Alejandro Juárez Corona

A01168444

Pruebas de software y aseguramiento de la calidad

Actividad 6.2

Ejercicio de programación 3. Pruebas unitarias

Modulo encargado de definir Hotel, Reservation y Customer:

"""

Module to handle a Hotel Reservation System

"""

# pylint: disable=R0903

class Reservation:

"""Class representing a reservation in a hotel."""

reservation\_counter: int = 1

def \_\_init\_\_(self,

check\_in\_date: str,

check\_out\_date: str,

id\_hotel: int,

id\_customer: int) -> None:

"""Initialize a Reservation object.

Args:

check\_in\_date (str): Check-in date in the format 'YYYY-MM-DD'.

check\_out\_date (str): Check-out date in the format 'YYYY-MM-DD'.

id\_hotel (int): The ID of the hotel.

id\_customer (int): The ID of the customer.

"""

self.reservation\_id: int = Reservation.reservation\_counter

Reservation.reservation\_counter += 1

self.check\_in\_date: str = check\_in\_date

self.check\_out\_date: str = check\_out\_date

self.id\_hotel: int = id\_hotel

self.id\_customer: int = id\_customer

class Customer:

"""Class representing a customer in a hotel."""

customer\_counter: int = 1

def \_\_init\_\_(self, name: str,

age: int,

elite\_status: bool = False,

id\_hotel: int = None) -> None:

"""Initialize a Customer object.

Args:

name (str): The name of the customer.

age (int): The age of the customer.

elite\_status (bool): Whether the customer has elite status.

id\_hotel (int): The ID of the hotel.

"""

self.id\_customer: int = Customer.customer\_counter

Customer.customer\_counter += 1

self.name: str = name

self.age: int = age

self.elite\_status: bool = elite\_status

self.id\_hotel: int = id\_hotel

@classmethod

def create\_customer(cls, name: str,

age: int,

elite\_status: bool = False,

id\_hotel: int = None) -> 'Customer':

"""Create a new Customer instance.

Args:

name (str): The name of the customer.

age (int): The age of the customer.

elite\_status (bool): Whether the customer has elite status.

id\_hotel (int): The ID of the hotel. Defaults to None.

Returns:

Customer: A new Customer instance.

"""

return cls(name, age, elite\_status, id\_hotel)

def delete\_customer(self, customer\_list: list) -> None:

"""Delete the customer from the provided list.

Args:

customer\_list:List of customers to delete the customer.

"""

customer\_list.remove(self)

def display\_info(self) -> None:

"""Display information about the customer."""

print(f"Customer ID: {self.id\_customer}")

print(f"Customer: {self.name}")

print(f"Age: {self.age}")

print(f"Elite Status: {'Yes' if self.elite\_status else 'No'}")

def display\_customer\_info(self) -> None:

"""Display information about the customer."""

self.display\_info()

def update\_elite\_status(self, new\_status: bool) -> None:

"""Update the elite status of the customer.

Args:

new\_status (bool): The new elite status.

"""

self.elite\_status = new\_status

class Hotel:

"""Class representing a hotel."""

def \_\_init\_\_(self, id\_hotel: int,

name: str,

address: str,

capacity: int) -> None:

"""Initialize a Hotel object.

Args:

id\_hotel (int): The ID of the hotel.

name (str): The name of the hotel.

address (str): The address of the hotel.

capacity (int): The capacity of the hotel.

"""

self.id\_hotel: int = id\_hotel

self.name: str = name

self.address: str = address

self.stars: int = 0

self.capacity: int = capacity

self.reservations: list = []

def create\_hotel(self, name: str,

address: str,

stars: int,

capacity: int) -> None:

"""Create or modify a hotel with the provided information.

Args:

name (str): The name of the hotel.

address (str): The address of the hotel.

stars (int): The star rating of the hotel.

capacity (int): The capacity of the hotel.

