```
"""
1. The scope of a variable in python is depends on where the value for the va
   is assigned in your source code.!!
2. variables assigned inside a function can only be seen by the code within t
   You cannot even refer to such variables from outside the function.
3. variables assigned inside a function do not clash with variables outside t
   even if the same variable names are used elsewhere. A variable named 'X' a
   (i.e., in a different function or at the top level of a module file)
   is a completely different variable from a name X assigned inside that func
"""
# Global variable
a = 10
b = 20

def func():
    return a + 1  # Prints message at global scope

def func():
    # the value for b is assigned inside the function
    b = 20  # Local Variable
    # the value for a is assigned outside the function
    return a + b   # a refers to value 10 which is Global variable

def func():
    # the value for a and b are assigned inside the function. So a and b are
    a = 20  # a is a local variable.
    b = 20  # b is a local Variable
```

```
28      return a + b
29
30  def func():
31      # Local Variable "a"!
32      # A variable can be either local or global variable but not both
33      result = a + b  # Exception! (a will not refer to Global value 10)
34      a = 20
35      b = 30
36      result = a + b
37      return result
38  func()
39
40  # Raises Exception!
41  """
42  Traceback (most recent call last):
43    File "<pyshell#9>", line 1, in <module>
44      func()
45    File "<pyshell#8>", line 4, in func
46      result = a + b  # Exception! (a will not refer to Global value 10)
47  UnboundLocalError: local variable 'a' referenced before assignment
48  """
49
50  def func():
51      # the values for a and b are assigned inside the function.
52      # So, 'a' and 'b' are considered as local vairables and not globals
53      a = a + 1
54      b = b + 1
55      return a + b
56
57  # Raises Exception!
58  """
59  Traceback (most recent call last):
60    File "<pyshell#5>", line 1, in <module>
61      func()
62    File "<pyshell#4>", line 4, in func
63      a = a + 1
64  UnboundLocalError: local variable 'a' referenced before assignment
65  """
66  func()
67
68  # UnboundLocalError is also caused when you try to assign a variable before a
69  # Example:
70  def func():
71      a = a + 1       # Adding 1 to un-initlised variable 'a'
72
```

```
73  def func():
74      # Local "a" to func and enclosing scope for wrapper
75      a = 20
76      def wrapper():
77          # Local "a" to wrapper
78          a = 30
79          return a + 1
80      return wrapper()
81
82  def func():
83      # local for func
84      a = 10
85      def wrapper():
86          a = a + 1
87          a = 500
88      return wrapper()
89
90  func()
91
92  # Raises Exception!
93  """
94  Traceback (most recent call last):
95    File "<pyshell#5>", line 1, in <module>
96      func()
97    File "<pyshell#4>", line 4, in func
98      a = a + 1
99  UnboundLocalError: local variable 'a' referenced before assignment
100 """
101 # len = "Global variable len"
102 def func():
103     # Local "a" to func and enclosing scope for wrapper
104     # len = "local len of func"
105     def wrapper():
106         # Local "a" to wrapper
107         # len = "local len of wrapper"
108         print(len)
109     return wrapper()
110
111 # UnboundLocalError is also caused when you try to assign a variable before a
112 def func():
113     # Declaring that the variable that you are referring to is global variabl
114     global a  # Refers to Global Variable "a"
115     a = 20
116     print(a)
117 # ------------------------------------------------------------------------
```

```
# variables inside comprehension will have its own scope.
number = -5
numbers = [ number for number in range(1, 8) ]
# prints -5
print(number)
# ------------------------------------------------------------------------
"""
1. In case of classes, when you look up for an attribute "message",Python tri
2. If the attribute exist on the instance, then it will return the value of t
3. If the attribute does not exist on the instance, it will lookup for the at
4. If the attribute exist on the class level, it will return the value of the
5. If the attribute does not exist on instance and at class level,then attrib
"""
class Spam:
    message = "Hello world"
    def __init__(self):
        self.message = "Hello universe"

s = Spam()
print(s.message)   # Prints "Hello universe"

class Spam:
    message = "Hello world"
    def __init__(self):
        self.x = 10
        self.y = 20

s = Spam()
print(s.message)    # Prints "Hello world"

# ------------------------------------------------------------------------
# Global variable
a = 10

def spam(number):
    number = number + 1
    print(number)

# Passing a immutable object to the function
# in this case python acts as call by value
spam(a)
print(a)
# ------------------------------------------------
a = [10]

```

```
163 def spam(numbers):
164     numbers = numbers.append(11)
165     print(numbers)
166 
167 # Passing a mutable object to the function
168 # in this case python acts as call by reference
169 spam(a)
170 print(a)
```