1.

OOP is about code reuse – you factor code to minimize redundancy and program by customizing what already exists instead of changing code in place or starting from scratch.

2.

An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from left to right (by default). The search stops at the first place the attribute is found.

3.

Classes are a kind of factory for creating multiple instances. Classes also support operator overloading methods, which instances inherit, and treat any functions nested in the class as methods for processing instances.

4.

It always receives the instance object that is the implied subject of the method call. It’s usually called 'self' by convention.

5.

If the __init__ method is coded or inherited in a class, Python calls it automatically each time an instance of that class is created.

```
class New():
def __init__(self, arg1, arg2):
self.first_var = arg1
self.second_var = arg2
```

6.

we create a class instance by calling the class name as though it were a function; any arguments passed into the class name show up as arguments two and beyond in the __init__ constructor method.

```
x = ClassName()
y = AnotherClass(arg1, arg2)
```

7.

```
class ClassName():
#some code here
```

8.

They are classes which are used to inherit from.

```
class Son(Father, Mother): ...
```

In this case Father and Mother are superclasses for Son subclass.