This code:

**>>>** x = 10

**>>> def** bar():

**...**  print(x)

**>>>** bar()

10

works, but this code:

**>>>** x = 10

**>>> def** foo():

**...**  print(x)

**...**  x += 1

results in an UnboundLocalError:

**>>>** foo()

Traceback (most recent call last):

*...*

UnboundLocalError: local variable 'x' referenced before assignment

This is because when you make an assignment to a variable in a scope, that variable becomes local to that scope and shadows any similarly named variable in the outer scope. Since the last statement in foo assigns a new value to x, the compiler recognizes it as a local variable. Consequently, when the earlier print(x) attempts to print the uninitialized local variable and an error result.

Solution use nonlocal or global keywords