Lab5: General Programming

This document gives step-by-step guide to finish Lab5, but feel to add in a few of your own attempts.

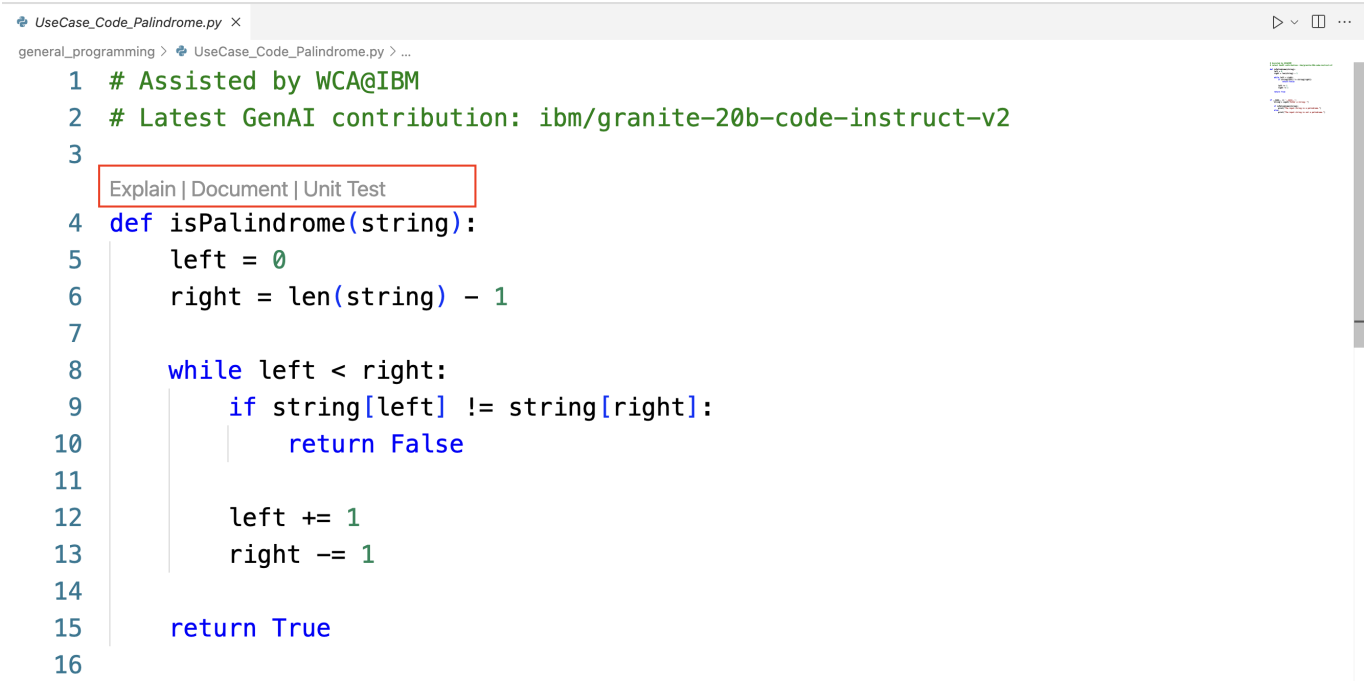
Lab5 covers:

- General programming with other languages, including
  - code auto completion
  - code explanation
  - code documentation
  - unit test generation
  - code translation

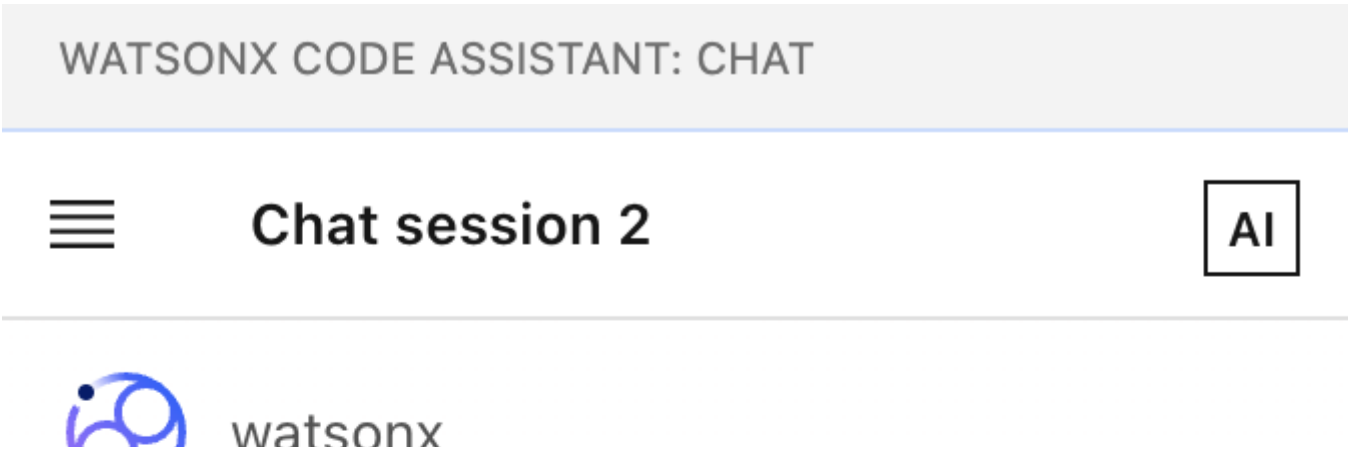
> **Note:** To help you get started, we've provided some sample code in the `lab5-sample-code` folder.

