Think Python:
Chapter 15

A programmer-defined type is known as a(n) _____.

A programmer-defined type is known as a(n) _____.

`class`

Defining a class with a name creates a class _____.

Defining a class with a name creates a class _____.

```
object

>>> class Airedale:
...     """Represents an Airedale Terrier.
...
...     Attributes: sex, age, weight
...     """
...
```

Creating a new class object is called _____.

Creating a new class object is called _____.

instantiation

Creating a new class object is accomplished via calling the class as if it were a(n) _____

Creating a new class object is accomplished via calling the class as if it were a(n) _____

`function`

A new object is said to be an _____ of its class.

A new object is said to be an _____ of its class.

`instance`

The prefix 0x in front of a number means that the number is in _____.

The prefix 0x in front of a number means that the number is in _____.

Hexadecimal (base 16: 0-9, A, B, C, D, E, F)

Elements of an object, to which one can assign values, are called _____.

Elements of an object, to which one can assign values, are called _____.

`attributes`

An attribute of an object can itself be an object, in which case, the second object is said to be _____.

An attribute of an object can itself be an object, in which case, the second object is said to be _____.

embedded

```
>>> Charlotte = Airedale()
>>> Charlotte.sex = "female"
>>> Charlotte.age = 10.25
>>> Charlotte.weight = 70
>>> Charlotte.sex
'female'
>>> Charlotte.age
10.25
>>> Charlotte.weight
70
>>> Charlotte.hair
```

```
>>> Charlotte = Airedale()
>>> Charlotte.sex = "female"
>>> Charlotte.age = 10.25
>>> Charlotte.weight = 70
>>> Charlotte.sex
'female'
>>> Charlotte.age
10.25
>>> Charlotte.weight
70
>>> Charlotte.hair

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Airedale' object has no attribute 'hair'
>>>
```

```
>>> def diet_results(dog):
...     dog.weight *= .9
...
>>> diet_results(Charlotte)
>>> Charlotte.sex
???
>>> Charlotte.age
???
>>> Charlotte.weight
???
>>>
```

```
>>> def diet_results(dog):
...     dog.weight *= .9
...
>>> diet_results(Charlotte)
>>> Charlotte.sex
'female'
>>> Charlotte.age
10.25
>>> Charlotte.weight
63.0
>>>
```

The point here is that the parameter *dog* is an alias for Charlotte, so changes to *dog* also change *Charlotte*.

Since aliasing can be difficult to deal with, an alternative is to copy an object, for which one must import the _____ module.

Since aliasing can be difficult to deal with, an alternative is to copy an object, for which one must import the _____ module.

`copy`

What is the term for this form of copying?

```
>>> import copy
>>> Charlotte = Airedale()
>>> Charlotte.weight = 70
>>> Emmy = copy.copy(Charlotte)
>>> Emmy.weight
70
>>> Emmy.weight = 45
>>> Emmy.weight
45
>>> Charlotte.weight
70
>>>
```

What is the term for this form of copying?

```
>>> import copy
>>> Charlotte = Airedale()
>>> Charlotte.weight = 70
>>> Emmy = copy.copy(Charlotte)
>>> Emmy.weight
70
>>> Emmy.weight = 45
>>> Emmy.weight
45
>>> Charlotte.weight
70
>>>
```

Shallow copy, because any embedded objects would not be copied.