Here's how you can accomplish each of these tasks:

**1. Create Thing class and print it**

class Thing:

pass

print(Thing) # Prints the class itself

example = Thing()

print(example) # Prints the object instance

**Printed values will be different** because one is the class reference, and the other is an instance.

**2. Create Thing2 with a class attribute**

class Thing2:

letters = 'abc'

print(Thing2.letters) # Accessing class attribute

**3. Create Thing3 with an instance attribute**

class Thing3:

def \_\_init\_\_(self):

self.letters = 'xyz'

example = Thing3()

print(example.letters) # Accessing instance attribute

Yes, you **must** create an object to access instance attributes.

**4. Create Element class with attributes**

class Element:

def \_\_init\_\_(self, name, symbol, number):

self.name = name

self.symbol = symbol

self.number = number

hydrogen = Element('Hydrogen', 'H', 1)

**5. Create an Element object from a dictionary**

element\_dict = {'name': 'Hydrogen', 'symbol': 'H', 'number': 1}

hydrogen = Element(\*\*element\_dict) # Unpacking dictionary into arguments

**6. Define dump() method to print attributes**

class Element:

def \_\_init\_\_(self, name, symbol, number):

self.name = name

self.symbol = symbol

self.number = number

def dump(self):

print(f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}')

hydrogen = Element('Hydrogen', 'H', 1)

hydrogen.dump()

**7. Modify dump() method to \_\_str\_\_**

class Element:

def \_\_init\_\_(self, name, symbol, number):

self.name = name

self.symbol = symbol

self.number = number

def \_\_str\_\_(self):

return f'Name: {self.name}, Symbol: {self.symbol}, Number: {self.number}'

hydrogen = Element('Hydrogen', 'H', 1)

print(hydrogen) # Automatically calls \_\_str\_\_

**8. Make attributes private and add getters**

class Element:

def \_\_init\_\_(self, name, symbol, number):

self.\_\_name = name

self.\_\_symbol = symbol

self.\_\_number = number

@property

def name(self):

return self.\_\_name

@property

def symbol(self):

return self.\_\_symbol

@property

def number(self):

return self.\_\_number

hydrogen = Element('Hydrogen', 'H', 1)

print(hydrogen.name, hydrogen.symbol, hydrogen.number) # Access via getters

**9. Define Bear, Rabbit, and Octothorpe classes**

class Bear:

def eats(self):

return 'berries'

class Rabbit:

def eats(self):

return 'clover'

class Octothorpe:

def eats(self):

return 'campers'

bear = Bear()

rabbit = Rabbit()

octothorpe = Octothorpe()

print(bear.eats()) # 'berries'

print(rabbit.eats()) # 'clover'

print(octothorpe.eats()) # 'campers'

**10. Define Laser, Claw, SmartPhone, and Robot**

class Laser:

def does(self):

return 'disintegrate'

class Claw:

def does(self):

return 'crush'

class SmartPhone:

def does(self):

return 'ring'

class Robot:

def \_\_init\_\_(self):

self.laser = Laser()

self.claw = Claw()

self.smartphone = SmartPhone()

def does(self):

return f'Laser: {self.laser.does()}, Claw: {self.claw.does()}, SmartPhone: {self.smartphone.does()}'

robot = Robot()

print(robot.does())