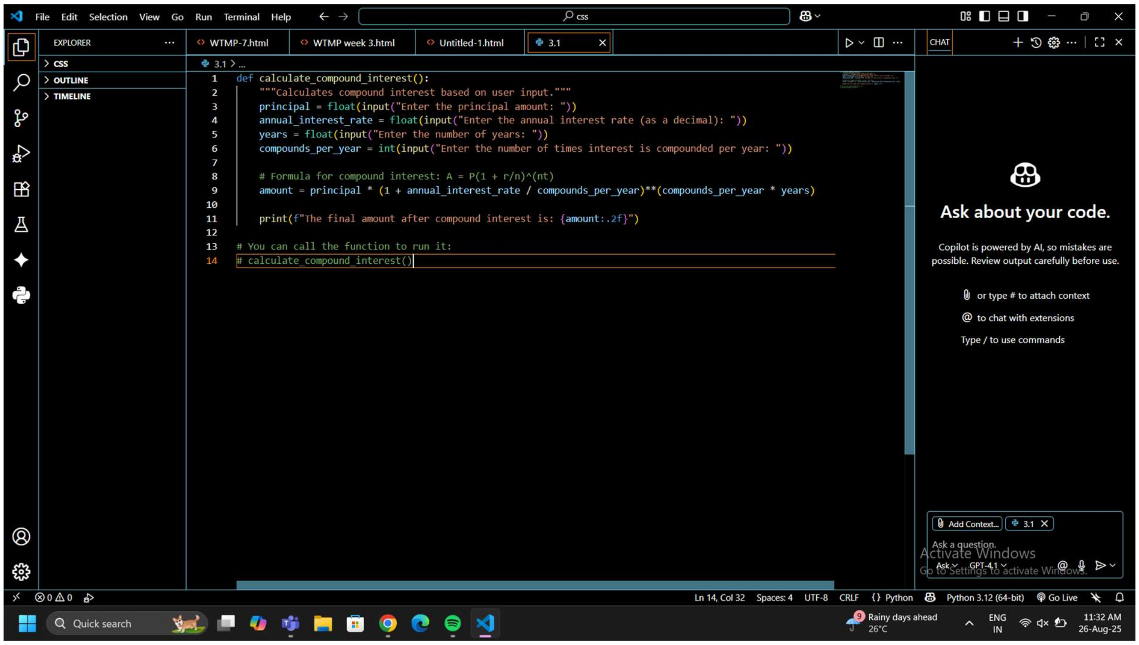
AI CODING

Lab-3.2

Task-1



```python
def calculate_compound_interest():
    """Calculates compound interest based on user input."""
    principal = float(input("Enter the principal amount: "))
    annual_interest_rate = float(input("Enter the annual interest rate (as a decimal): "))
    years = float(input("Enter the number of years: "))
    compounds_per_year = int(input("Enter the number of times interest is compounded per year: "))

    # Formula for compound interest: A = P(1 + r/n)^(nt)
    amount = principal * (1 + annual_interest_rate / compounds_per_year)**(compounds_per_year * years)

    print(f"The final amount after compound interest is: {amount:.2f}")

# You can call the function to run it:
# calculate_compound_interest()
```

File Edit Selection View Go Run Terminal Help
WTMP-7.html
WTMP week 3.html
Untitled-1.html
3.1
EXPLORER
CSS
OUTLINE
TIMELINE
CHAT
Ask about your code.
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.
or type # to attach context
to chat with extensions
Type / to use commands
Add Context...
Ask a question.
Ln 14, Col 32 Spaces: 4 UTF-8 CRLF Python Python 3.12 (64-bit) Go Live
Quick search
Rainy days ahead 26°C
11:32 AM 26-Aug-25

Output:

- Principal amount: 1000
- Annual interest rate: 0.05
- Number of years: 10
- Number of times interest is compounded per year: 4

The output would be:

```
The final amount after compound interest is: 1648.72
```