To get started, open a new chat session.

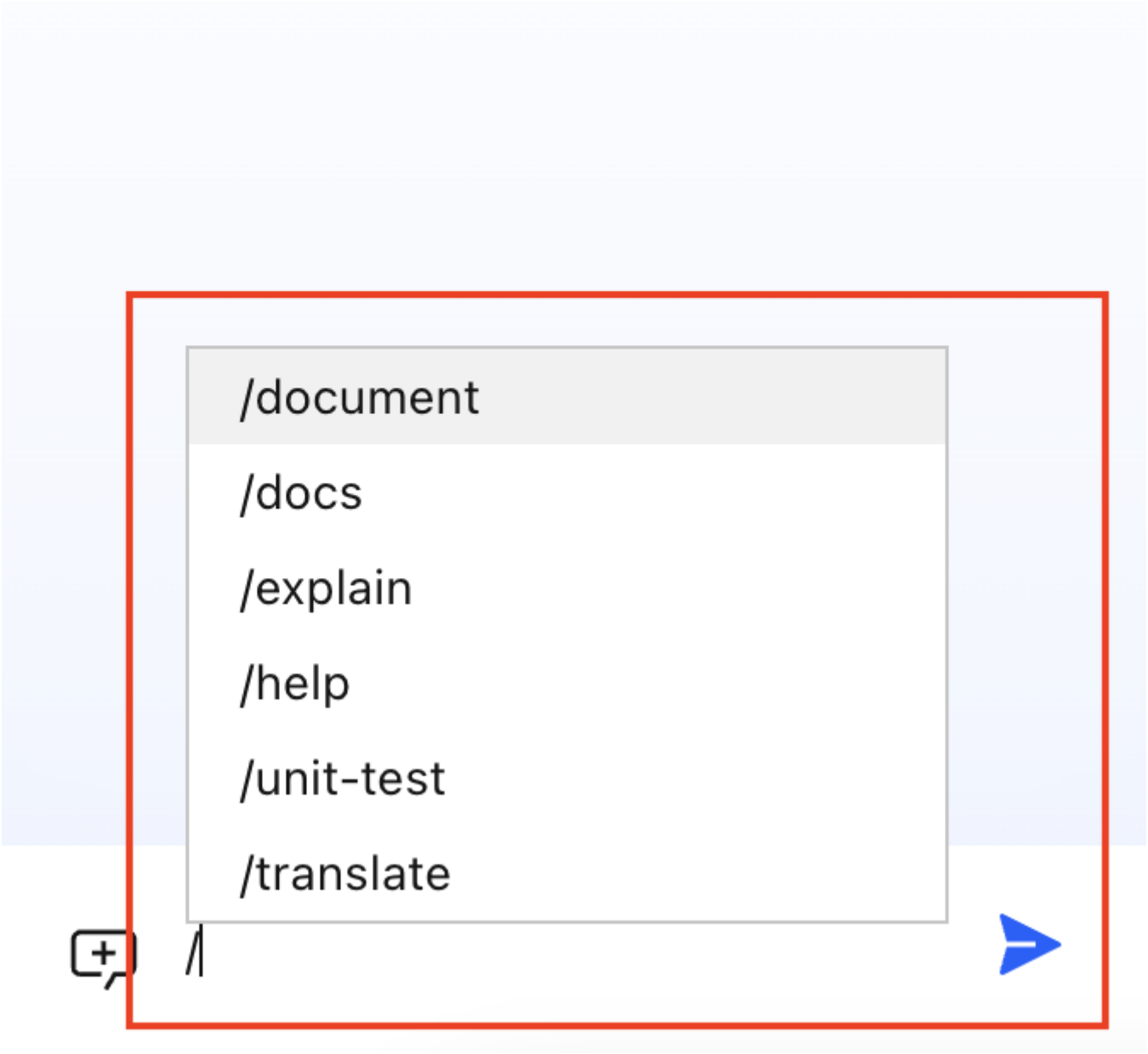
You can preview options for general programming capabilities in any inline functions.



