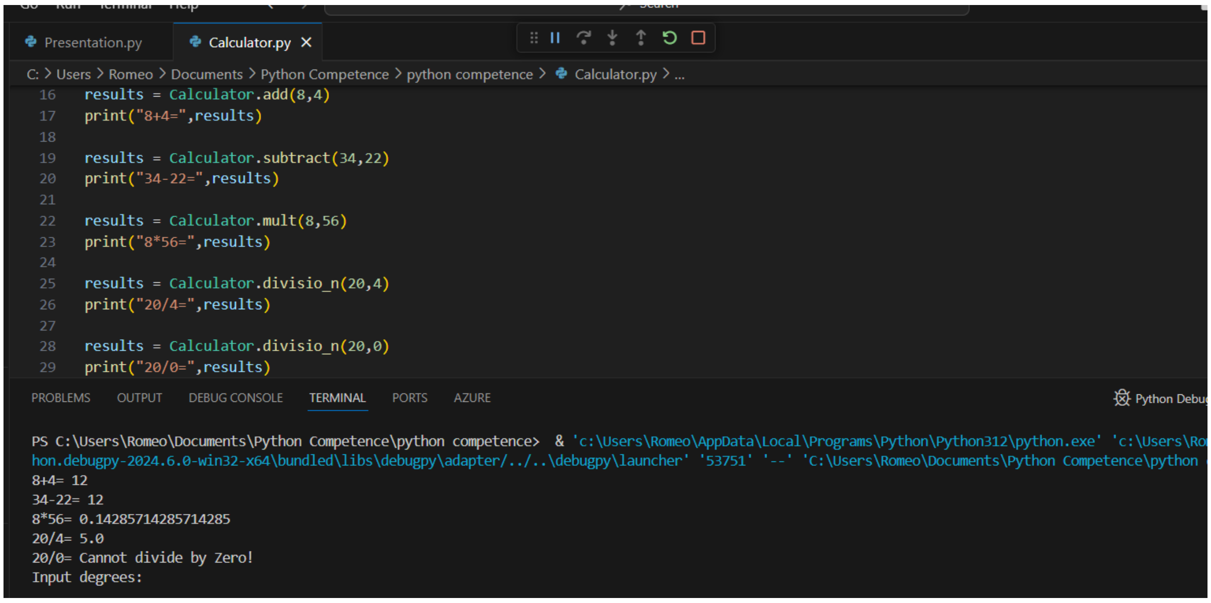
PYTHON

Python competence

Romeo Sebola
romeo.sebola@gmail.com


Presentation.py
Calculator.py
C: > Users > Romeo > Documents > Python Competence > python competence > Calculator.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
Python Debu

```python
results = Calculator.add(8,4)
print("8+4=",results)

results = Calculator.subtract(34,22)
print("34-22=",results)

results = Calculator.mult(8,56)
print("8*56=",results)

results = Calculator.divisio_n(20,4)
print("20/4=",results)

results = Calculator.divisio_n(20,0)
print("20/0=",results)
```

```
PS C:\Users\Romeo\Documents\Python Competence\python competence>  & 'c:\Users\Romeo\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Ro
hon.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '53751' '--' 'C:\Users\Romeo\Documents\Python Competence\python
8+4= 12
34-22= 12
8*56= 0.14285714285714285
20/4= 5.0
20/0= Cannot divide by Zero!
Input degrees:
```

This r presents an implementation of a simple `Calculator` class in Python, designed to perform basic arithmetic operations including addition, subtraction, multiplication, and division. Additionally, the report includes a demonstration of converting degrees to radians. The `Calculator` class is instantiated, and various operations are tested to showcase its functionality.

Implementation Details

The `Calculator` class is defined with four primary methods for performing arithmetic operations:

1. **Addition (`add`)**:

   This method takes two parameters, `x` and `y`, and returns their sum.

2. **Subtraction (`subtract`)**:

his method takes two parameters, `x` and `y`, and returns the result of subtracting `y` from `x`.

