Started on	Monday, 2 June 2025, 9:12 AM
State	Finished
Completed on	Monday, 2 June 2025, 9:58 AM
Time taken	45 mins 25 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program for simply using the overloading operator for adding two objects.

For example:

Input	Result
23 21 hello world	: 44 : helloworld

Answer: (penalty regime: 0 %)

```python
class Adder:
    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return Adder(self.value + other.value)

    def __str__(self):
        return str(self.value)


# Input and result demonstration
if __name__ == "__main__":
    # Example 1: Adding integers
    a = Adder(23)
    b = Adder(21)
    result1 = a + b
    print(": ", result1)  # Output: : 44

    # Example 2: Adding strings
    c = Adder("hello")
    d = Adder("world")
```

	Input	Expected	Got	
✓	23 21 hello world	: 44 : helloworld	: 44 : helloworld	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Create an abstract base class has a concrete method sleep() that will be the same for all the child classes. So, we do not define it as an abstract method, thus saving us from code repetition. On the other hand, the sounds that animals make are all different. For that purpose, define the sound() method as an abstract method. then implement it in all child classes.

For example:

Result
I am going to sleep in a while I can meow I can hiss

Answer: (penalty regime: 0 %)

Reset answer

```python
from abc import ABC, abstractmethod

class Animal(ABC):
    def sleep(self):
        print("I am going to sleep in a while")

    @abstractmethod
    def sound(self):
        pass


class Cat(Animal):
    def sound(self):
        print("I can meow")


class Snake(Animal):
    def sound(self):
        print("I can hiss")


# Testing the classes
```

	Expected	Got	

✓	`I am going to sleep in a while`
	`I can meow`
	`I can hiss`
		`I am going to sleep in a while`
		`I can meow`
		`I can hiss`
			✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Create two new classes: Lion and Giraffe The outputs from this program are carnivore and herbivore, respectively. The two classes both use the method name diet, but they define those methods differently. An object instantiated from the Lion class will use the method as it is defined in that class. The Giraffe class may have a method with the same name, but objects instantiated from the Lion class won't interact with it.

For example:

Result
carnivore herbivore

Answer: (penalty regime: 0 %)

Reset answer

```
class Lion:
    def diet(self):
        print("carnivore")

class Giraffe:
    def diet(self):
        print("herbivore")


# Creating objects and calling the diet method
if __name__ == "__main__":
    lion = Lion()
    giraffe = Giraffe()

    lion.diet()     # Output: carnivore
    giraffe.diet()  # Output: herbivore

```

	Expected	Got	
✓	carnivore herbivore	carnivore herbivore	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Incorrect

Mark 0.00 out of 20.00

Write a python program to get the account balance of the user and print whether the user have to pay penalty for zero balance.

For example:

Input	Result
`-500`	`Balance is below 0, add funds now or you will be charged a penalty.`

Answer: (penalty regime: 0 %)

```python
# Get account balance from the user
balance = float(input())

# Check balance and print message
if balance <= 0:
    print("Balance is below 0, add funds now or you will be charged a penalty.")
else:
    print("Your account is :")

```

	Input	Expected	Got	
✓	`-500`	`Balance is below 0, add funds now or you will be charged a penalty.`	`Balance is below 0, add funds now or you will be charged a penalty.`	✓

	Input	Expected	Got	
✖	1000	Your balance is: 1000	Your account is :	✖

Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Create `Counter` class which has one attribute called `current` which defaults to zero. And it has three methods:

- `increment()` increases the value of the `current` attribute by one.
- `value()` returns the current value of the `current` attribute
- `reset()` sets the value of the `current` attribute to zero

create a new instance of the `Counter` class and calls the `increment()` method three times before showing the current value of the counter to the screen

For example:

Result
3

Answer: (penalty regime: 0 %)

Reset answer

```python
class Counter:
    def __init__(self):
        self.current = 0

    def increment(self):
        self.current += 1

    def value(self):
        return self.current

    def reset(self):
        self.current = 0


# Create an instance and test the functionality
if __name__ == "__main__":
    counter = Counter()
    counter.increment()
    counter.increment()
    counter.increment()
    print(counter.value())  # Output: 3

```

	Expected	Got	
✓	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.