```
In [ ]: # Q1. Declare a complex number and store it in a variable.
        # Check the type and print the id of the same.
```

```
In [2]: x=4+9j
        print(x)
        print(type(x))
        print(id(x))
```

```
(4+9j)
<class 'complex'>
2343470768848
```

```
In [ ]:
```

```
In [ ]: Q2. Arithmetic Operations on complex number
        Take two different complex numbers.
        Store them in two different variables.
        Do below operations on them:-
         Find sum of both numbers
         Find difference between them
         Find the product of both numbers.
         Find value after dividing first num with second number
         Find the result of the first num to the power of the second number.
```

```
In [4]: # Take two different complex numbers.
        # Store them in two different variables.
        x=4+9j
        y=2+2j
        print(x)
        print(y)
        print(type(x))
        print(type(y))
```

```
(4+9j)
(2+2j)
<class 'complex'>
<class 'complex'>
```

In [5]:
```
# sum of both numbers
print(x+y)
# difference between them
print(x-y)
# the product of both numbers.
print(x*y)
# value after dividing first num with second number
print(x/y)
# the result of the first num to the power of the second number.
print(x**y)
```

```
(6+11j)
(2+7j)
(-10+26j)
(3.25+1.25j)
(8.003445128934182+5.436410773160743j)
```

In [ ]:
```
Q3. Comparison Operation not applicable between instance of complex values.
Object reusability concept is not applicable on complex number
```

In [1]:
```
x=4+9j
y=3-1j
print(x==y)
```

```
False
```

In [3]:
```
#Comparison Operation not applicable between instance of complex values.
print(x>y)
print(x<y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 2
      1 #Comparison Operation not applicable between instance of complex valu
es.
----> 2 print(x>y)
      3 print(x<y)

TypeError: '>' not supported between instances of 'complex' and 'complex'
```

In [8]:
```
#Object reusability concept is not applicable on complex number
a=2+3j
b=2+3j
```

In [9]:
```
a is b
```

Out[9]:
```
False
```

In [ ]:
```
```

In [ ]:
```
Q4. Equality Operator
Take two different complex numbers.
Store them in two different variables.
Equate them using equality operators (==, !=)
Observe the output(return type should be boolean)
```

In [10]:
```
# Take two different complex numbers.
# Store them in two different variables.
x=4+9j
y=3-1j
print(x)
print(y)
```

```
(4+9j)
(3-1j)
```

In [11]:
```
x==y
```

Out[11]:
```
False
```

In [12]:
```
x!=y
```

Out[12]:
```
True
```

In [ ]:
```

```

In [ ]:
```
Q5. Logical operators
Observe the output of below code
Cross check the output manually
print(10+20j and 20+30j) #20+30j
#----------------------------------------->Output is 20+30j
print(0+0j and 20+30j) #0+0j
#----------------------------------------->Output is 0j
print(20+30j and 0+0j) #0+0j
#----------------------------------------->Output is 0j
print(0+0j and 0+0j) #0+0j
#----------------------------------------->Output is 0j
print(10+20j or 20+30j) #10+20j
#----------------------------------------->Output is 10+20j
print(0+0j or 20+30j) #20+30j
#----------------------------------------->Output is 20+30j
print(20+30j or 0+0j) #20+30j
#----------------------------------------->Output is 20+30j
print(0+0j or 0+0j) #0+0j
#----------------------------------------->Output is 0j
print(not 10+20j) #False
#----------------------------------------->Output is False
print(not 0+0j) #True
#----------------------------------------->Output is True
```

In [13]:
```
print(10+20j and 20+30j) #20+30j
print(0+0j and 20+30j) #0+0j
print(20+30j and 0+0j) #0+0j
print(0+0j and 0+0j) #0+0j
print(10+20j or 20+30j) #10+20j
print(0+0j or 20+30j) #20+30j
print(20+30j or 0+0j) #20+30j
print(0+0j or 0+0j) #0+0j
print(not 10+20j) #False
print(not 0+0j) #True
```

```
(20+30j)
0j
0j
0j
(10+20j)
(20+30j)
(20+30j)
0j
False
True
```

In [ ]:

In [ ]:
```


Q6. What is the output of the expression inside the print statement.
Cross check before running the program.
a = 10+20j
b = 10+20j
print(a is b) #False #True or False?
print(a is not b) #True #True or False?
```

In [15]:
```
a = 10+20j
b = 10+20j
print(a is b)
print(a is not b)
```

```
False
True
```

In [ ]:

In [ ]:

```


Q7. Membership operation
in, not in are two membership operators and it returns boolean value
print('2.7' in 'Python2.7.8') #True
print(10+20j in [10,10.20,10+20j,'Python']) #True
print(10+20j in (10,10.20,10+20j,'Python')) #True
print(30+40j in {1,20.30,30+40j}) #True
print(30+40j in {1:100, 2.3:200, 30+40j:300}) #True
print(10 in range(20)) #True
```

In [16]:

```
print('2.7' in 'Python2.7.8')
print(10+20j in [10,10.20,10+20j,'Python'])
print(10+20j in (10,10.20,10+20j,'Python'))
print(30+40j in {1,20.30,30+40j}) #True
print(30+40j in {1:100, 2.3:200, 30+40j:300})
print(10 in range(20))
```

```
True
True
True
True
True
True
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: