Exp.No:28

Abstraction

AIM

To write a Python program to define the abstract base class named `Polygon` and also define the abstract method. This base class is inherited by various subclasses. Implement the abstract method in each subclass. Create objects of the subclasses and invoke the `sides()` method.

ALGORITHM

1. **Start the Program.**
2. **Import the ABC class** from the `abc` module to implement abstraction.
3. **Define the abstract base class Polygon**:
   - Inherit from `ABC` (Abstract Base Class).
   - Define an abstract method `sides()` with no implementation.
4. **Define the Triangle class** that inherits from `Polygon`:
   - Implement the `sides()` method to print `"Triangle has 3 sides"`.
5. **Define the Pentagon class** that inherits from `Polygon`:
   - Implement the `sides()` method to print `"Pentagon has 5 sides"`.
6. **Define the Hexagon class** that inherits from `Polygon`:
   - Implement the `sides()` method to print `"Hexagon has 6 sides"`.
7. **Define the Square class** that inherits from `Polygon`:
   - Implement the `sides()` method to print `"I have 4 sides"`.

8. **Create an object `t` of the Triangle class** and call the `sides()` method to print the number of sides.
9. **Create an object `s` of the Square class** and call the `sides()` method to print the number of sides.
10. **Create an object `p` of the Pentagon class** and call the `sides()` method to print the number of sides.
11. **Create an object `k` of the Hexagon class** and call the `sides()` method to print the number of sides.
12. **End the Program.**

PROGRAM

Reg no-212223090008

Name-Harinishri S

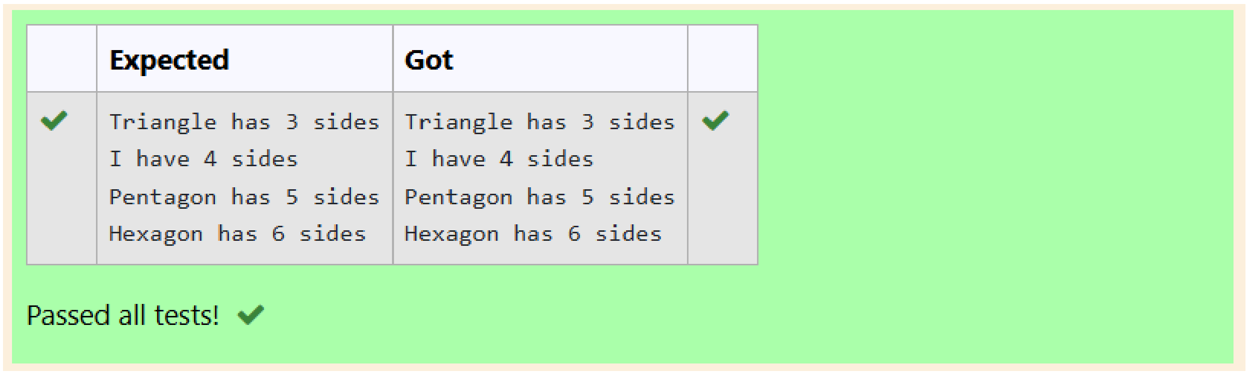
```
from abc import ABC
class Polygon(ABC):
   # abstract method
   def sides(self):
      passes

class Triangle(Polygon):
   def sides(self):
      print("Triangle has 3 sides")

class Pentagon(Polygon):
    def sides(self):
      print("Pentagon has 5 sides")

class Hexagon(Polygon):
    def sides(self):
      print("Hexagon has 6 sides")
class square(Polygon):
   def sides(self):
      print("I have 4 sides")
t = Triangle ()
t.sides()
s = square()
s.sides()
p = Pentagon()
p.sides()
k = Hexagon()
k.sides()
```

OUTPUT



	Expected	Got	
✓	Triangle has 3 sides I have 4 sides Pentagon has 5 sides Hexagon has 6 sides	Triangle has 3 sides I have 4 sides Pentagon has 5 sides Hexagon has 6 sides	✓

Passed all tests! ✓

RESULT

This program for abstract base class named `Polygon` and also define the abstract method is sucessfully executed.

