In [25]:

```
import math                                                    #for sqrt and atan
```

In [26]:

```
class myComplex:                                               #defining the class
    def __init__(self, r, i = 0.0):
        self.real = r
        self.imag = i

    def __printcomplex__(self):
        if self == None:
            print("\n")
        else:
            print(self.real, " + (", self.imag, ")j")

    def __sum__(self, com_2):
        r_sum = self.real + com_2.real
        i_sum = self.imag + com_2.imag
        return myComplex(r_sum, i_sum)

    def __product__(self, com_2):
        r_prod = self.real*com_2.real - self.imag*com_2.imag
        i_prod = self.real*com_2.imag + self.imag*com_2.real
        return myComplex(r_prod, i_prod)

    def __division__(self, com_2):
        if com_2.real == 0 and com_2.imag == 0:
            print("Division by zero is undefined")
            return myComplex("Undefined", "Undefined")
        else:
            numerator = self.__product__(com_2.__conjugate__())
            denominator = com_2.real*com_2.real + com_2.imag*com_2.imag
            r_div = numerator.real / denominator
            i_div = numerator.imag / denominator
            return myComplex(r_div, i_div)

    def __conjugate__(self):
        r_conj = self.real
        i_conj = -1 * self.imag
        return myComplex(r_conj, i_conj)

    def __modulus__(self):
        return math.sqrt(self.real*self.real + self.imag*self.imag)

    def __phaseangle__(self):
        print("Phase Angle will be given in degrees. It runs from 0 to 360 degrees.")
        if self.real == 0 and self.imag == 0:
            print("Phase Angle is not defined for 0 + 0 j")
        elif self.real > 0 and self.imag > 0:                                  #1st quadrant
            arg = math.degrees(math.atan(self.imag / self.real))
            return arg
        elif self.real < 0 and self.imag > 0:                                  #2nd quadrant
            arg = 180 - math.degrees(math.atan(self.imag / (-1 * self.real)))
            return arg
        elif self.real < 0 and self.imag < 0:                                  #3rd quadrant
            arg = 180 + math.degrees(math.atan(self.imag / self.real))
            return arg
        else:                                                                  #4th quadrant
            arg = 360 - math.degrees(math.atan((-1 * self.imag) / self.real))
            return arg
```

In [27]:

```
#DEFINING THE COMPLEX NUMBERS
a = myComplex(1, 2)
b = myComplex(0, 0)

#PERFORMING OPERATIONS
complex_sum = a.__sum__(b)
complex_product = a.__product__(b)
complex_division = a.__division__(b)
complex_conjugate = b.__conjugate__()
complex_modulus = b.__modulus__()
complex_phase = b.__phaseangle__()

#PRINTING
print("\n", "a")
a.__printcomplex__()

print("\n", "b")
b.__printcomplex__()

print("\n", "a + b")
complex_sum.__printcomplex__()

print("\n", "a * b")
complex_product.__printcomplex__()

print("\n", "a / b")
complex_division.__printcomplex__()

print("\n", "conjugate of b")
```

```
complex_conjugate.__printcomplex__()

print("\n", "modulus of b")
print(complex_modulus)

print("\n", "phase angle of b")
print(complex phase)
```

```
Division by zero is undefined
Phase Angle will be given in degrees. It runs from 0 to 360 degrees.
Phase Angle is not defined for 0 + 0 j

 a
1  + ( 2 )j

 b
0  + ( 0 )j

 a + b
1  + ( 2 )j

 a * b
0  + ( 0 )j

 a / b
Undefined  + ( Undefined )j

 conjugate of b
0  + ( 0 )j

 modulus of b
0.0

 phase angle of b
None
```