They align with options starting with backslash provided in the chat window.



Hi, how can I help you today?



1. Code Auto Completion

When developers write code, WCA will provide real-time inline suggestions.

```
Explain | Document | Unit Test
def isPalindrome(string):
    left = 0
    right = len(string) - 1

    while left < right:
        if string[left] != string[right]:
            return False

        left += 1
        right -= 1

    return True
```

1/1 Accept Tab Accept Word ⌘ → ···

```
def check_prime(num):
```

2. Code Explanation

WCA can help explain the details and functions defined in the code sample.

This you can do by clicking the explain option on top of your functions.

Or, you can explain the entire file by typing the following in chat:

```
/explain @UseCase_Code_Palindrome.py
```

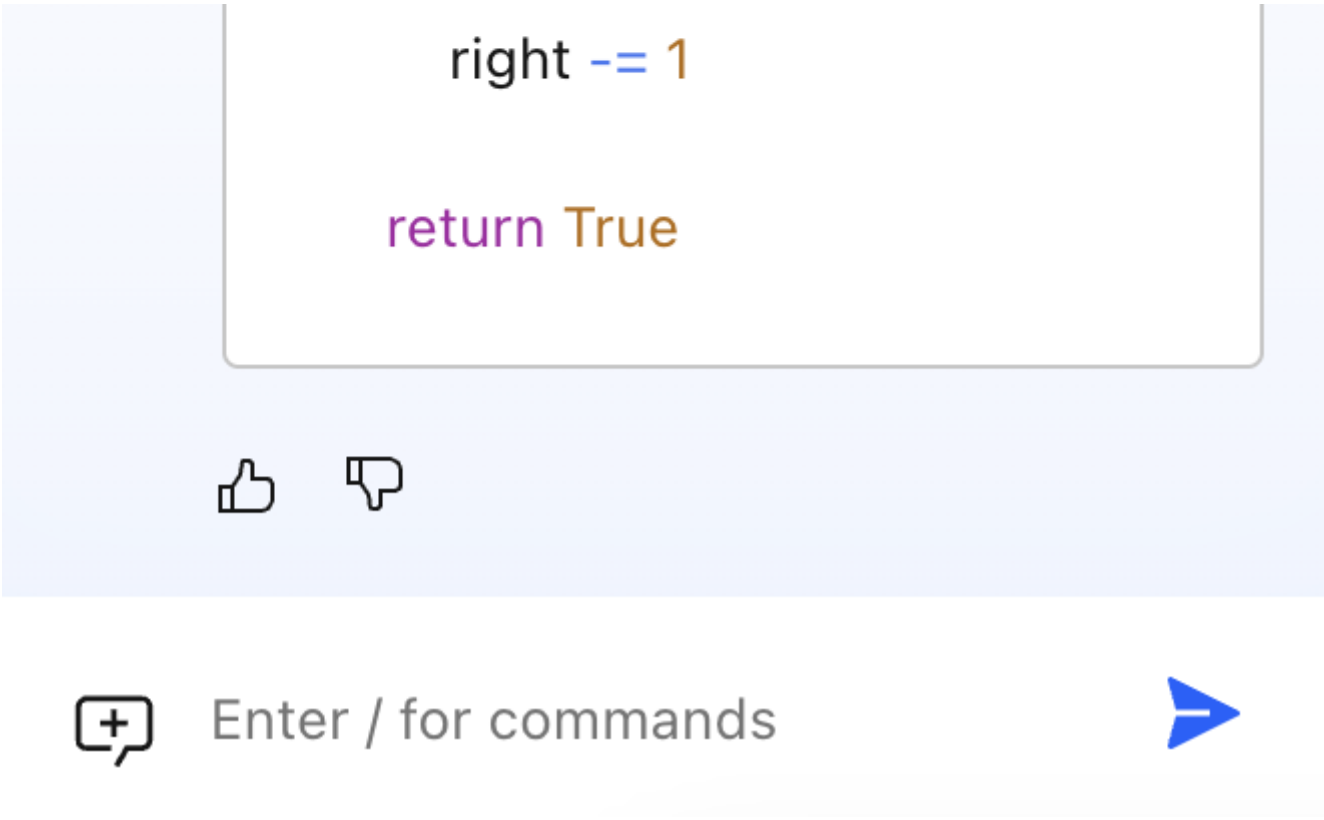
> **Note:** Always use the @ symbol to refer to a specific file. This syntax does not appear in the conversation history.

