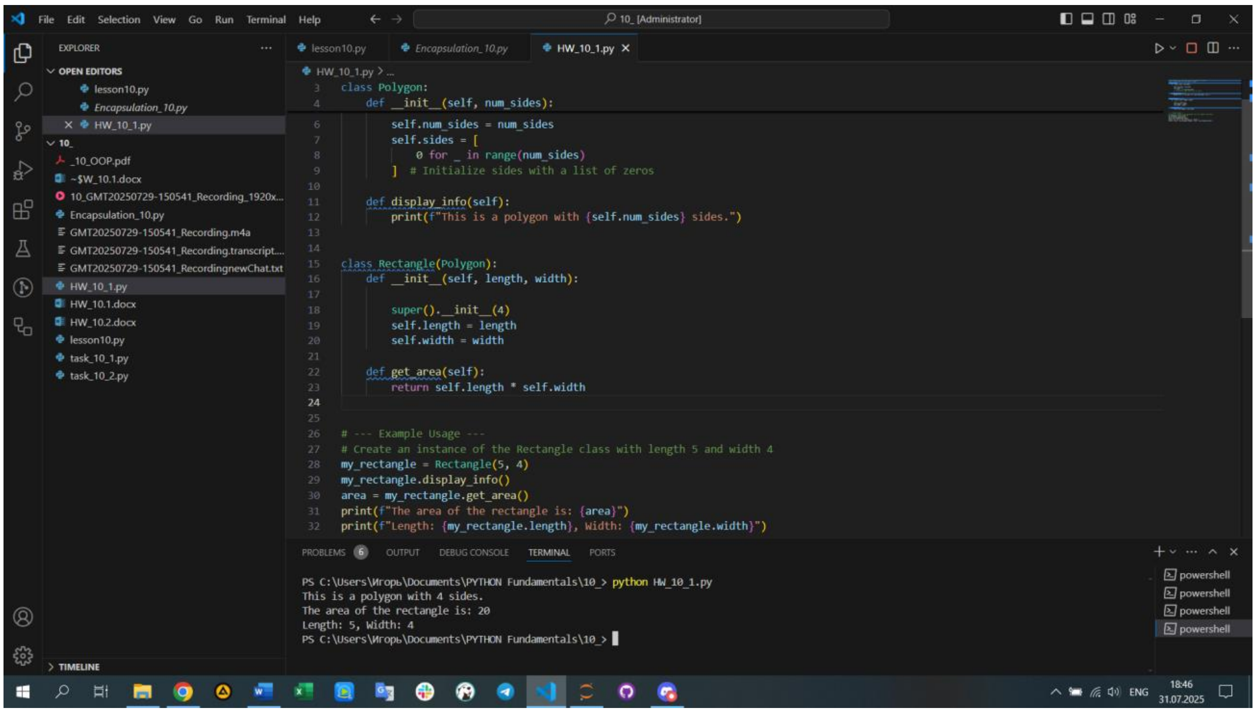
HW_10.1

Task 1. Create a polygon class and a rectangle class that inherits from the polygon class and finds the square of rectangle.

```python
# The Polygon class is a base class representing a polygon.
# It's initialized with the number of sides.
class Polygon:
    def __init__(self, num_sides):
        """
        Initializes a Polygon object.

        Args:
            num_sides (int): The number of sides of the polygon.
        """
        self.num_sides = num_sides
        self.sides = [0 for _ in range(num_sides)] # Initialize sides with a list
of zeros

    def display_info(self):
        """Prints the number of sides of the polygon."""
        print(f"This is a polygon with {self.num_sides} sides.")

# The Rectangle class inherits from the Polygon class.
# This means a Rectangle "is-a" Polygon with 4 sides.
class Rectangle(Polygon):
    def __init__(self, length, width):
        """
        Initializes a Rectangle object.

        Args:
            length (float): The length of the rectangle.
            width (float): The width of the rectangle.
        """
        # Call the constructor of the base class (Polygon)
        # We pass 4 because a rectangle always has 4 sides.
        super().__init__(4)
        self.length = length
        self.width = width

    def get_area(self):
        """
        Calculates and returns the area of the rectangle.

        Returns:
            float: The area of the rectangle.
        """
        return self.length * self.width
```

```python
# --- Example Usage ---

# Create an instance of the Rectangle class with length 5 and width 4
my_rectangle = Rectangle(5, 4)

# Use the inherited method from the Polygon class
my_rectangle.display_info()

# Calculate and print the area using the get_area method
area = my_rectangle.get_area()
print(f"The area of the rectangle is: {area}")

# You can also access the attributes directly
print(f"Length: {my_rectangle.length}, Width: {my_rectangle.width}")
```



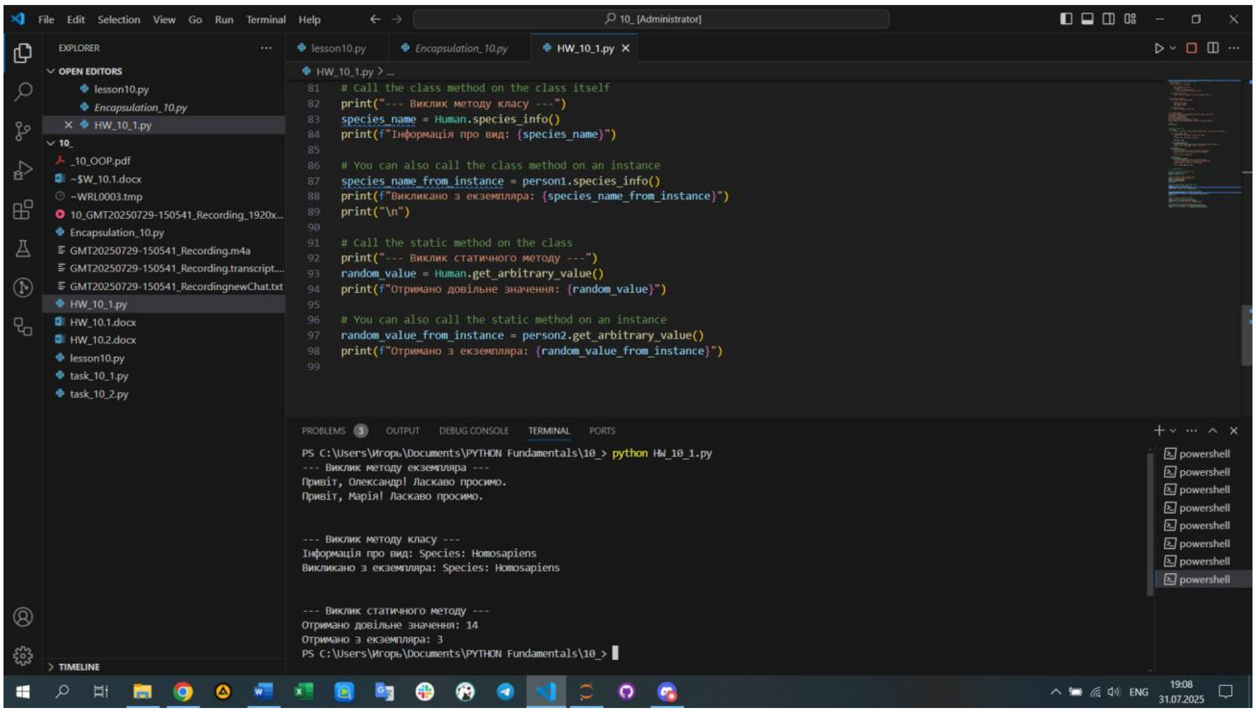
```python
class Polygon:
    def __init__(self, num_sides):
        self.num_sides = num_sides
        self.sides = [
            0 for _ in range(num_sides)
        ]  # Initialize sides with a list of zeros

    def display_info(self):
        print(f"This is a polygon with {self.num_sides} sides.")


class Rectangle(Polygon):
    def __init__(self, length, width):

        super().__init__(4)
        self.length = length
        self.width = width

    def get_area(self):
        return self.length * self.width


# --- Example Usage ---
# Create an instance of the Rectangle class with length 5 and width 4
my_rectangle = Rectangle(5, 4)
my_rectangle.display_info()
area = my_rectangle.get_area()
print(f"The area of the rectangle is: {area}")
print(f"Length: {my_rectangle.length}, Width: {my_rectangle.width}")
```

```
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_> python HW_10_1.py
This is a polygon with 4 sides.
The area of the rectangle is: 20
Length: 5, Width: 4
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_>
```

Task 2. Create a class Human, everyone has a name, create a method in the class that displays a welcome message to each person. Create a class method in the class that returns information that it is a species of “Homosapiens”. And in the class create a static method that returns an arbitrary method.

```python
import random

# Define the Human class
class Human:
    """
    A class to represent a human, demonstrating instance, class, and static 
methods.
    """

    def __init__(self, name):
        """
        Initializes a new Human instance with a name.

        Args:
            name (str): The name of the human.
        """
        self.name = name

    def welcome_message(self):
        """
        An instance method that displays a personalized welcome message.
        """
        print(f"Привіт, {self.name}! Ласкаво просимо.")

    @classmethod
    def species_info(cls):
        """
        A class method that returns the species information.
        It can be called on the class itself or an instance.
        """
        return "Species: Homosapiens"

    @staticmethod
    def get_arbitrary_value():
        """
        A static method that returns an arbitrary value.
        It doesn't require access to the class or instance state.
        In this example, it returns a random number.
        """
        return random.randint(1, 100)

# --- Example Usage ---

# Create an instance of the Human class
person1 = Human("Олександр")
person2 = Human("Марія")

# Call the instance method on an object
print("--- Виклик методу екземпляра ---")
person1.welcome_message()
person2.welcome_message()
print("\n")

# Call the class method on the class itself
print("--- Виклик методу класу ---")
species_name = Human.species_info()
```

```python
print(f"Інформація про вид: {species_name}")

# You can also call the class method on an instance
species_name_from_instance = person1.species_info()
print(f"Викликано з екземпляра: {species_name_from_instance}")
print("\n")

# Call the static method on the class
print("--- Виклик статичного методу ---")
random_value = Human.get_arbitrary_value()
print(f"Отримано довільне значення: {random_value}")

# You can also call the static method on an instance
random_value_from_instance = person2.get_arbitrary_value()
print(f"Отримано з екземпляра: {random_value_from_instance}")
```



```python
# Call the class method on the class itself
print("--- Виклик методу класу ---")
species_name = Human.species_info()
print(f"Інформація про вид: {species_name}")

# You can also call the class method on an instance
species_name_from_instance = person1.species_info()
print(f"Викликано з екземпляра: {species_name_from_instance}")
print("\n")

# Call the static method on the class
print("--- Виклик статичного методу ---")
random_value = Human.get_arbitrary_value()
print(f"Отримано довільне значення: {random_value}")

# You can also call the static method on an instance
random_value_from_instance = person2.get_arbitrary_value()
print(f"Отримано з екземпляра: {random_value_from_instance}")
```

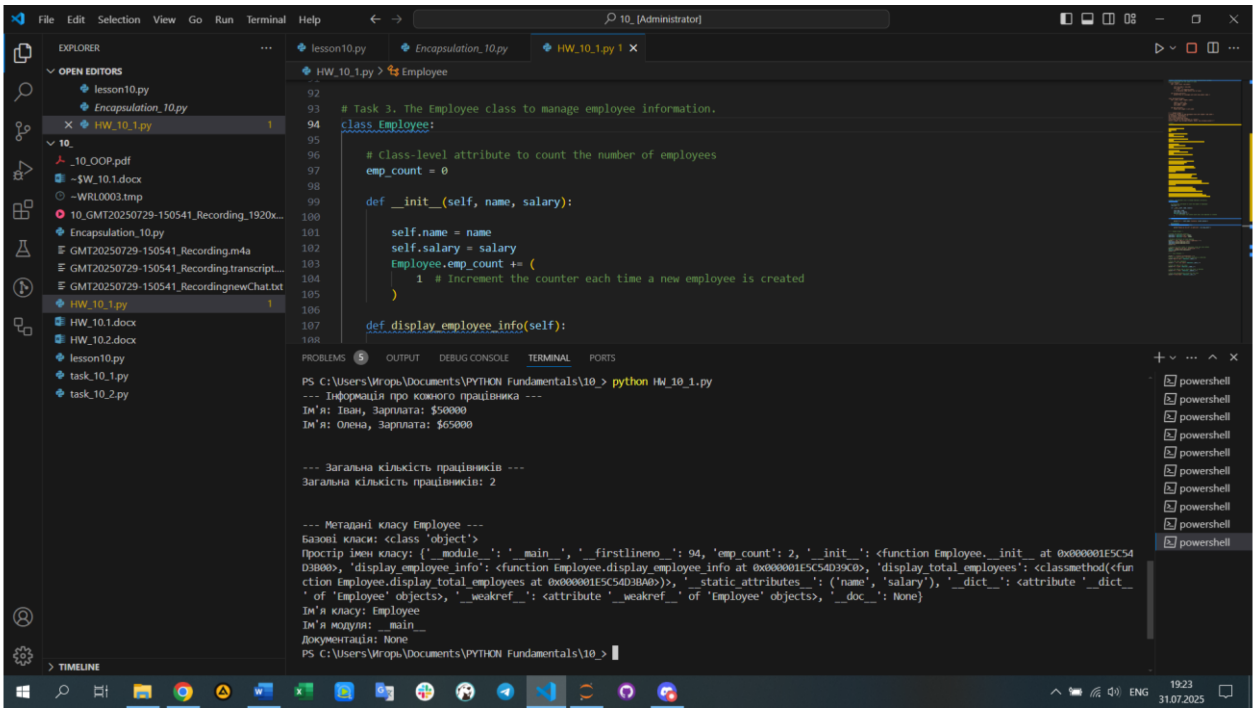
```
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_> python HW_10_1.py
--- Виклик методу екземпляра ---
Привіт, Олександр! Ласкаво просимо.
Привіт, Марія! Ласкаво просимо.


--- Виклик методу класу ---
Інформація про вид: Species: Homosapiens
Викликано з екземпляра: Species: Homosapiens


--- Виклик статичного методу ---
Отримано довільне значення: 14
Отримано з екземпляра: 3
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_>
```

Task 3. Create an employee class. Each employee has characteristics such as name and salary. The class should have a counter that calculates the total number of employees, as well as method that prints the total number of employees, as well as a method that prints the total number of employees and a method that displays information about each employee in particular, namely the name and salary.

In addition to creating a class, display information about the base classes from which the employee class is inherited (__base__), the class namespace (__dict__), the class name (__name__), the module name in which the class is defined (__module__), the documentation bar (__doc__)

```python
# The Employee class to manage employee information.
class Employee:
    """
    Класс Employee для зберігання інформації про працівника.
    Він містить лічильник для загальної кількості працівників, а також
    методи для відображення деталей про працівника.
    """

    # Class-level attribute to count the number of employees
    emp_count = 0

    def __init__(self, name, salary):
        """
        Initializes a new Employee instance.

        Args:
            name (str): The name of the employee.
            salary (float): The salary of the employee.
        """
        self.name = name
        self.salary = salary
        Employee.emp_count += 1  # Increment the counter each time a new employee is created

    def display_employee_info(self):
        """
        Displays the name and salary for a specific employee.
        """
        print(f"Ім'я: {self.name}, Зарплата: ${self.salary}")

    @classmethod
    def display_total_employees(cls):
        """
        A class method to display the total number of employees.
        It can be called on the class itself or an instance.
        """
        print(f"Загальна кількість працівників: {cls.emp_count}")

# --- Example Usage ---

# Create two Employee instances
employee1 = Employee("Іван", 50000)
employee2 = Employee("Олена", 65000)

# Display information for each employee
print("--- Інформація про кожного працівника ---")
```

```python
employee1.display_employee_info()
employee2.display_employee_info()
print("\n")

# Display the total number of employees using the class method
print("--- Загальна кількість працівників ---")
Employee.display_total_employees()
print("\n")

# --- Class Metadata ---

print("--- Метадані класу Employee ---")
# Print the base classes from which Employee is inherited
print(f"Базові класи: {Employee.__base__}")

# Print the class namespace
print(f"Простір імен класу: {Employee.__dict__}")

# Print the name of the class
print(f"Ім'я класу: {Employee.__name__}")

# Print the module name in which the class is defined
print(f"Ім'я модуля: {Employee.__module__}")

# Print the documentation string for the class
print(f"Документація: {Employee.__doc__}")
```



```python
# Task 3. The Employee class to manage employee information.
class Employee:

    # Class-level attribute to count the number of employees
    emp_count = 0

    def __init__(self, name, salary):

        self.name = name
        self.salary = salary
        Employee.emp_count += (
            1  # Increment the counter each time a new employee is created
        )

    def display_employee_info(self):
```

```
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_> python HW_10_1.py
--- Інформація про кожного працівника ---
Ім'я: Іван, Зарплата: $50000
Ім'я: Олена, Зарплата: $65000


--- Загальна кількість працівників ---
Загальна кількість працівників: 2


--- Метадані класу Employee ---
Базові класи: <class 'object'>
Простір імен класу: {'__module__': '__main__', '__firstlineno__': 94, 'emp_count': 2, '__init__': <function Employee.__init__ at 0x000001E5C54
D3B00>, 'display_employee_info': <function Employee.display_employee_info at 0x000001E5C54D39C0>, 'display_total_employees': <classmethod(<fun
ction Employee.display_total_employees at 0x000001E5C54D3BA0>)>, '__static_attributes__': ('name', 'salary'), '__dict__': <attribute '__dict__
' of 'Employee' objects>, '__weakref__': <attribute '__weakref__' of 'Employee' objects>, '__doc__': None}
Ім'я класу: Employee
Ім'я модуля: __main__
Документація: None
PS C:\Users\Игорь\Documents\PYTHON Fundamentals\10_>
```