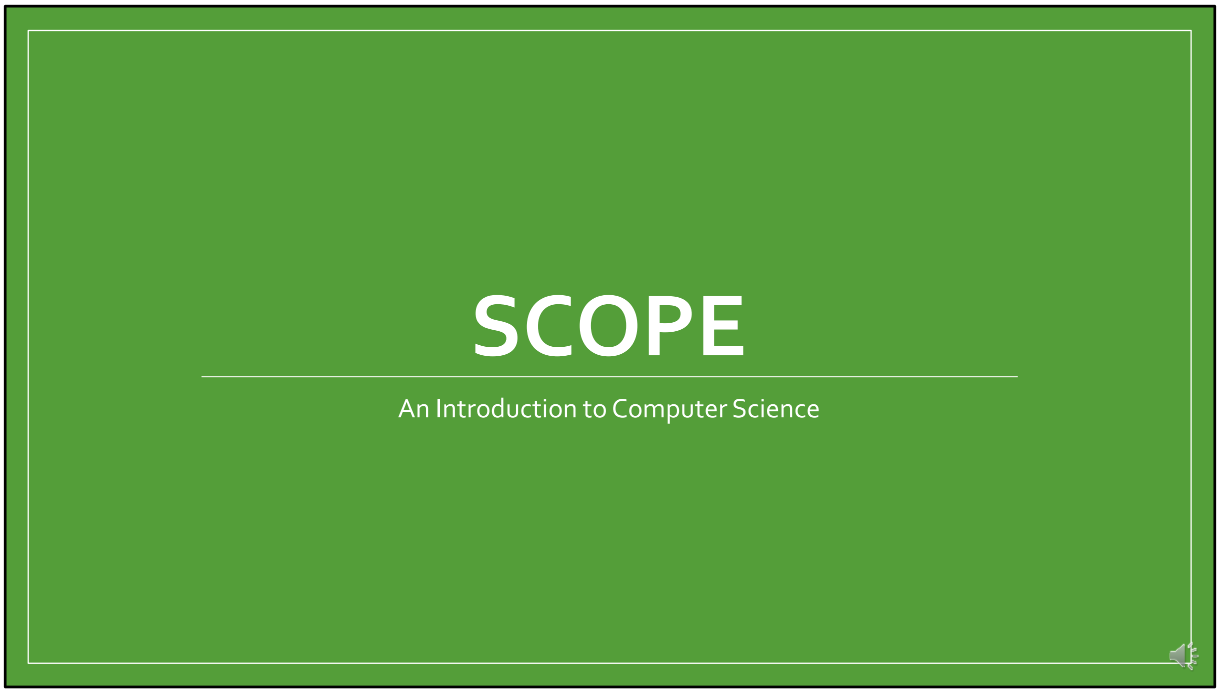
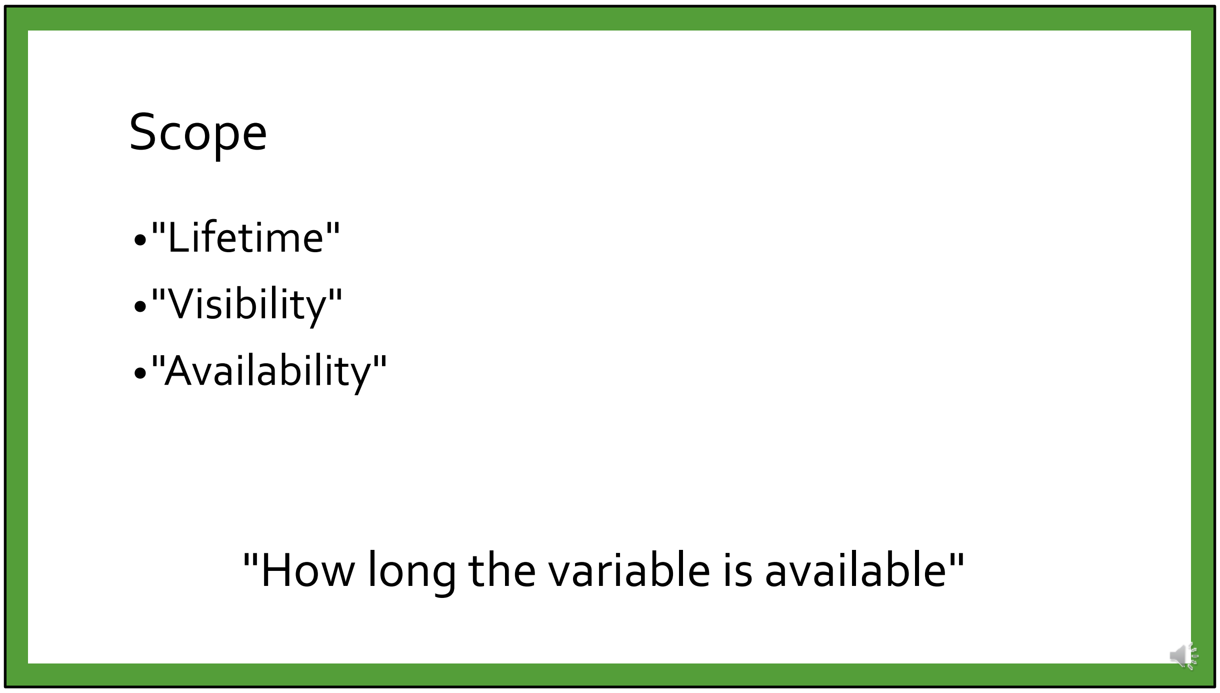


