CECS 342 - Lab Assignment 4 - Closure and Coroutine

Due Date: Sunday, March 31st

Team Members: Bryan Tineo & Maxwell Guillermo

Completion of Lab Assignment:

Both team members contributed equally and collaborated throughout the completion of the lab assignment.

Code Decorator.py:

```
# Define a decorator to check authorization before executing a function
from datetime import datetime

def decorator(func):
    """
    A wrapper function that checks if you're allowed to do something before actually
    doing it. If you're not allowed, it stops and tells you so.
    """
    def wrapper():
        # Check if you have permission to perform the action
        time = datetime.now().hour

        if 6 <= time < 18:
            # If not authorized, tell the user they can't proceed
            return func()
        # If authorized, go ahead and perform the action
        return "You are not authorized"
    return wrapper



# Decorate functions with the authorization check
# This means "make sure I'm allowed to do this before doing it"
@decorator
def do_A():
    """Does Task A and tells you it's done."""
    return "Do A"
```

```python
@decorator
def do_B():
    """Does Task B and tells you it's done."""
    return "Do B"

@decorator
def do_C():
    """Does Task C and tells you it's done."""
    return "Do C"

# Main part of the script where we try to do Tasks A, B, and C
if __name__ == "__main__":
    # Try to do Task A and print what happens
    print(do_A())  # Expected: "Do A" if allowed; "You are not authorized" if not
    # Try to do Task B and print what happens
    print(do_B())  # Expected: "Do B" if allowed; "You are not authorized" if not
    # Try to do Task C and print what happens
    print(do_C())  # Expected: "Do C" if allowed; "You are not authorized" if not
```

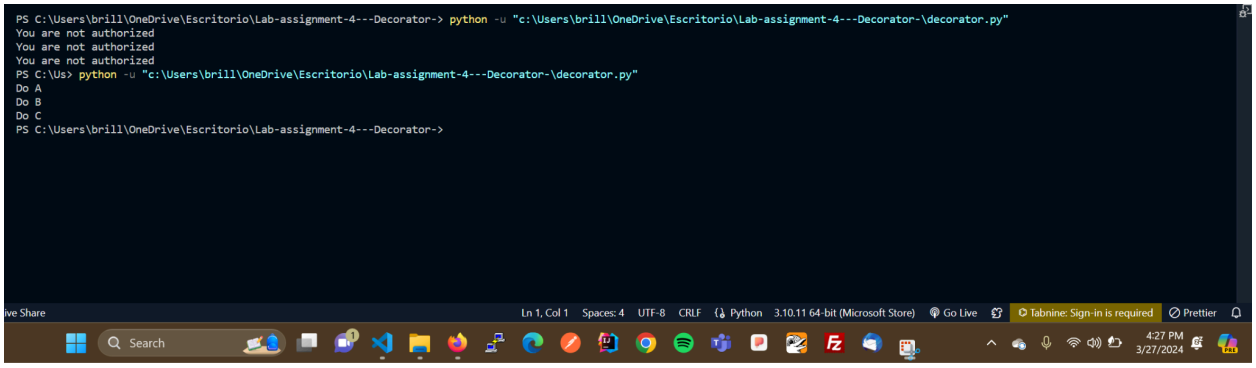
Code Coroutine.py:

```python
import asyncio

# define asynchronous function factorial
async def factorial(name, number):

    # initialize the factorial answer to 1
    ans = 1

    # loop from 1 to number + 1 sice python's for loop excludes the last number
    for i in range(1, number + 1):

        # display taks name, currently factorial value
        # current loop variable

        print(f"Task {name}: Computer factorial({number}), currently i = {i}...")


        ans *= i
```