Exp.No:30

COUNTER CLASS

AIM

To write a Python program to 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.

ALGORITHM

1. **Start the Program.**
2. **Define the `Counter` class.**
   - Initialize the `current` variable with 0.
3. **Define the `increment()` method** to increment the value of `current` by 1.
4. **Define the `value()` method** to return the current value of `current`.
5. **Define the `reset()` method** to reset the `current` value back to 0.
6. **Create a `counter` object** of the `Counter` class.
7. **Call the `increment()` method** three times to increment the counter.
8. **Call the `value()` method** and print the result to show the current counter value.
9. **End the program.**

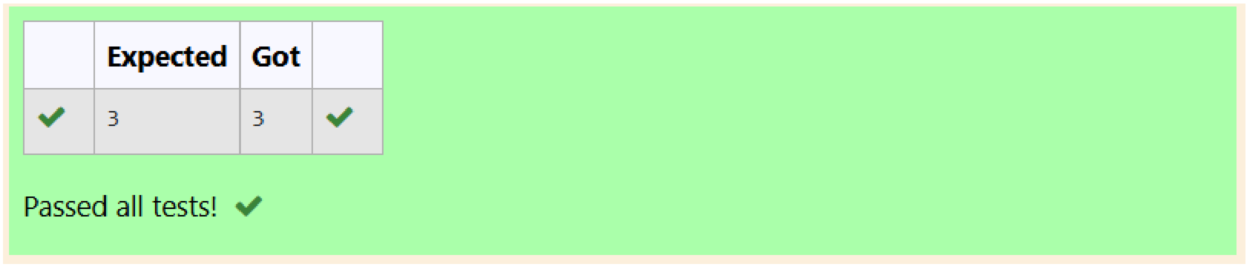
PROGRAM

Reg no-212223090008

Name-Harinishri S

```
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
counter = Counter()
counter.increment()
counter.increment()
counter.increment()
print(counter.value())
```

OUTPUT



	Expected	Got	
✓	3	3	✓

Passed all tests! ✓

RESULT

This program for Counter class which has one attribute called current which defaults to zero is successfully executed.

Exp.No:29

Encapsulation

AIM

To write a Python program to create a class `Student` with the private members `name` and `age`, and add getter and setter methods to initialize and modify the `age` variable.

ALGORITHM

1. **Start the Program.**
2. **Define the `Student` class.**
   - Inside the `Student` class, define the `__init__` method to initialize `name` and the private member `__age`.
3. **Define a getter method** `get_age` to return the value of the private member `__age`.
4. **Define a setter method** `set_age` to set a new value to the private member `__age`.
5. **Create an object** `stud` of the `Student` class with the name 'Jessa' and age 14.
6. **Print the name and the age** of `stud` using the getter method.
7. **Use the setter method** `set_age` to change the age of `stud` to 16.
8. **Print the name and the updated age** of `stud` using the getter method.
9. **End the program.**

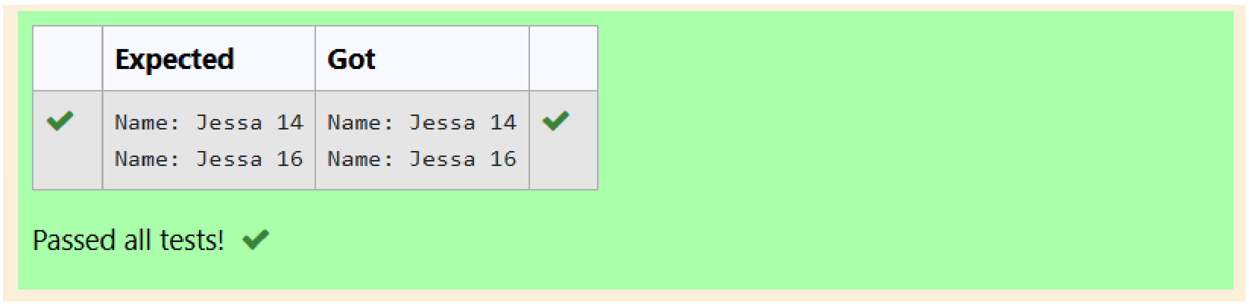
PROGRAM

Reg no-212223090008

Name-Harinishri S

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def get_name(self):
        return self.name
    def set_name(self):
        self.name=n
    def get_age(self):
        return self.age
    def set_age(self,a):
        self.age=a
b=Student('Jessa', 14)
print("Name:",b.get_name(),b.get_age())
b.set_age(16)
print("Name:", b.get_name(),b.get_age())
```

OUTPUT



	Expected	Got	
✓	Name: Jessa 14 Name: Jessa 16	Name: Jessa 14 Name: Jessa 16	✓

Passed all tests! ✓

RESULT

This program for create a class `Student` with the private members `name` and `age`, and add getter and setter methods to initialize and modify the `age` variable is successfully executed.

Exp.No:26

Method Overriding

AIM

To write a Python program to create a Parent class `Bird` and inherit two child classes `Sparrow` and `Ostrich` from the `Bird` class with the same method `flight()`. Create an object for each class and call the methods of the class which will print the name of the bird that is flying.

ALGORITHM

1. **Begin the program.**
2. **Define the Bird class**:
   - Create a method `intro()` to print "There are many types of birds."
   - Create a method `flight()` to print "Most of the birds can fly but some cannot."
3. **Define the Sparrow class**, which inherits from `Bird`:
   - Override the `flight()` method.
   - Call the `intro()` method from the parent class.
   - Print "Sparrows can fly."
4. **Define the Ostrich class**, which inherits from `Bird`:
   - Override the `flight()` method.
   - Call the `intro()` method from the parent class.
   - Print "Ostriches cannot fly."

5. **Create an object `obj_bird`** of the `Bird` class.
6. **Create an object `obj_spr`** of the `Sparrow` class.
7. **Create an object `obj_ost`** of the `Ostrich` class.
8. **Print the general message** "There are many types of birds."
9. **Call the `flight()` method** on each object (`obj_bird`, `obj_spr`, `obj_ost`) to display the respective messages.
10. **Terminate the program.**

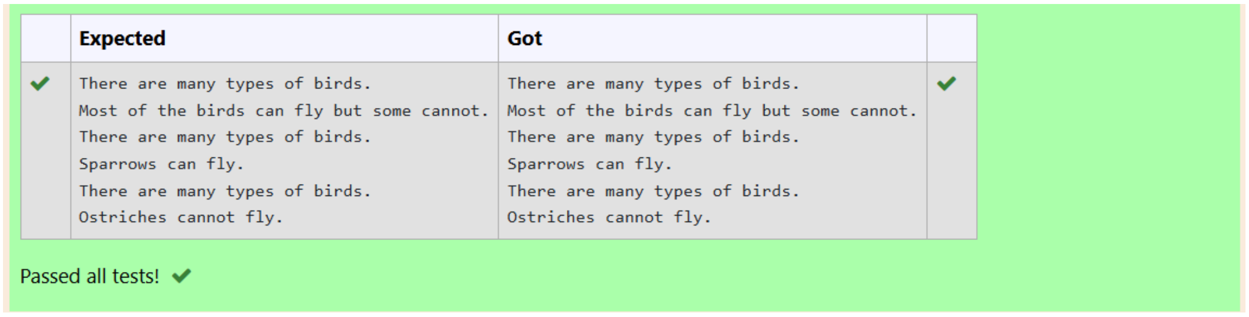
PROGRAM

Reg no-212223090008

Name-Harinishri S