# SCOPE

An Introduction to Computer Science

Let's learn about Scope.

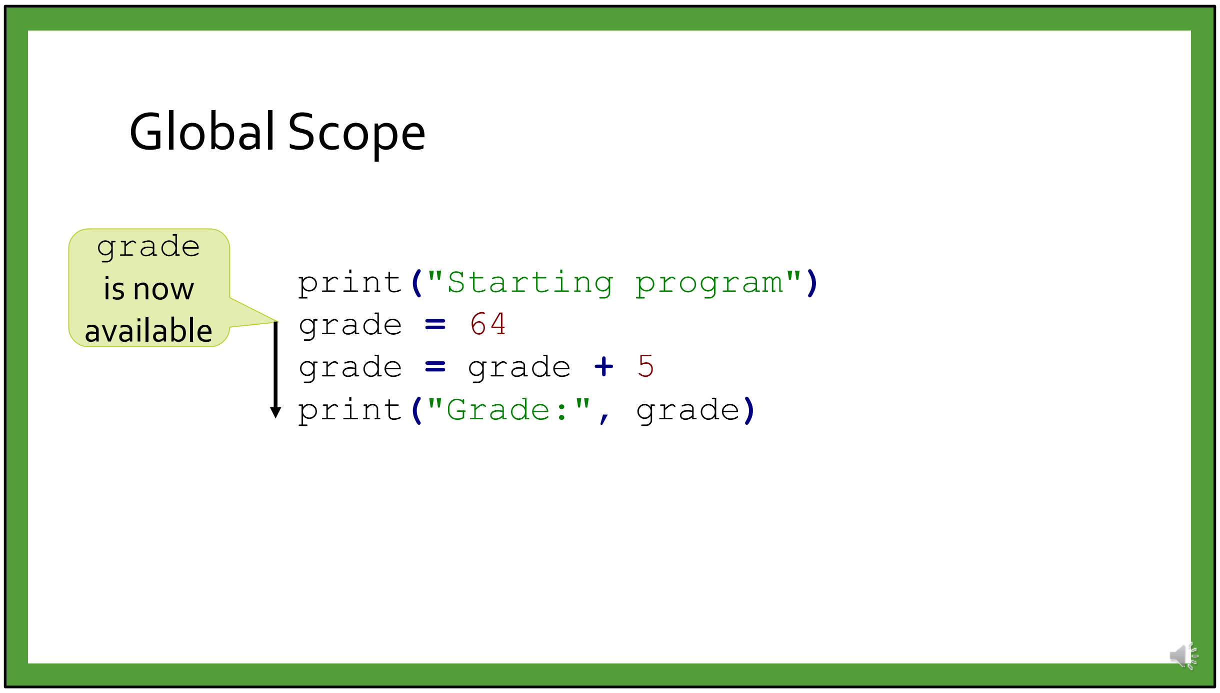


## Scope

- "Lifetime"
- "Visibility"
- "Availability"

"How long the variable is available"

```
In a program, the **scope** of a variable indicates how long that v
ariable is available.
This is also known as the "lifetime" or "visibility" of a variable.
```