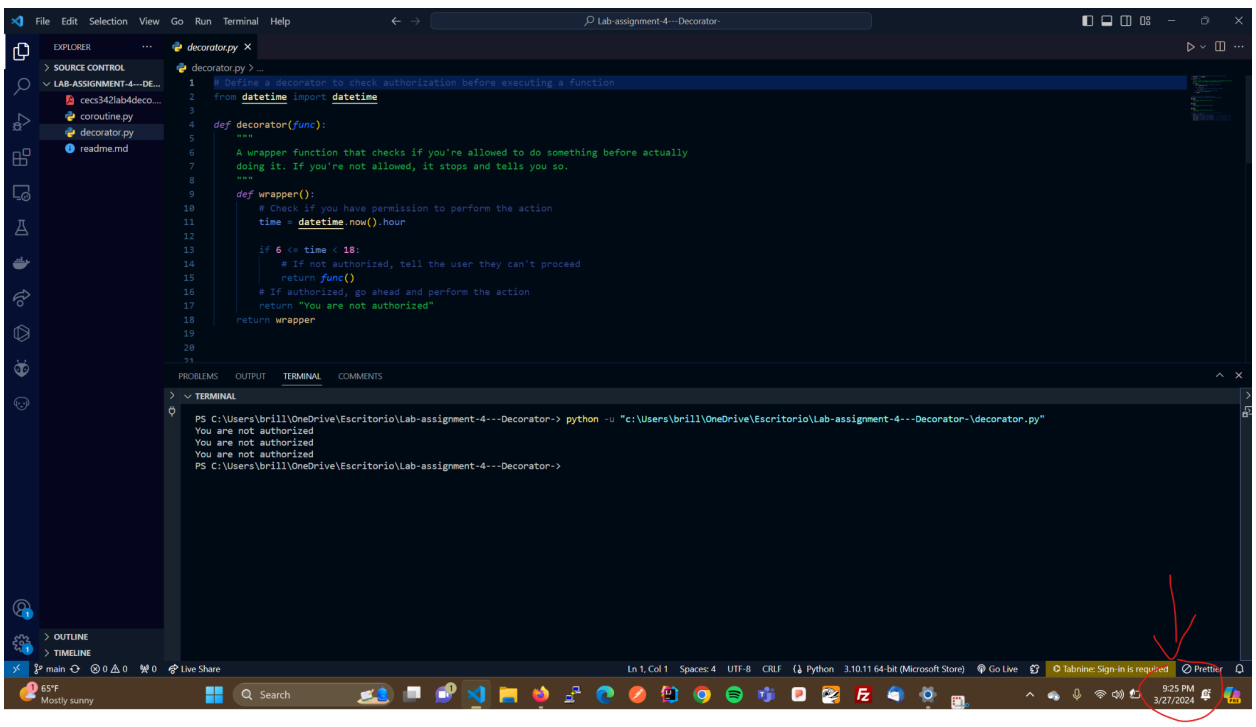
```
        # delay 1 second, this delay will allow other concurrent tasks to run while we
are sleeping
        await asyncio.sleep(1)

    # display the task name and the final factorial value
    print(f"Task {name}: factorial({number}) = {ans}")
    # return the final factorial value
    return ans

async def main():
    # scheduel tbree calls "factoral" concurrently
    # using asyncio.gather will initialize all takss that will return an object for
each task made (task A, Task B, Task C)
    batch = asyncio.gather(
        factorial("A", 3),
        factorial("B",4),
        factorial("C",5)
                            )


    funcA,funcB,funcC = await batch

    print([funcA,funcB,funcC])




if __name__ == "__main__":

    # use asyncio to run the main function
    asyncio.run(main())
```

Output Decorator.py:

decorator.py during daytime:



decorator.py during nighttime:



Output Coroutine.py:

```
PS C:\Users\brill\OneDrive\Escritorio\Lab-assignment-4---Decorator-> python -u "c:\Users\brill\OneDrive\Escritorio\Lab-assignment-4---Decorator-\coroutine.py"
Task A: Computer factorial(3), currently i = 1...
Task B: Computer factorial(4), currently i = 1...
Task C: Computer factorial(5), currently i = 1...
Task A: Computer factorial(3), currently i = 2...
Task B: Computer factorial(4), currently i = 2...
Task C: Computer factorial(5), currently i = 2...
Task A: Computer factorial(3), currently i = 3...
Task B: Computer factorial(4), currently i = 3...
Task C: Computer factorial(5), currently i = 3...
Task A: factorial(3) = 6
Task B: Computer factorial(4), currently i = 4...
Task C: Computer factorial(5), currently i = 4...
Task B: factorial(4) = 24
Task C: Computer factorial(5), currently i = 5...
Task C: factorial(5) = 120
[6, 24, 120]
PS C:\Users\brill\OneDrive\Escritorio\Lab-assignment-4---Decorator->
```