3. Code Documentation

WCA can help generate documentation strings for a given code sample.

> **Note:** Don't forget the @ symbol to refer to a file.

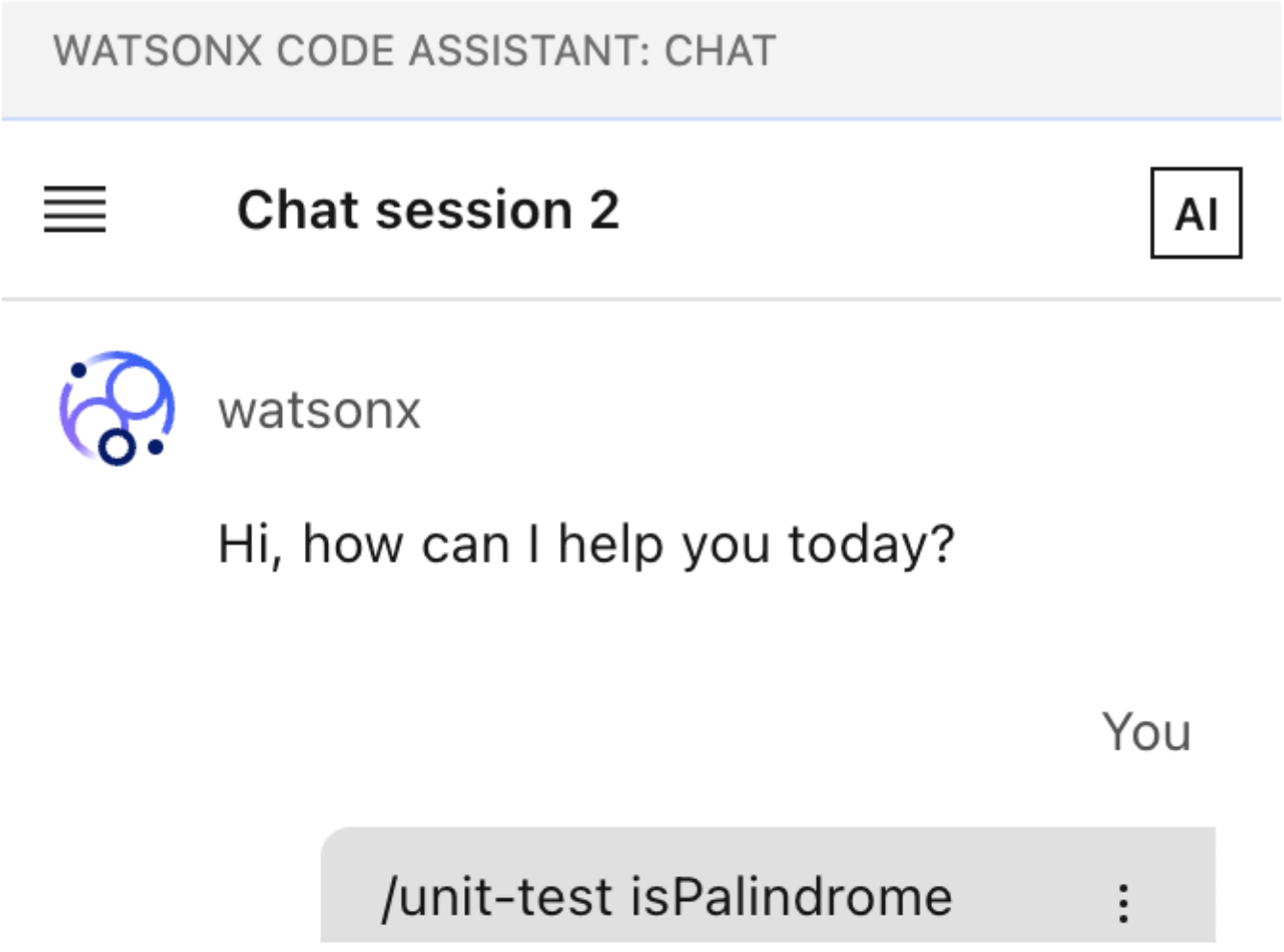
```
# Assisted by watsonx Code Assistant
def isPalindrome(string):
    """
    Check if a given string is a palindrome.

    Args:
        string (str): The string to check.

    Returns:
        bool: True if the string is a palindrome, False otherwise.
    """
    left = 0
    right = len(string) - 1

    while left < right:
        if string[left] != string[right]:
            return False

        left += 1
```



4. Unit test generation

Similar to what we see for Java, unit test generation is available for other languages.