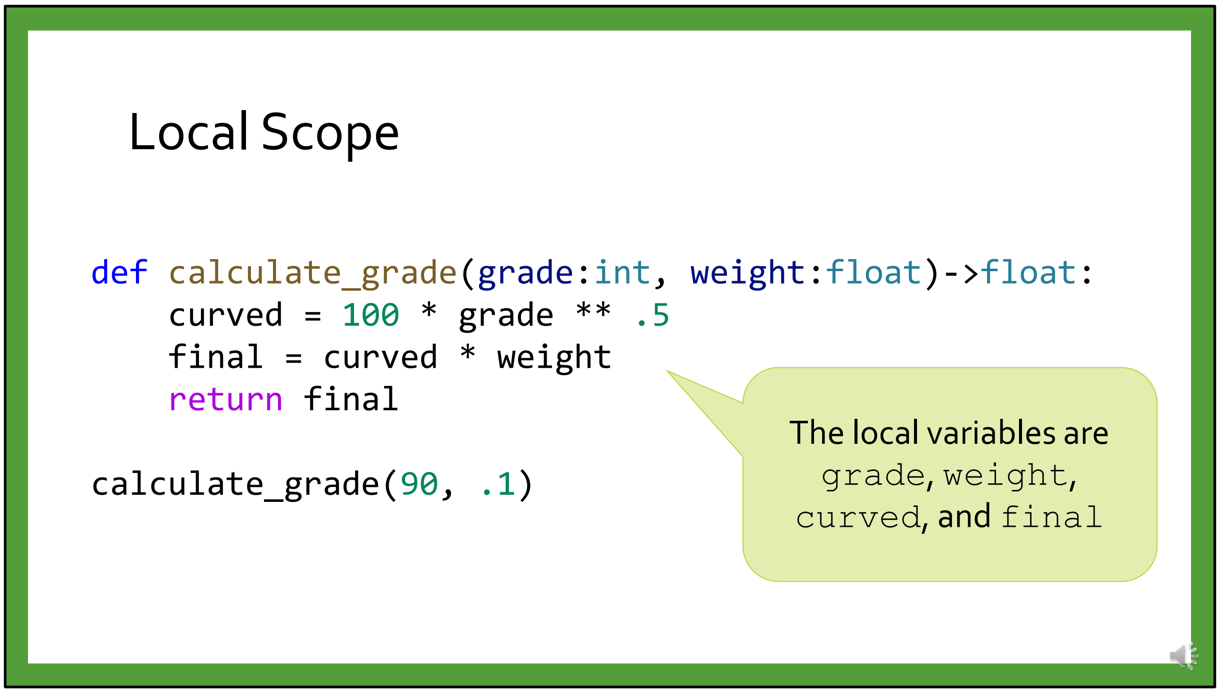


## Global Scope

grade is now available

```
print("Starting program")
grade = 64
grade = grade + 5
print("Grade:", grade)
```

Variables defined at the top level are known as global variables.
Once a variable is defined, it is available on subsequent lines.
That variable lives until the end of the program.