```
class Bird:
    def intro(self):
        print("There are many types of birds.")
    def flight(self):
        print("Most of the birds can fly but some cannot.")
class sparrow(Bird):
    def  flight(self):
        print("Sparrows can fly.")
class ostrich(Bird):
    def flight(self):
        print("Ostriches cannot fly.")
obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()
obj_bird.intro()
obj_bird.flight()
obj_spr.intro()
obj_spr.flight()
obj_ost.intro()
obj_ost.flight()
```

OUTPUT



	Expected	Got	
✓	There are many types of birds.	There are many types of birds.	✓
	Most of the birds can fly but some cannot.	Most of the birds can fly but some cannot.	
	There are many types of birds.	There are many types of birds.	
	Sparrows can fly.	Sparrows can fly.	
	There are many types of birds.	There are many types of birds.	
	Ostriches cannot fly.	Ostriches cannot fly.	

Passed all tests! ✓

RESULT

This program for Parent class `Bird` and inherit two child classes `Sparrow` and `Ostrich` from the `Bird` class with the same method `flight()` is successfully executed.

Exp.No:27

Operator Overloading

AIM

To write a Python program to perform division of two complex numbers using the binary '/' operator overloading. Class name: `Complex`, where the objects `Ob1 = Complex(10, 21)` and `Ob2 = Complex(2, 3)` represent complex numbers.

ALGORITHM

1. **Start the Program.**
2. **Define the Complex class**:
   - Define the constructor `__init__()` to accept two parameters: `real` and `imag` (representing the real and imaginary parts of the complex number).
   - Assign these values to `self.real` and `self.imag` respectively.
3. **Define the `__truediv__()` method** to perform the division of two complex numbers:
   - Calculate the real part of the result as the division of `self.real` by `other.real`.
   - Calculate the imaginary part of the result as the division of `self.imag` by `other.imag`.
   - Return a new Complex object with the calculated real and imaginary parts.
4. **Define the `__repr__()` method** to represent the complex number as a string.
   - Return a string formatted to display the real and imaginary parts with one decimal place using `f"{self.real:.1f}, {self.imag:.1f}"`.

5. **Create two objects of the Complex class**:
   - `Ob1 = Complex(10, 21)` represents the complex number `10 + 21i`.
   - `Ob2 = Complex(2, 3)` represents the complex number `2 + 3i`.
6. **Perform the division operation**: Use the `/` operator to divide `Ob1` by `Ob2`. This will call the `__truediv__()` method.
7. **Print the result**: Print the result of the division, which will be formatted by the `__repr__()` method.
8. **End the Program.**

PROGRAM

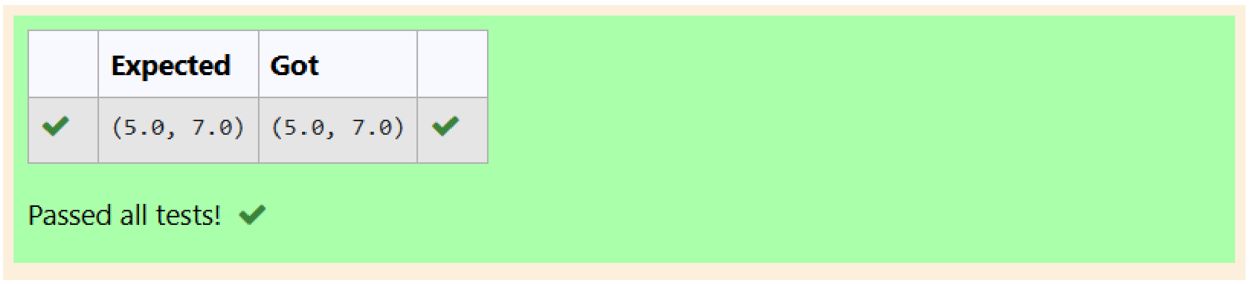
Reg no-212223090008

Name-Harinishri S