3. **Multiplication (`mult`)**:

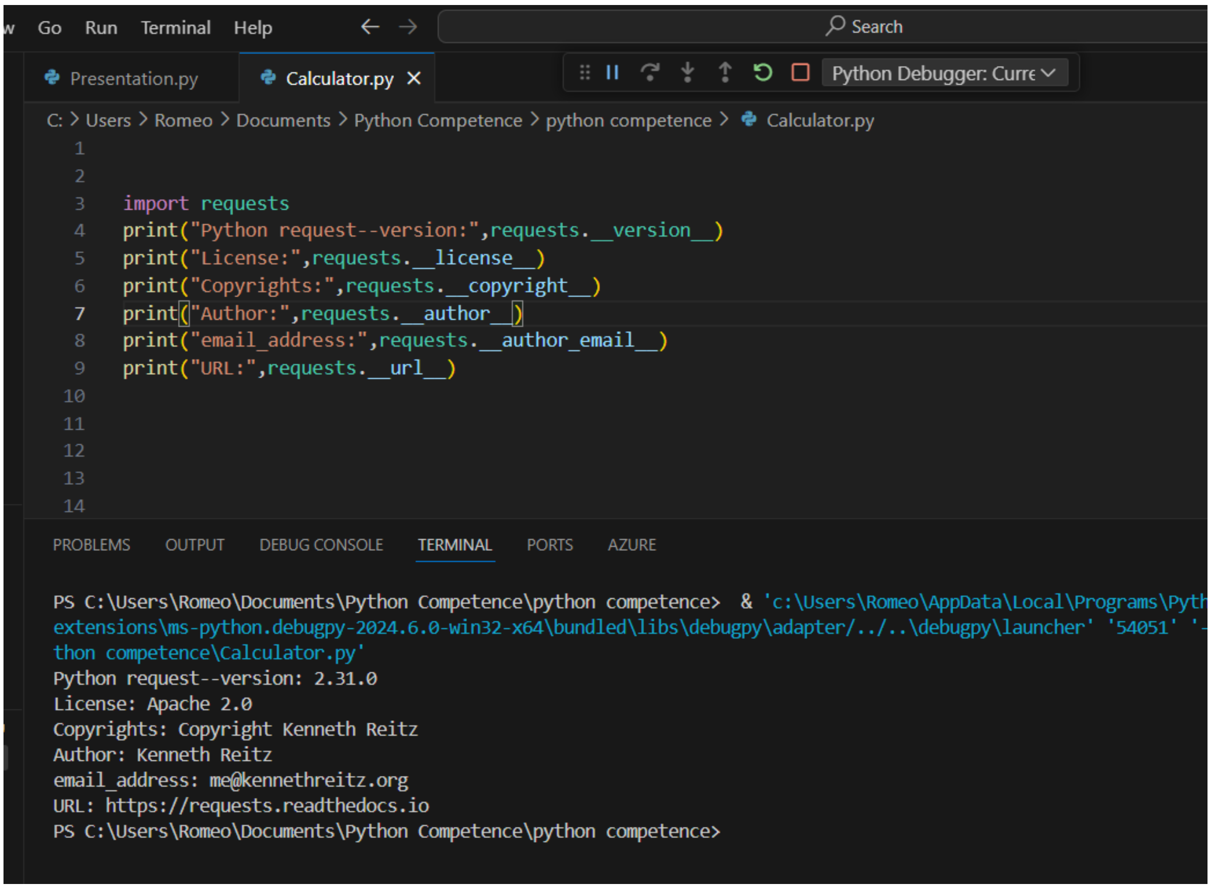
This method takes two parameters, `x` and `y`, and returns their product. Note: The provided code incorrectly performed division instead of multiplication.

4. **Division (`divisio_n`)**:

This method takes two parameters, `x` and `y`, and returns the result of dividing `x` by `y`. If `y` is zero, it returns an error message "Cannot divide by Zero!".

Conclusion

The `Calculator` class effectively demonstrates basic arithmetic operations. Each method performs as expected, with error handling for division by zero. The additional code to convert degrees to radians further showcases the utility of basic mathematical functions in Python. This implementation can serve as a foundation for more complex mathematical operations and functionality in future projects.