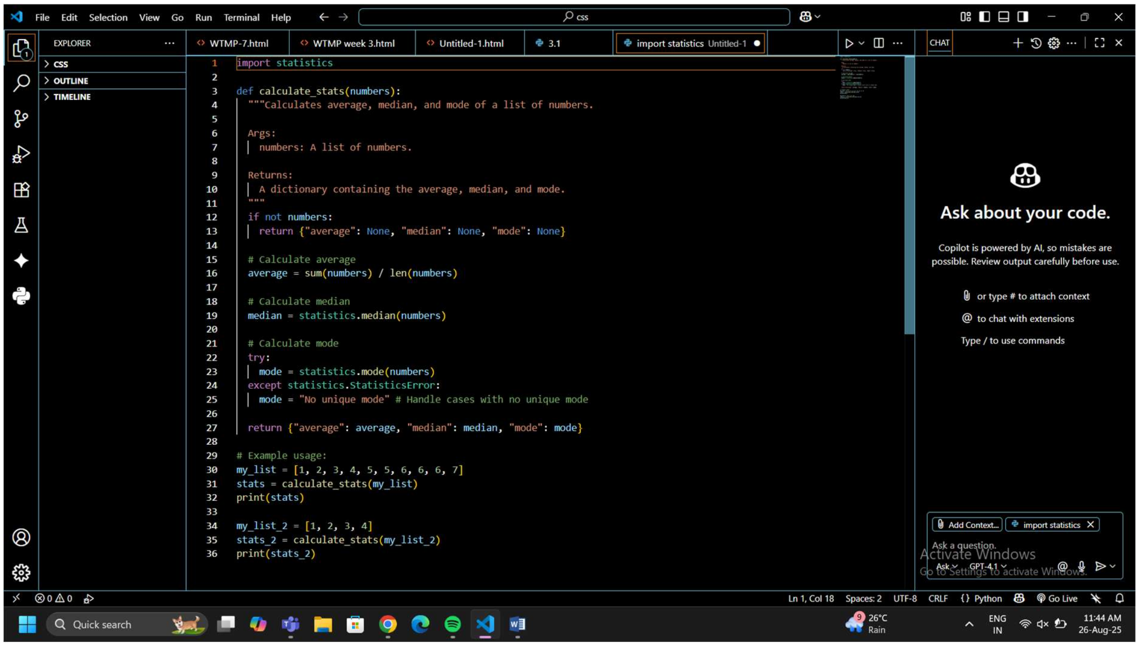
Explanation:

This code defines a Python function called `calculate_compound_interest` that calculates the compound interest based on user input. Here's a breakdown:

- `def calculate_compound_interest():`: This line defines the function.
- `"""Calculates compound interest based on user input."""`: This is a docstring that explains what the function does.
- `principal = float(input("Enter the principal amount: "))`: This line prompts the user to enter the principal amount and converts the input to a floating-point number.

- `annual_interest_rate = float(input("Enter the annual interest rate (as a decimal): "))`: This line prompts the user for the annual interest rate and converts it to a float.

Test-2



```python
import statistics

def calculate_stats(numbers):
  """Calculates average, median, and mode of a list of numbers.

  Args:
    numbers: A list of numbers.

  Returns:
    A dictionary containing the average, median, and mode.
  """
  if not numbers:
    return {"average": None, "median": None, "mode": None}

  # Calculate average
  average = sum(numbers) / len(numbers)

  # Calculate median
  median = statistics.median(numbers)

  # Calculate mode
  try:
    mode = statistics.mode(numbers)
  except statistics.StatisticsError:
    mode = "No unique mode" # Handle cases with no unique mode

  return {"average": average, "median": median, "mode": mode}

# Example usage:
my_list = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]
stats = calculate_stats(my_list)
print(stats)

my_list_2 = [1, 2, 3, 4]
stats_2 = calculate_stats(my_list_2)
print(stats_2)
```

Ask about your code.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

to chat with extensions

Type / to use commands

Output:

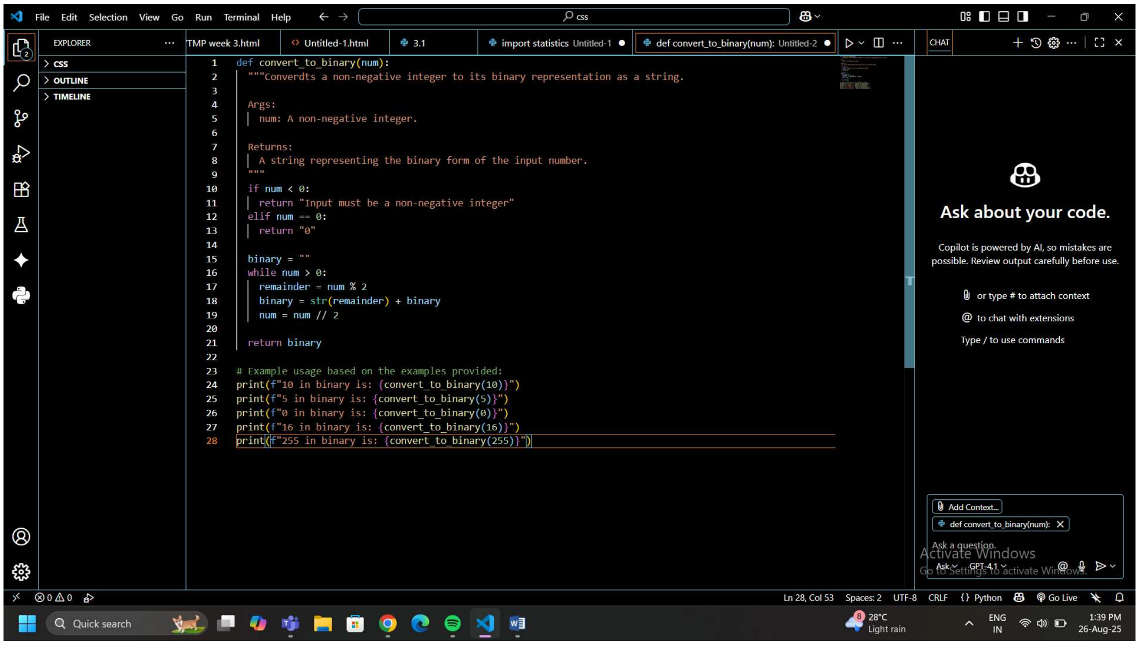
```
{'average': 4.5, 'median': 5.0, 'mode': 6}
{'average': 2.5, 'median': 2.5, 'mode': 1}
```

Explanation:

This code defines a Python function called `caluculate stats`

- `import statistics`: This line imports the `statistics` module, which provides functions for calculating statistical properties of data.
- `def calculate_stats(numbers):`: This line defines the function `calculate_stats` that accepts one argument, `numbers`, which is expected to be a list.
- `"""Calculates average, median, and mode of a list of numbers. ... """`: This is a docstring explaining the function's purpose, arguments, and return value.
- `if not numbers:`: This checks if the input list `numbers` is empty.
- `return {"average": None, "median": None, "mode": None}`: If the list is empty, the function returns a dictionary with `None` values for average, median, and mode, as these cannot be calculated for an empty list.

Test3:


File Edit Selection View Go Run Terminal Help
css
EXPLORER
CSS
OUTLINE
TIMELINE
TMP week 3.html
Untitled-1.html
3.1
import statistics Untitled-1
def convert_to_binary(num): Untitled-2
CHAT
Ask about your code.
Copilot is powered by AI, so mistakes are possible. Review output carefully before use.
or type # to attach context
@ to chat with extensions
Type / to use commands
Add Context...
def convert_to_binary(num):
Ask a question.
Ask
GPT-4.1
Activate Windows
Go to Settings to activate Windows.
Ln 28, Col 53 Spaces: 2 UTF-8 CRLF {} Python Go Live
Quick search
28°C
Light rain
ENG
IN
1:39 PM
26-Aug-25

```python
def convert_to_binary(num):
  """Converdts a non-negative integer to its binary representation as a string.

  Args:
    num: A non-negative integer.

  Returns:
    A string representing the binary form of the input number.
  """
  if num < 0:
    return "Input must be a non-negative integer"
  elif num == 0:
    return "0"

  binary = ""
  while num > 0:
    remainder = num % 2
    binary = str(remainder) + binary
    num = num // 2

  return binary

# Example usage based on the examples provided:
print(f"10 in binary is: {convert_to_binary(10)}")
print(f"5 in binary is: {convert_to_binary(5)}")
print(f"0 in binary is: {convert_to_binary(0)}")
print(f"16 in binary is: {convert_to_binary(16)}")
print(f"255 in binary is: {convert_to_binary(255)}")
```

Output:

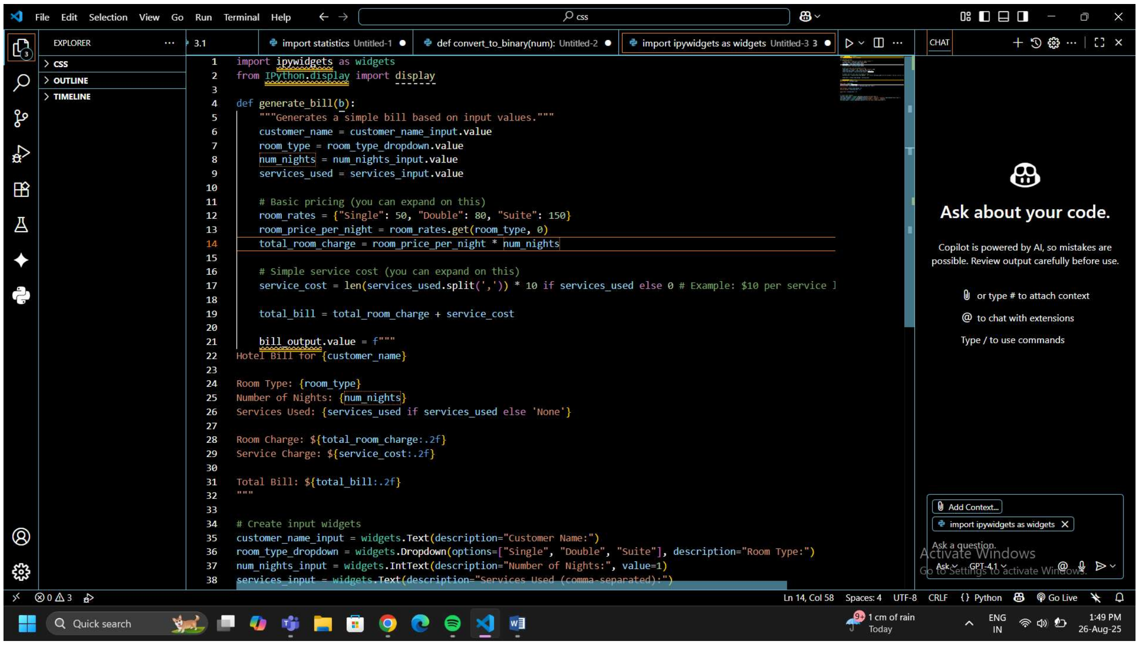
```
10 in binary is: 1010
5 in binary is: 101
0 in binary is: 0
16 in binary is: 10000
255 in binary is: 11111111
```

Explanation: Python function called `convert_to_binary` that takes a non-negative integer as input and returns its binary representation as a string. Here's a breakdown:

- `def convert_to_binary(num):`: This line defines the function `convert_to_binary` that accepts one argument, `num`.
- `"""Converts a non-negative integer to its binary representation as a string. ... """`: This is a docstring explaining the function's purpose, arguments, and return value.
- `if num < 0:`: This checks if the input number is negative.
- `return "Input must be a non-negative integer"`: If the number is negative, the function returns an error message string.
- `elif num == 0:`: This checks if the input number is 0.
- `return "0"`: If the number is 0, the function returns the string "0", which is the binary representation of 0.
- `binary = ""`: This initializes an empty string called `binary` which will store the binary digits.
- `while num > 0:`: This starts a `while` loop that continues as long as the value of `num` is greater than 0.
- `remainder = num % 2`: This calculates the remainder when `num` is divided by 2. This remainder will be either 0 or 1, which are the binary digits.
- `binary = str(remainder) + binary`: This converts the `remainder` to a string and prepends it to the `binary` string. This builds the binary representation in reverse order.