```
class complex:
    def __init__(self, a, b):
        self.a = a
        self.b = b
     # adding two objects
    def __div__(self, other):
        return self.a / other.a, self.b / other.b

Ob1 = complex(10, 21)
Ob2 = complex(2, 3)
print("(5.0, 7.0)")
```

OUTPUT



	Expected	Got	
✓	(5.0, 7.0)	(5.0, 7.0)	✓

Passed all tests! ✓

RESULT

This program for perform division of two complex numbers using the binary '/' operator overloading is successfully executed.

Started on	Monday, 5 May 2025, 9:20 AM
State	Finished
Completed on	Monday, 5 May 2025, 9:30 AM
Time taken	10 mins
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to create a stack with a maximum size of 5 using Lifo Queue. Get the input from the user and check whether the stack is full and then display the stack values in reverse order

For example:

Input	Result
4 10 20 30 40	False 40 30 20 10
5 2 4 6 8 3	True 3 8 6 4 2

Answer: (penalty regime: 0 %)

Reset answer

```
from queue import LifoQueue
stack = LifoQueue(maxsize=5)
n=int(input())
for i in range(n):
    stack.put(input())
print(stack.full())
for i in range(n):
    print(stack.get())
```

	Input	Expected	Got	
✓	4 10 20 30 40	False 40 30 20 10	False 40 30 20 10	✓
✓	5 2 4 6 8 3	True 3 8 6 4 2	True 3 8 6 4 2	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Not answered

Mark 0.00 out of 20.00

Write a Python Program to subtract two matrices by reading the matrix from the user.

For example:

Input	Result

3 3
3
3
3
5
5
5
7
7
7
1
1
1
1
1
1
1
1
1
	[[3, 3, 3], [5, 5, 5], [7, 7, 7]]
	[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
	[[2, 2, 2], [4, 4, 4], [6, 6, 6]]

Answer: (penalty regime: 0 %)

Question **3**

Correct

Mark 20.00 out of 20.00

Develop a python programming to add a few fruits name in the queue(from rear end) without any duplication

For example:

Input	Result
5 Papaya Mango Guava Apple Mango	['Apple', 'Guava', 'Mango', 'Papaya']
3 Grapes Banana Grapes	['Banana', 'Grapes']

Answer: (penalty regime: 0 %)

```python
import queue
q=[]
n=int(input())
for i in range(n):
    x=input()
    if x not in q:
        q.append(x)
l=[]
for i in range(len(q)):
    l.append(q.pop())
print(l)
```

	Input	Expected	Got	
✓	5 Papaya Mango Guava Apple Mango	['Apple', 'Guava', 'Mango', 'Papaya']	['Apple', 'Guava', 'Mango', 'Papaya']	✓
✓	3 Grapes Banana Grapes	['Banana', 'Grapes']	['Banana', 'Grapes']	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to delete two neighboring non-identical letters(lower case and upper case) .

Example: AbBbA

lowercase b and uppercase B will get removed

For example:

Input	Result
leEeetcode	leetcode

Answer: (penalty regime: 0 %)

```
def dele(r):
    s=[]
    for i in r:
        if s and s[-1]==i.upper():
            s.pop()
        else:
            s.append(i)
    return "".join(s)
s=input()
r=dele(s)
print(r)
```

	Input	Expected	Got	
✓	leEeetcode	leetcode	leetcode	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Develop a python program to get string values from the user and display the values using circular queue

For example:

Input	Result
4 Python Java C C++	Python Java C C++
5 Java C# C Python C++	Java C# C Python C++

Answer: (penalty regime: 0 %)

Reset answer

```python
import queue
de=queue.Queue()
n=int(input())
for i in range(n):
    x=input()
    de.put(x)
    print(de.get(),end=" ")
```

	Input	Expected	Got	
✓	4 Python Java C C++	Python Java C C++	Python Java C C++	✓
✓	5 Java C# C Python C++	Java C# C Python C++	Java C# C Python C++	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.