"""

self.name = name

self.address = address

self.stars = stars

self.capacity = capacity

def delete\_hotel(self) -> None:

"""Delete the hotel."""

del self

def display\_info(self) -> None:

"""Display information about the hotel."""

print(f"Hotel ID: {self.id\_hotel}")

print(f"Hotel: {self.name}")

print(f"Address: {self.address}")

print(f"Stars: {self.stars}")

print(f"Capacity: {self.capacity} guests")

print("Reservations:")

for reservation in self.reservations:

one = f"{reservation.reservation\_id},{reservation.guest\_name}, "

two = f"{reservation.check\_in\_date},{reservation.check\_out\_date}"

print(one + two)

def modify\_info(self, name: str = None,

address: str = None,

stars: int = None,

capacity: int = None) -> None:

"""Modify the information of the hotel.

Args:

name (str): The new name of the hotel. Defaults to None.

address (str): The new address of the hotel. Defaults to None.

stars (int): The new star rating of the hotel. Defaults to None.

capacity (int): The new capacity of the hotel. Defaults to None.

"""

if name:

self.name = name

if address:

self.address = address

if stars:

self.stars = stars

if capacity:

self.capacity = capacity

def reserve\_room(self, check\_in\_date: str, check\_out\_date: str) -> int:

"""Reserve a room in the hotel.

Args:

check\_in\_date (str): Check-in date in the format 'YYYY-MM-DD'.

check\_out\_date (str): Check-out date in the format 'YYYY-MM-DD'.

Returns:

int: The ID of the reservation.

"""

reservation = Reservation(check\_in\_date,

check\_out\_date,

self.id\_hotel,

len(self.reservations) + 1)

self.reservations.append(reservation)

return reservation.reservation\_id

def cancel\_reservation(self, reservation\_id: int) -> bool:

"""Cancel a reservation in the hotel.

Args:

reservation\_id (int): The ID of the reservation to be canceled.

Returns:

bool: True if the reservation is canceled, False otherwise.

"""

for reservation in self.reservations:

if reservation.reservation\_id == reservation\_id:

self.reservations.remove(reservation)

return True

return False

Módulo encargado de las pruebas unitarias

import unittest

from datetime import date

from hotel\_reservation\_module import Hotel, Customer, Reservation

class TestHotelReservationSystem(unittest.TestCase):

def setUp(self):

# Create instances of the necessary classes for the tests

self.hotel = Hotel(1, "Example Hotel", "123 Main St", 100)

self.customer = Customer.create\_customer("John Doe", 30)

self.reservation = Reservation("2024-03-01", "2024-03-05", 1, 1)

def test\_reserve\_room(self):

# Ensure a room can be reserved successfully

reservation\_id = self.hotel.reserve\_room("2024-03-01", "2024-03-05")

self.assertEqual(len(self.hotel.reservations), 1)

def test\_cancel\_reservation(self):

# Ensure a reservation can be canceled successfully

self.hotel.reservations.append(self.reservation)

result = self.hotel.cancel\_reservation(self.reservation.reservation\_id)

self.assertTrue(result)

self.assertEqual(len(self.hotel.reservations), 0)

def test\_create\_customer(self):

# Ensure a customer can be created successfully

self.assertEqual(self.customer.name, "John Doe")

self.assertEqual(self.customer.age, 30)

def test\_update\_elite\_status(self):

# Ensure the customer's elite status can be updated successfully

self.customer.update\_elite\_status(True)

self.assertTrue(self.customer.elite\_status)

def test\_display\_info(self):

# Ensure information can be displayed correctly

self.customer.display\_info()

def test\_display\_customer\_info(self):

# Ensure customer information can be displayed correctly

self.customer.display\_customer\_info()

def test\_modify\_hotel\_info(self):

# Ensure hotel information can be modified correctly

self.hotel.modify\_info(name="New Name", stars=5)

self.assertEqual(self.hotel.name, "New Name")

self.assertEqual(self.hotel.stars, 5)

def test\_delete\_customer(self):

# Ensure a customer can be deleted correctly from the list of customers

customer\_list = [self.customer]

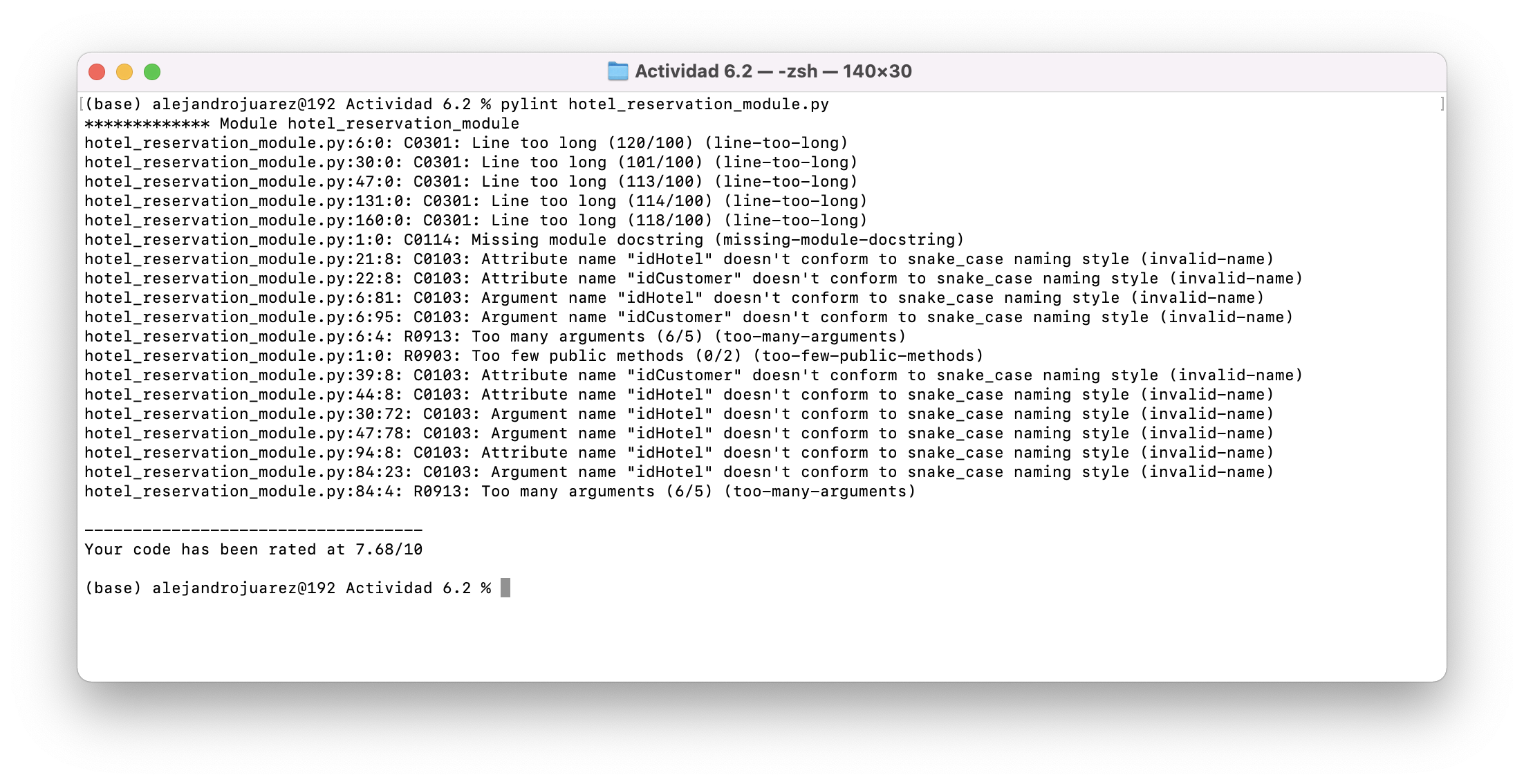
self.customer.delete\_customer(customer\_list)

self.assertEqual(len(customer\_list), 0)