In [28]:

```
#DEFINING THE COMPLEX NUMBERS
a = myComplex(1, 2)
b = myComplex(3, 4)

#PERFORMING OPERATIONS
complex_sum = a.__sum__(b)
complex_product = a.__product__(b)
complex_division = a.__division__(b)
complex_conjugate = b.__conjugate__()
complex_modulus = b.__modulus__()
complex_phase = b.__phaseangle__()

#PRINTING
print("\n", "a")
a.__printcomplex__()

print("\n", "b")
b.__printcomplex__()

print("\n", "a + b")
complex_sum.__printcomplex__()

print("\n", "a * b")
complex_product.__printcomplex__()

print("\n", "a / b")
complex_division.__printcomplex__()

print("\n", "conjugate of b")
complex_conjugate.__printcomplex__()

print("\n", "modulus of b")
print(complex_modulus)

print("\n", "phase angle of b")
print(complex phase)
```

```
Phase Angle will be given in degrees. It runs from 0 to 360 degrees.

 a
1  + ( 2 )j

 b
3  + ( 4 )j

 a + b
4  + ( 6 )j

 a * b
-5  + ( 10 )j

 a / b
0.44  + ( 0.08 )j

 conjugate of b
3  + ( -4 )j

 modulus of b
5.0

 phase angle of b
```

phase angle of b
53.13010235415598

In [29]:

```
#DEFINING THE COMPLEX NUMBERS
a = myComplex(0, 0)
b = myComplex(0, 0)

#PERFORMING OPERATIONS
complex_sum = a.__sum__(b)
complex_product = a.__product__(b)
complex_division = a.__division__(b)
complex_conjugate = b.__conjugate__()
complex_modulus = b.__modulus__()
complex_phase = b.__phaseangle__()

#PRINTING
print("\n", "a")
a.__printcomplex__()

print("\n", "b")
b.__printcomplex__()

print("\n", "a + b")
complex_sum.__printcomplex__()

print("\n", "a * b")
complex_product.__printcomplex__()

print("\n", "a / b")
complex_division.__printcomplex__()

print("\n", "conjugate of b")
complex_conjugate.__printcomplex__()

print("\n", "modulus of b")
print(complex_modulus)

print("\n", "phase angle of b")
print(complex_phase)
```

```
Division by zero is undefined
Phase Angle will be given in degrees. It runs from 0 to 360 degrees.
Phase Angle is not defined for 0 + 0 j

 a
0  + ( 0 )j

 b
0  + ( 0 )j

 a + b
0  + ( 0 )j

 a * b
0  + ( 0 )j

 a / b
Undefined  + ( Undefined )j

 conjugate of b
0  + ( 0 )j

 modulus of b
0.0

 phase angle of b
None
```

In [30]:

```
#DEFINING THE COMPLEX NUMBERS
a = myComplex(0, 0)
b = myComplex(-1, 3)

#PERFORMING OPERATIONS
complex_sum = a.__sum__(b)
complex_product = a.__product__(b)
complex_division = a.__division__(b)
complex_conjugate = b.__conjugate__()
complex_modulus = b.__modulus__()
complex_phase = b.__phaseangle__()

#PRINTING
print("\n", "a")
a.__printcomplex__()

print("\n", "b")
b.__printcomplex__()

print("\n", "a + b")
complex_sum.__printcomplex__()

print("\n", "a * b")
complex_product.__printcomplex__()

print("\n", "a / b")
```

```
complex_division.__printcomplex__()

print("\n", "conjugate of b")
complex_conjugate.__printcomplex__()

print("\n", "modulus of b")
print(complex_modulus)

print("\n", "phase angle of b")
print(complex_phase)
```

```
Phase Angle will be given in degrees. It runs from 0 to 360 degrees.

 a
0  + ( 0 )j

 b
-1  + ( 3 )j

 a + b
-1  + ( 3 )j

 a * b
0  + ( 0 )j

 a / b
0.0  + ( 0.0 )j

 conjugate of b
-1  + ( -3 )j

 modulus of b
3.1622776601683795

 phase angle of b
108.43494882292201
```

In [ ]:

```
```