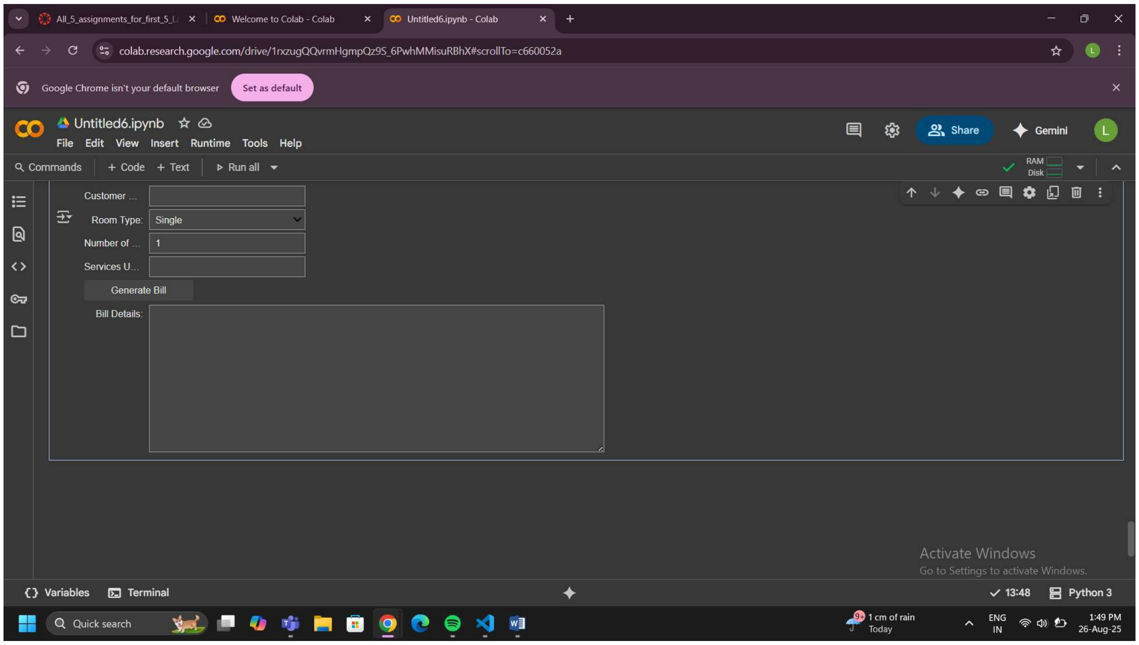
- `num = num // 2`: This performs integer division of `num` by 2, effectively moving to the next bit in the binary conversion.
- `return binary`: After the loop finishes (when `num` becomes 0), the function returns the `binary` string, which now holds the correct binary representation.
- `# Example usage based on the examples provided:`: This is a comment indicating the start of example code.
- `print(f"10 in binary is: {convert_to_binary(10)}")`: This calls the function with 10 and prints the result in a formatted string.

Test-4



```python
import ipywidgets as widgets
from IPython.display import display

def generate_bill(b):
    """Generates a simple bill based on input values."""
    customer_name = customer_name_input.value
    room_type = room_type_dropdown.value
    num_nights = num_nights_input.value
    services_used = services_input.value

    # Basic pricing (you can expand on this)
    room_rates = {"Single": 50, "Double": 80, "Suite": 150}
    room_price_per_night = room_rates.get(room_type, 0)
    total_room_charge = room_price_per_night * num_nights

    # Simple service cost (you can expand on this)
    service_cost = len(services_used.split(',')) * 10 if services_used else 0 # Example: $10 per service

    total_bill = total_room_charge + service_cost

    bill_output.value = f"""
Hotel Bill for {customer_name}

Room Type: {room_type}
Number of Nights: {num_nights}
Services Used: {services_used if services_used else 'None'}

Room Charge: ${total_room_charge:.2f}
Service Charge: ${service_cost:.2f}

Total Bill: ${total_bill:.2f}
"""

# Create input widgets
customer_name_input = widgets.Text(description="Customer Name:")
room_type_dropdown = widgets.Dropdown(options=["Single", "Double", "Suite"], description="Room Type:")
num_nights_input = widgets.IntText(description="Number of Nights:", value=1)
services_input = widgets.Text(description="Services Used (comma-separated):")
```

Output:


All_5_assignments_for_first_5_L
Welcome to Colab - Colab
Untitled6.ipynb - Colab
colab.research.google.com/drive/1rxzugQQvrmHgmpQz9S_6PwhMMisuRBhX#scrollTo=c660052a
Google Chrome isn't your default browser
Set as default
Untitled6.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM
Disk
Activate Windows
Go to Settings to activate Windows.
Variables
Terminal
13:48
Python 3
Quick search
1 cm of rain
Today
ENG
IN
1:49 PM
26-Aug-25

Customer ...

Room Type: Single

Number of ... 1

Services U...

