py
Help on conversion line 1/24 (END) (press h for help or q to quit)

```

Built-in Modules			
<code>_abc</code>	<code>signal</code>	<code>tracemalloc</code>	
<code>_ast</code>	<code>_secrets</code>	<code>_typing</code>	
<code>_codecs</code>	<code>_star</code>	<code>_warnings</code>	
<code>_collections</code>	<code>_string</code>	<code>_weakref</code>	
<code>_functools</code>	<code>_suggestions</code>	<code>atexit</code>	
<code>_imp</code>	<code>_symboltable</code>	<code>builtins</code>	
<code>_io</code>	<code>_sysconfig</code>	<code>errno</code>	
<code>_locale</code>	<code>_thread</code>	<code>faulthandler</code>	
<code>_operator</code>	<code>_tokenize</code>	<code>gc</code>	

/Users/navedahmedshaih/Documents/3-2/AIAC			
<code>calculator</code>	<code>example</code>	<code>lab_9_1</code>	
<code>conversion</code>	<code>lab_8_1</code>	<code>lab_exam_1</code>	

/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13			
<code>_functools</code>	<code>copy</code>	<code>mimetypes</code>	<code>sys._compile</code>
<code>_hashlib</code>	<code>copyreg</code>	<code>multibytecodec</code>	<code>sys._constants</code>
<code>_phello</code> (package)	<code>csv</code>	<code>multiprocessing</code> (package)	<code>sys._name</code>
<code>_aix_support</code>	<code>ctypes</code> (package)	<code>netrc</code>	<code>sys._psize</code>
<code>_android_support</code>	<code>curses</code> (package)	<code>ntpath</code>	<code>sys._stack_info</code>
<code>_argparse</code>	<code>database</code>	<code>numbers</code>	<code>sys._staged幕</code>
<code>_collections_abc</code>	<code>datetime</code>	<code>opcode</code>	<code>sys._strptime</code>
<code>_colorizer</code>	<code>dbm</code> (package)	<code>operator</code>	<code>sys._structseq</code>
<code>_compat_pickle</code>	<code>decimal</code>	<code>os</code>	<code>sys._structseq_map</code>
<code>_compression</code>	<code>difflib</code>	<code>pathlib</code> (package)	<code>sys._syncfile</code>
<code>_cosmopolitan</code>	<code>dis</code>	<code>pdb</code>	<code>tabnanny</code>
<code>_csv</code>	<code>doctest</code>	<code>pickle</code>	<code>telnetlib</code>
<code>_ctypes</code>	<code>email</code> (module)	<code>marshal</code>	<code>text</code>
<code>_ctypes_test</code>	<code>encodings</code> (module)	<code>nickle</code>	<code>textwrap</code>
<code>_collections</code>	<code>ensurepip</code> (package)	<code>nickleools</code>	<code>time</code>
<code>_abc</code>	<code>enum</code>	<code>olefile</code>	<code>threading</code>
<code>_colorizer</code>	<code>filecmp</code>	<code>platform</code>	<code>timeit</code>
<code>_compat_pickle</code>	<code>fido</code>	<code>stat</code>	<code>tcl</code>
<code>_ctypes</code>	<code>format</code>	<code>statvfs</code>	<code>tkinter</code> (package)
<code>_ctypes_test</code>	<code>fractions</code>	<code>statut</code>	<code>trace</code>
<code>_ctypes</code>	<code>functools</code>	<code>subprocess</code>	<code>traceback</code>
<code>_ctypes</code>	<code>hashlib</code>	<code>subprocess</code>	<code>traceback</code>
<code>_ctypes</code>	<code>hmac</code>	<code>sys</code>	<code>tracemalloc</code>
<code>_ctypes</code>	<code>html</code> (package)	<code>pydoc</code>	<code>try</code>
<code>_ctypes</code>	<code>http.cookiejar</code>	<code>pydoc_data</code> (package)	<code>turtle</code>
<code>_ctypes</code>	<code>idna</code> (package)	<code>queue</code>	<code>types</code>
<code>_ctypes</code>	<code>imath</code> (package)	<code>random</code>	<code>unidecode</code> (package)
<code>_ctypes</code>	<code>importlib</code>	<code>re</code> (package)	<code>typing</code>
<code>_ctypes</code>	<code>importlib</code> (package)	<code>reprlib</code>	<code>unittest</code> (package)
<code>_ctypes</code>		<code>decorator</code>	<code>urllib</code> (package)
<code>_ctypes</code>		<code>dummy</code>	<code>urllib3</code>
<code>_ctypes</code>		<code>sched</code>	<code>venv</code> (package)
<code>_ctypes</code>		<code>secrets</code>	<code>warnings</code>

[index](#)  
conversion [/Users/navedahmedshaik/Documents/3-2/AIAC/conversion.py](#)

### conversion.py

Utility functions for number system conversions:

- Decimal to Binary
- Binary to Decimal
- Decimal to Hexadecimal

### Functions

#### `binary_to_decimal(b)`

Convert a binary string to its decimal integer representation.

#### `decimal_to_binary(n)`

Convert a decimal integer to its binary string representation.

#### `decimal_to_hexadecimal(n)`

Convert a decimal integer to its hexadecimal string representation.

## 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.

## SCREENSHOT OF GENERATED CODE:

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows a tree view of files and folders. Visible files include lab\_9\_1.py, calculator.py, conversion.py, course.py (selected), conversion.html, and calculator.html.
- EDITOR:** Displays the content of course.py. The code defines three functions: add\_course, remove\_course, and get\_course. Each function has a detailed docstring describing its purpose, parameters, and return value.
- PROBLEMS:** Shows 0 problems.
- OUTPUT:** Shows 0 output.
- DEBUG CONSOLE:** Shows 0 output.
- TERMINAL:** Shows 0 output.
- PORTS:** Shows 0 output.
- DOC:** Shows the generated documentation for course.py. It includes sections for NAME, DESCRIPTION, FUNCTIONS, and DATA.
- FILE:** Shows the path /Users/navedahmedshaik/Documents/3-2/AIAC/course.py.
- STATUS:** Shows the current file is course.py, with 31 lines of code, 0 errors, and 0 warnings. It also shows the Python version is 3.13.5.

[index](#)  
[course](#) /Users/navedahmedshaik/Documents/3-2/AIAC/course.py

## course.py

Simple course management module for adding, removing, and retrieving course information.

### Functions

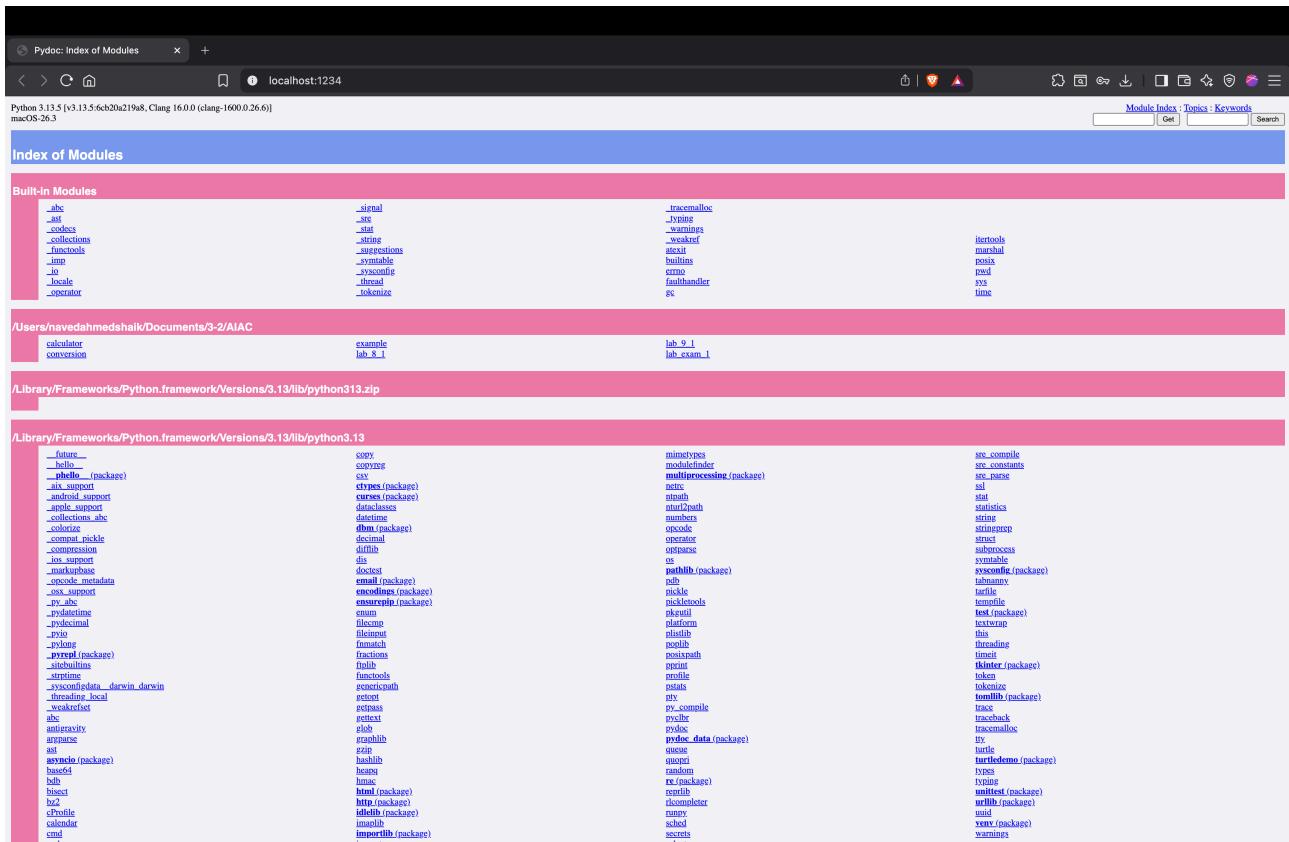
**add\_course(course\_id, name, credits)**  
Add a course with ID, name, and credit value.

**get\_course(course\_id)**  
Retrieve course details by ID. Returns course info or None if not found.

**remove\_course(course\_id)**  
Remove a course by its ID. Returns True if removed, False if not found.

### Data

**courses = {}**



The screenshot shows a browser window titled "Pydoc: Index of Modules" at the URL "localhost:1234". The page displays the Python module index. At the top, it shows the Python version (3.13.5) and build details. Below this is a search bar and a navigation bar with links like "Module Index", "Topics", and "Keywords". The main content area is divided into sections: "Built-in Modules" (listing standard library modules like abc, ast, collections, etc.), "/Users/navedahmedshaik/Documents/3-2/AIAC" (listing local modules like calculator, conversion), and "/Library/Frameworks/Python.framework/Versions/3.13/lib/python313.zip" (listing built-in modules like copy, os, random, etc.). Each module entry includes a brief description and a list of its members.