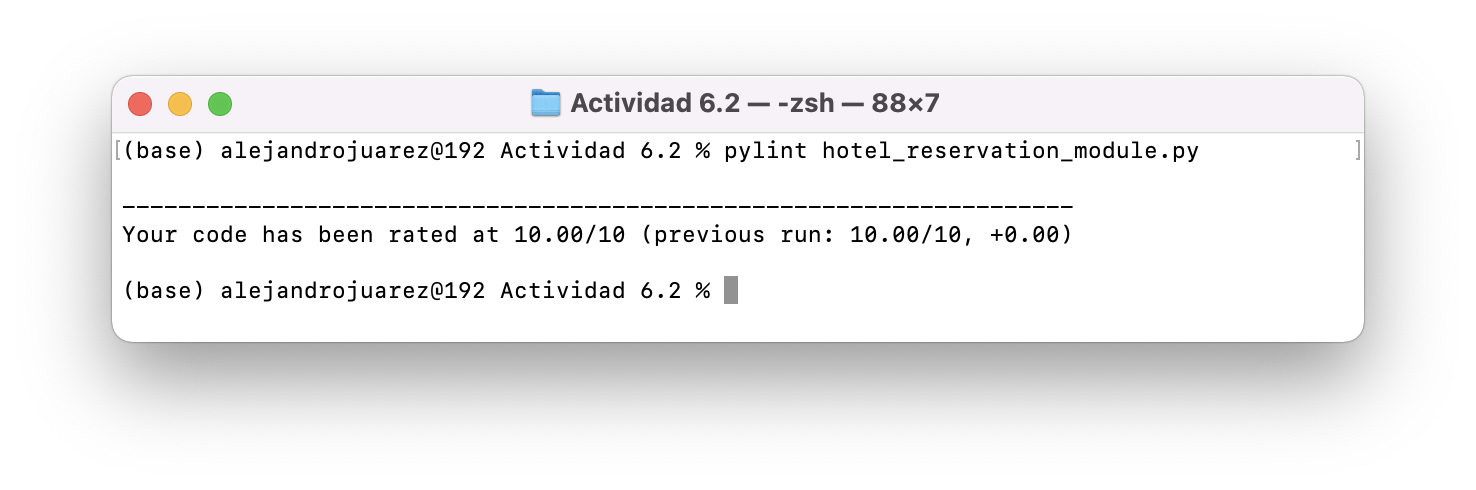
if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

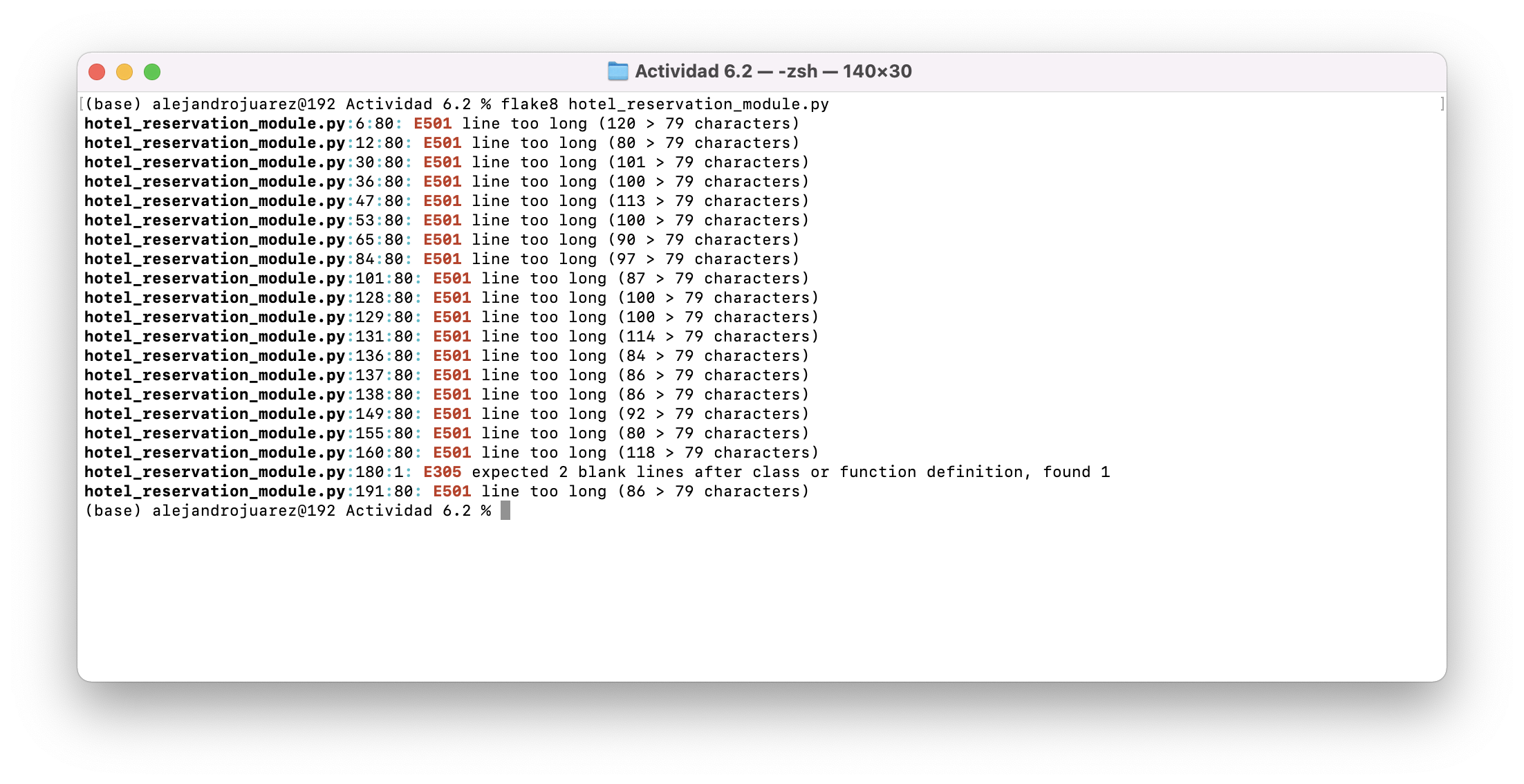
Chequeo con Pylint



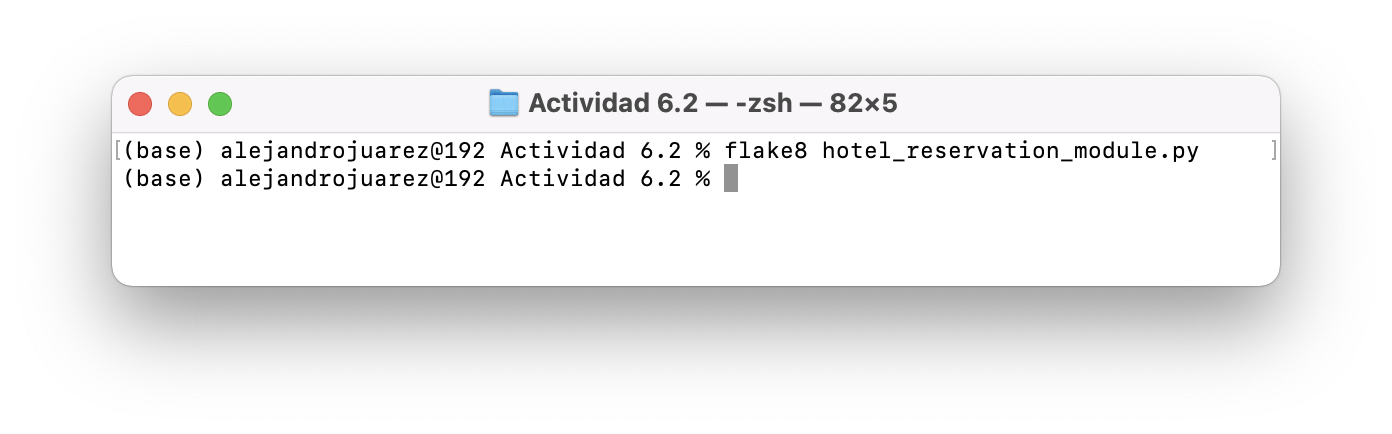
Corrección de errores de Pylint



Chequeo de errores con Flake8



Corrección de errores con Flake8



Ejecución de Pruebas Unitarias