Generate Bill

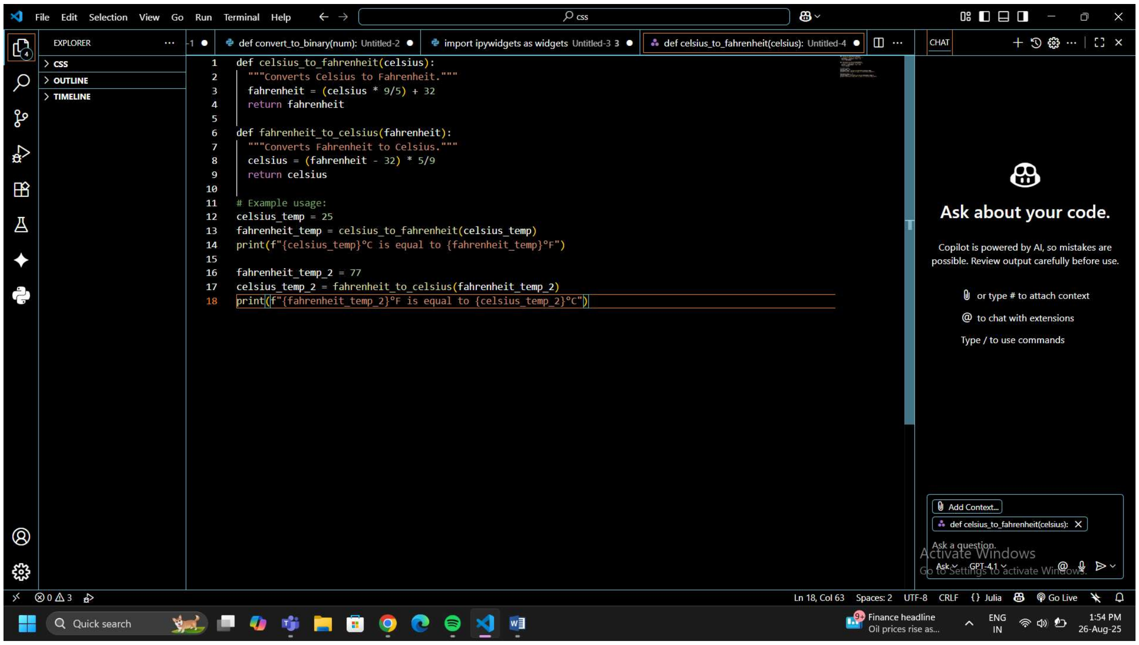
Bill Details:

Explanation:

The selected code snippet is the beginning of the `create_hotel_billing_ui` function and sets up the main window for the user interface using the `tkinter` library.

- `import tkinter as tk`: This line imports the `tkinter` library, which is the standard Python interface to the Tcl/Tk GUI toolkit. It's imported with the alias `tk` for easier use.
- `from tkinter import ttk`: This line imports the `ttk` module from `tkinter`. `ttk` stands for "themed widgets" and provides access to themed widget set, which gives the GUI a more native look and feel across different operating systems.

Task-5



```python
def celsius_to_fahrenheit(celsius):
  """Converts Celsius to Fahrenheit."""
  fahrenheit = (celsius * 9/5) + 32
  return fahrenheit

def fahrenheit_to_celsius(fahrenheit):
  """Converts Fahrenheit to Celsius."""
  celsius = (fahrenheit - 32) * 5/9
  return celsius

# Example usage:
celsius_temp = 25
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp}°C is equal to {fahrenheit_temp}°F")

fahrenheit_temp_2 = 77
celsius_temp_2 = fahrenheit_to_celsius(fahrenheit_temp_2)
print(f"{fahrenheit_temp_2}°F is equal to {celsius_temp_2}°C")
```

Ask about your code.

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

or type # to attach context

@ to chat with extensions

Type / to use commands

Output:

```
25°C is equal to 77.0°F
77°F is equal to 25.0°C
```

Explanation: code defines a Python function called `celsius_to_fahrenheit`.

- `def celsius_to_fahrenheit(celsius):`: This line defines the function named `celsius_to_fahrenheit` that takes one argument, `celsius`. This argument is expected to be a numerical value representing a temperature in Celsius.
- `"""Converts Celsius to Fahrenheit."""`: This is a docstring that explains what the function does.
- `fahrenheit = (celsius * 9/5) + 32`: This is the core of the function where the conversion happens. It applies the standard formula to convert Celsius to Fahrenheit: multiply the Celsius temperature by 9/5 and then add 32. The result is stored in the `fahrenheit` variable.
- `return fahrenheit`: This line returns the calculated Fahrenheit temperature.