Complex object demo Program:

```
a = 3 + 4j  ——  # Ref 'a' points to object 3+4j
print(a)  ——  # value of object 'a' i.e. 3+4j
print(type(a))  ——  # type of object 'a' i.e < class 'complex'>
print(id(a))  ——  # Address of object 'a'
print (a.real)  ——  # 3.0 3+4j where real is 3.0 and
                      imag is 4.0
print (a.imag)  ——  # 4.0
print (type (a.real))  ——  # type of obj 'a' i.e <class
                              ~~complex~~ 'float'>
                       where 3.0 is a float.
print (type (a.imag))  —  # <class 'float'> where 4.0
                             is float.
```

Find Outputs:

```
a = 6j  —  # Ref 'a' points to object 6j
print (a)  —  # value of object 'a' i.e 6j
print(type (a))  —  # type of object 'a' i.e <class 'complex'>
print (a.real)  —  # 0.0 (real part) in 6j
print (a.imag)  —  # 6.0 (imag part) in 6j
print (5+j6)  —  # Error (Syntax Error) [6j not j6]
print (3+4i)  —  # Error (syntax Error) [only supports
                                   j not any other]
print (4+j)  —  # Error (syntax Error) [j is not defined]
print (4+1j)   — # 4+1j
print (4+0j)   — # 4+0j

# Bool object demo program:

 a= True       — # Ref 'a' points of object True.
print (a)      — # value of object 'a' i.e True
print (type(a))  — # type of object 'a' i.e <class 'bool'>
 b= False      — # Ref 'b'. points to object False
print (b)      — # Value of object 'b' i.e False.
print (type(b))  — # type of object 'b' i.e <class 'bool'>
print (True+
       True)   — 2 # True=1, False=0. (1+1=2)
print (True+False) — # 1 # True=1, False=0 (1+0=1)
print (False + True) — # 1 (1+0=1)
print (False +False) — # 0  False=0, True=1
print (True + True + True)  —— # 3  True=1 (1+1+1)
print (25+10.8 + True)  —— # 36.8 (25+10.8+1) python
                 treats 25 (int) & 10.8 (float)
              as objects when operations are made
              on them.

print ( True > False)  —— # True. because True = 1,
                  False = 0  So, (1>0)

print (True) - # True

print (False) - # False
```

print (true) — # Error, because true is not in upper case. python only support Upper case not lowercase.

print (false) - # Error Syntax Error.

Find outputs :

a = 006247 ——— # Ref 'a' points to object 006247

print (a) — ~~# value of object 'a' i.e <class 'binary int'>~~

006247 is octal because of octal use only 0 to 7 no.s. here octal converted to decimal

$6\times8^3 + 2\times8^2 + 4\times8^1 + 7\times8^0$

$= 6\times512 + 2\times64 + 4\times8 + 7\times1$

$= 3072 + 128 + 32 + 7$

print (a) — # 3239.

print (type (a)) — # type of obj 'a' <class 'int'>

print (id (a)) — # Address of object 'a'

b = 0o 6247 — # Ref 'b' points to object 0o 6247

print (b) —— # 3239 As it supports lower & Upper case letters and the objects are reusable.

print (id (b)) — # Address of object 'b'.

c = 3239 — # Ref ~~of~~ 'c' points to object 3239

print (c) — # 3239 Value of object 'a' i.e 3239

print (id (c)) — # Address of object 'c' print (0o9248) —— # Error because octal contains only 0 to 7, here we have 9 and 8 so Syntax Error.

Find outputs :

a = 0XA7B9 —— # Ref 'a' ~~the~~ points to object 0XA7B9

print(a) —— # 0XA7B9 is hexadecimal because it accepts 0 to 9 A to F.

$A \times 16^3 \quad 7 \times 16^2 \quad B \times 16^1 \quad 9 \times 16^0$

$10 \times 16^3 + 7 \times 16^2 + 11 \times 16 + 9 \times 1$

$= 40960 + 1792 + 176 + 9$

$= 42937$

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

print (type (a)) — # type of obj 'a' <class 'int'>.

b = 0XBEEF — # Ref 'a' points to object 0XBEEF

print (A7B9) — #Error A7B9 not defined

print ('A7B9') — # A7B9 as it is quoted which mean string.

print (0XBEER) — # Error because R is not an hexadecimal

print (0XH4D) — # Error because H and 4 are not defined as hexadecimal.

print (0xA7G9B) — # Error as G is not a hexadecimal.

```
# Find outputs :

a = 9248          — # Ref 'a' points to obj 9248
print (a)         — # value of object 'a' i.e 9248
print (type (a))  — # type of obj 'a' i.e <class 'int'>
```

16|7|25

```
# Find outputs :

a = "Rama Rao"
print (a)          — # Value of obj 'a' i.e Rama Rao.
print(type (a))    — # < ~~int~~ class 'str~~ing~~' >
print (id(a))      — # Address of object a.

b = 'Hyd'
                     # Value of obj 'b'. i.e 'Hyd'
```

```
print ( how to print 'd'
print ( how to print 'y'
print ( how to print 'H'

print (a[-4])        — #
                       in
print (a[0] == a[-3])
a [2] = 'c'          — #
print (25 [0])       —
print ('25' [0])     —
print (True [1])     —
print ('True' [1])   —

# Find output :
```