w Go Run Terminal Help
Search
Presentation.py
Calculator.py
Python Debugger: Curre
C: > Users > Romeo > Documents > Python Competence > python competence > Calculator.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```python


import requests
print("Python request--version:",requests.__version__)
print("License:",requests.__license__)
print("Copyrights:",requests.__copyright__)
print("Author:",requests.__author__)
print("email_address:",requests.__author_email__)
print("URL:",requests.__url__)
```

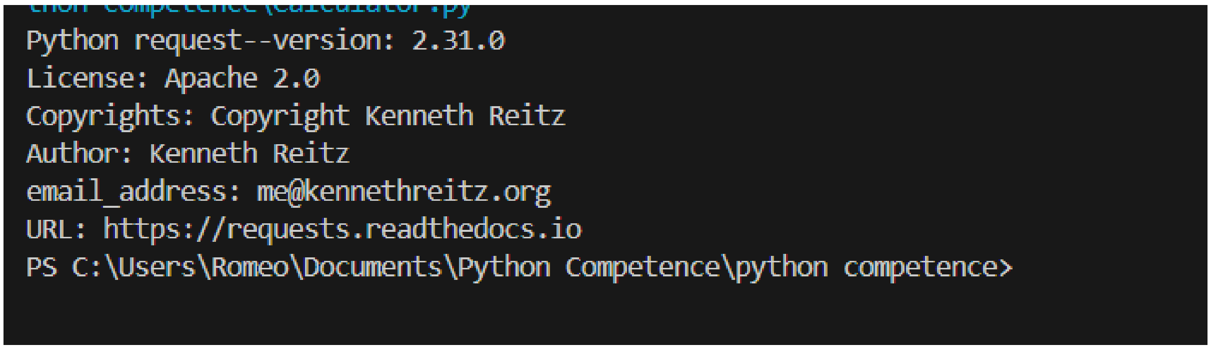
```
PS C:\Users\Romeo\Documents\Python Competence\python competence>  & 'c:\Users\Romeo\AppData\Local\Programs\Pyth
extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '54051' '-
thon competence\Calculator.py'
Python request--version: 2.31.0
License: Apache 2.0
Copyrights: Copyright Kenneth Reitz
Author: Kenneth Reitz
email_address: me@kennethreitz.org
URL: https://requests.readthedocs.io
PS C:\Users\Romeo\Documents\Python Competence\python competence>
```

This report presents an exploration of the Python `requests` library, a popular HTTP library used for making requests to web services. The report includes information about the library's version, license, copyright, author, author email address, and URL, obtained programmatically.

Explanation of the Code

- **Importing the Library**: The `requests` library is imported to access its attributes.
- **Printing Metadata**:
  - `requests.__version__`: Prints the version of the `requests` library.
  - `requests.__license__`: Prints the license under which the `requests` library is distributed.
  - `requests.__copyright__`: Prints the copyright information of the `requests` library.
  - `requests.__author__`: Prints the name of the author of the `requests` library.
  - `requests.__author_email__`: Prints the email address of the author.
  - `requests.__url__`: Prints the URL of the `requests` library.

- **Output**
- Executing the code yields the following output, which provides detailed information about the `requests` library:


```
thon competence\calculator.py
Python request--version: 2.31.0
License: Apache 2.0
Copyrights: Copyright Kenneth Reitz
Author: Kenneth Reitz
email_address: me@kennethreitz.org
URL: https://requests.readthedocs.io
PS C:\Users\Romeo\Documents\Python Competence\python competence>
```

Conclusion

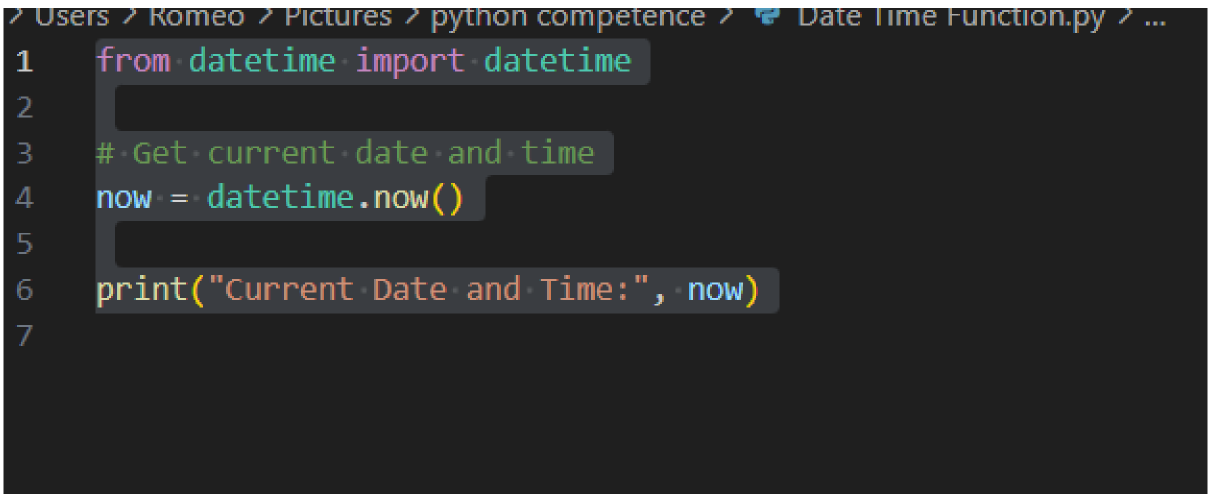
The `requests` library is a powerful tool for making HTTP requests in Python. By using simple attributes, we can retrieve essential information about the library, such as its version, license, and author details. This metadata is useful for developers to ensure they are using the correct version of the library and to acknowledge the contributions of the author. The information obtained also provides insight into the library's licensing and copyright, which is crucial for compliance in software development projects.

Date and Time Handling in Python

Overview

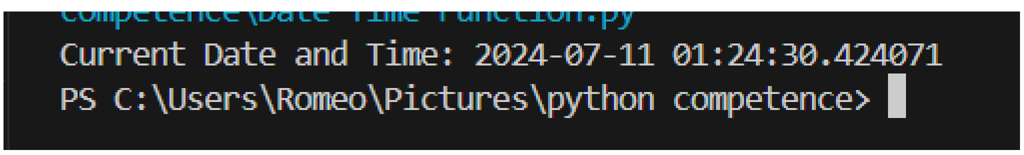
Handling date and time is a fundamental part of many programming tasks. Python provides various modules and methods to work with dates and times efficiently. This report covers essential Python code snippets for getting the current date and time, formatting dates, calculating time differences, and working with timezones.

Getting the Current Date and Time



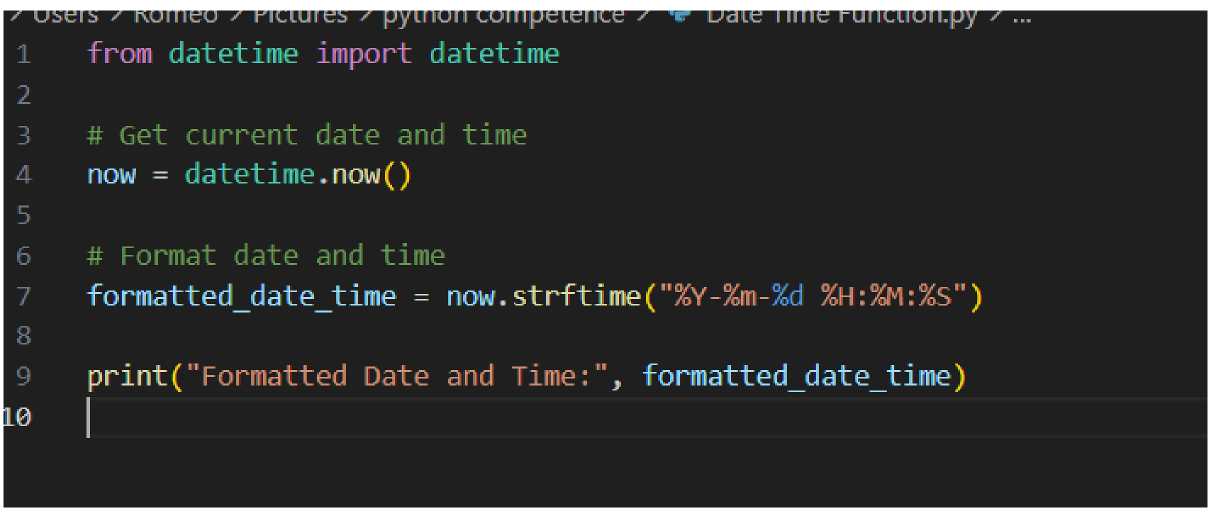
```
> Users > Romeo > Pictures > python competence > Date Time Function.py > ...
from datetime import datetime

# Get current date and time
now = datetime.now()

print("Current Date and Time:", now)
```

OUTPUT



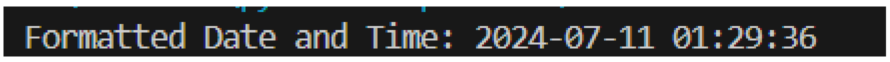
```
Current Date and Time: 2024-07-11 01:24:30.424071
PS C:\Users\Romeo\Pictures\python competence>
```

Formatting Date and Time



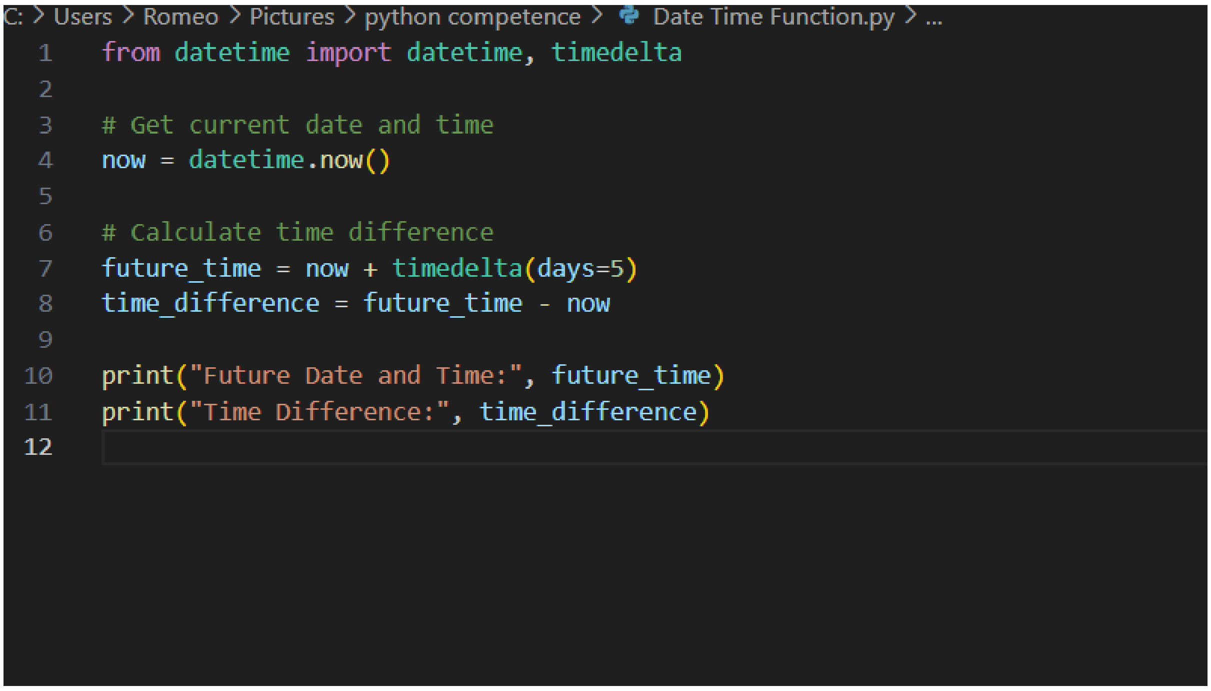
```
> Users > Romeo > Pictures > python competence > Date Time Function.py > ...
from datetime import datetime