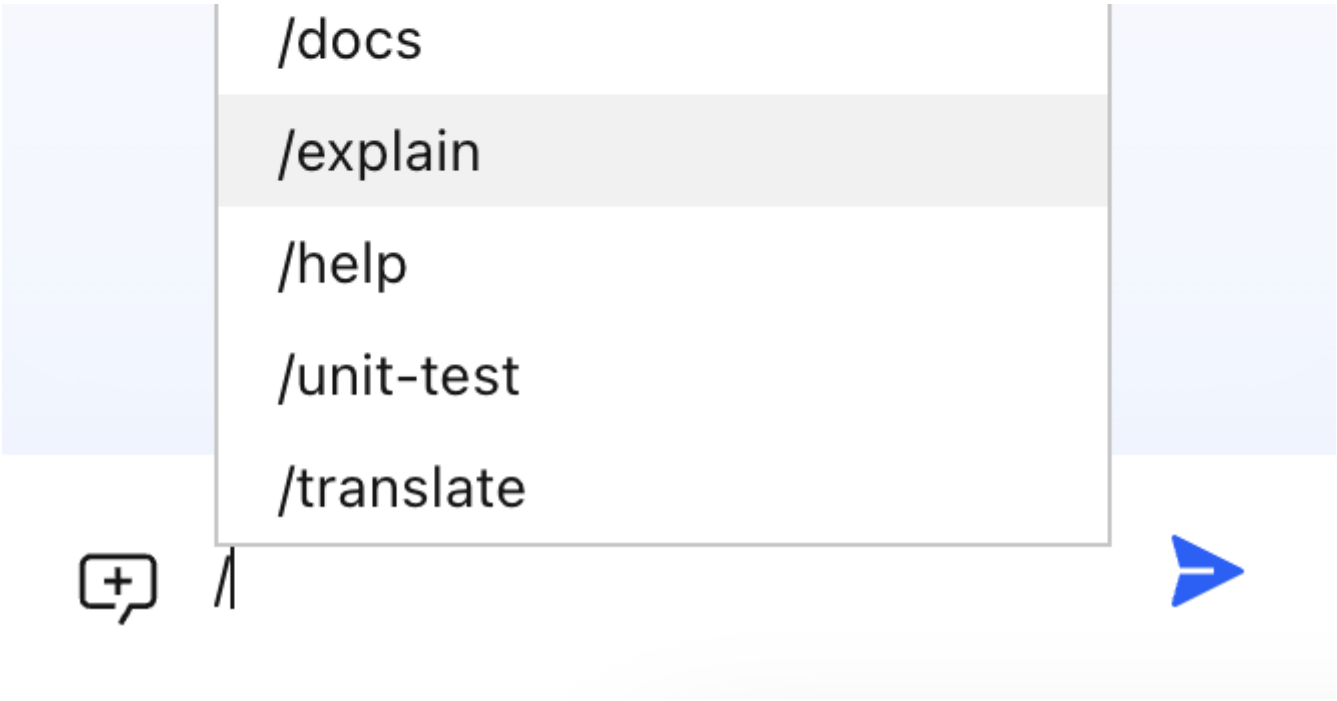
> **Note:** Don't forget the @ symbol to refer to a file.



watsonx

```
# Assisted by watsonx Code Assistant
def test_isPalindrome():
    assert isPalindrome("racecar") == True
    assert isPalindrome("hello") == False
    assert isPalindrome("A man, a plan, a canal, Panama") == True
    assert isPalindrome("12345") == False
```

/document



5. Code Translation

Code translation is provided via the following syntax:

```
/translate from SOURCE_LANGUAGE to TARGET_LANGUAGE
@REFERENCE_FILE|CODE_SNIPPET
```

> **Note:** You might want to start a new chat session for this command.

Please note from SOURCE_LANGUAGE is an optional argument, and can be skipped while doing translation.

For this lab, we will conver the python palindrom code to go language. Try the following prompt:

```
/translate from python to go @UseCase_Code_Palindrome.py
```

```go
// Assisted by watsonx Code Assistant
package main

import "fmt"

func isPalindrome(s string) bool {
    left, right := 0, len(s)-1
    for left < right {
        if s[left] != s[right] {
            return false
        }
        left++
        right--
    }
    return true
}

func main() {
    var s string
    fmt.Println("Enter a string:")
    fmt.Scanln(&s)
    if isPalindrome(s) {
        fmt.Println("The input string is a palindrome.")
    } else {
        fmt.Println("The input string is not a palindrome.")
    }
}
```