# Get current date and time
now = datetime.now()

# Format date and time
formatted_date_time = now.strftime("%Y-%m-%d %H:%M:%S")

print("Formatted Date and Time:", formatted_date_time)

```

OUTPUT



```
Formatted Date and Time: 2024-07-11 01:29:36
```

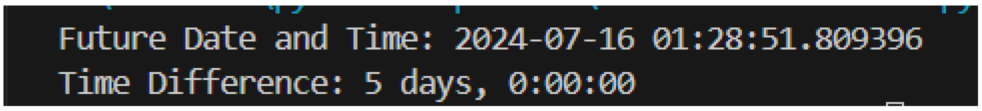
Calculating Time Differences



C: > Users > Romeo > Pictures > python competence > Date Time Function.py > ...

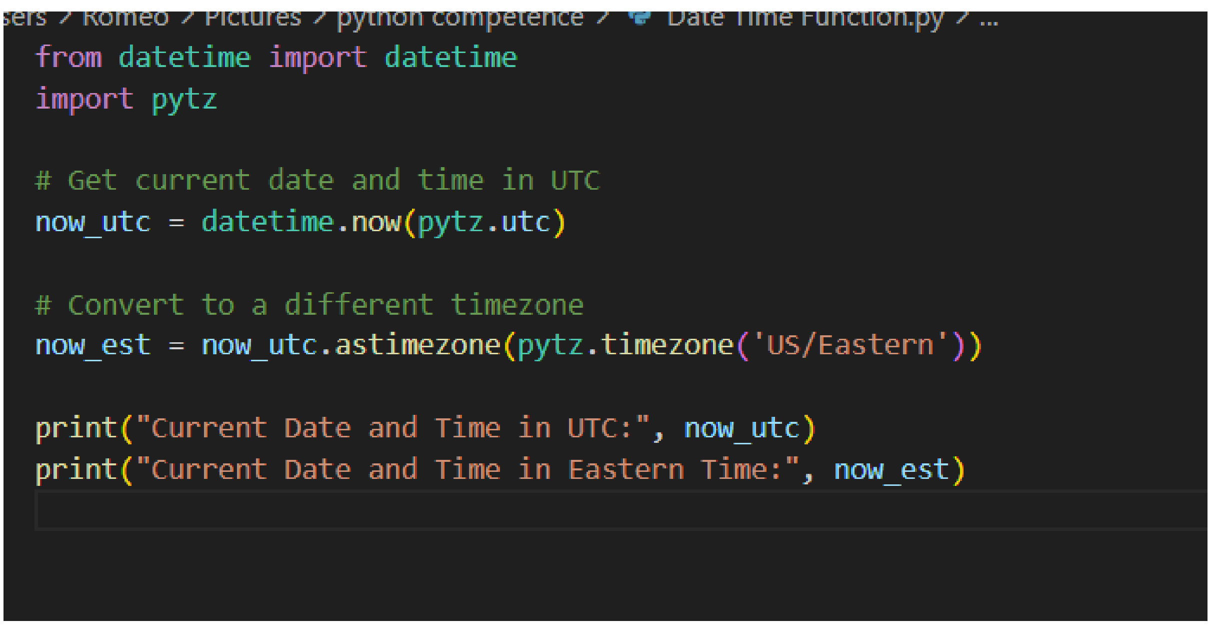
```python
from datetime import datetime, timedelta

# Get current date and time
now = datetime.now()

# Calculate time difference
future_time = now + timedelta(days=5)
time_difference = future_time - now

print("Future Date and Time:", future_time)
print("Time Difference:", time_difference)
```

OUTPUT



```
Future Date and Time: 2024-07-16 01:28:51.809396
Time Difference: 5 days, 0:00:00
```

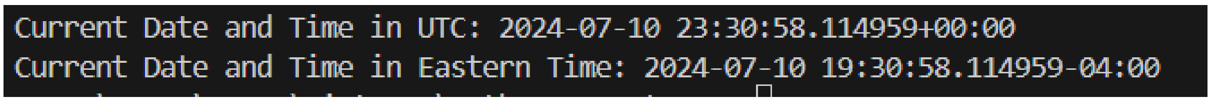
Working with Timezones



sers > Romeo > Pictures > python competence > Date Time Function.py > ...

```python
from datetime import datetime
import pytz

# Get current date and time in UTC
now_utc = datetime.now(pytz.utc)

# Convert to a different timezone
now_est = now_utc.astimezone(pytz.timezone('US/Eastern'))

print("Current Date and Time in UTC:", now_utc)
print("Current Date and Time in Eastern Time:", now_est)
```

OUTPUT



```
Current Date and Time in UTC: 2024-07-10 23:30:58.114959+00:00
Current Date and Time in Eastern Time: 2024-07-10 19:30:58.114959-04:00
```

Conclusion

Python's `datetime` module, along with the `pytz` module, provides powerful tools for handling date and time. These modules enable developers to get the current date and time, format them for display, calculate time differences, and work with timezones. These capabilities are essential for applications that require precise time tracking and manipulation. By understanding and utilizing these tools, developers can efficiently manage date and time in their Python programs.

4o