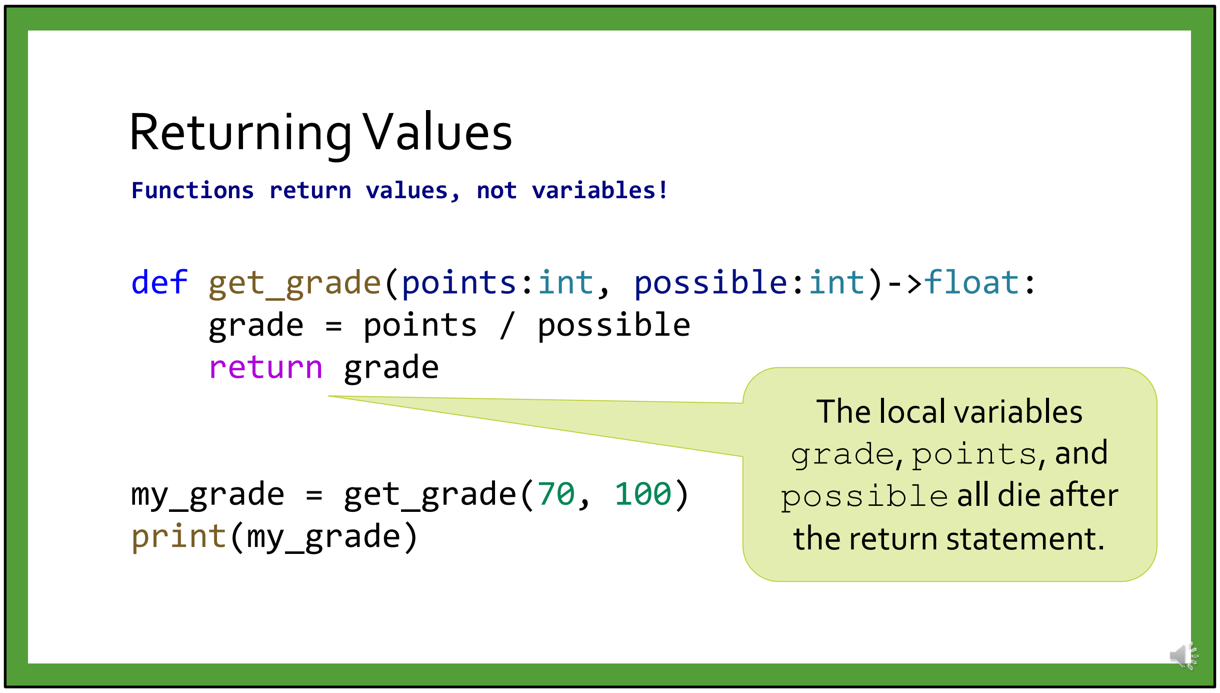


## Local Scope

```
def calculate_grade(grade:int, weight:float)->float:
    curved = 100 * grade ** .5
    final = curved * weight
    return final

calculate_grade(90, .1)
```

The local variables are `grade`, `weight`, `curved`, and `final`

Each function has its own local scope.
Variables defined as parameters or within a function live until the
 function ends.
These are local variables.
Variables defined in one function's are not available outside the f
unction.
This simplifies the reading of any function -
 you only need to worry about
 things defined in the function itself.



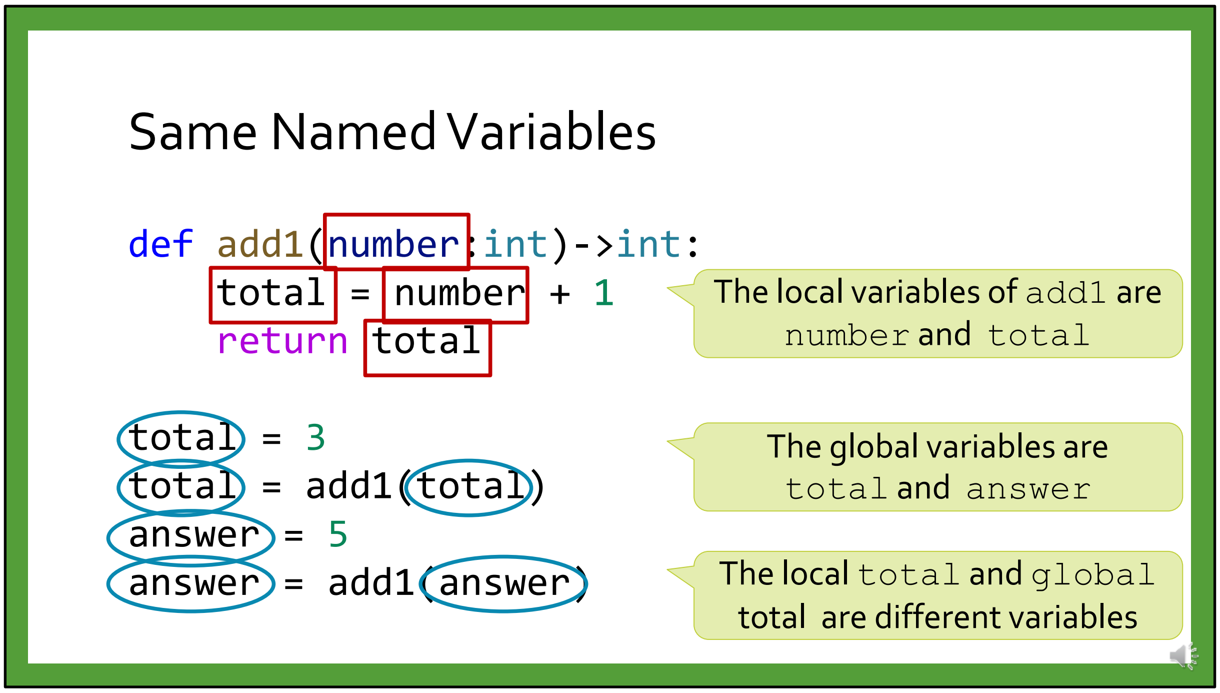
## Returning Values

**Functions return values, not variables!**

```
def get_grade(points:int, possible:int)->float:
    grade = points / possible
    return grade


my_grade = get_grade(70, 100)
print(my_grade)
```

The local variables `grade`, `points`, and `possible` all die after the return statement.

```
Functions return values, not variables.
This is so important, I'm going to say it again:
**functions return values, not variables.**
A variable has a value, so when you write a statement like the one
shown, you are returning the variable's value, not the variable its
elf.
The variable disappears after the function ends, so returning the v
alue
is the only way to make it available.
```



## Same Named Variables

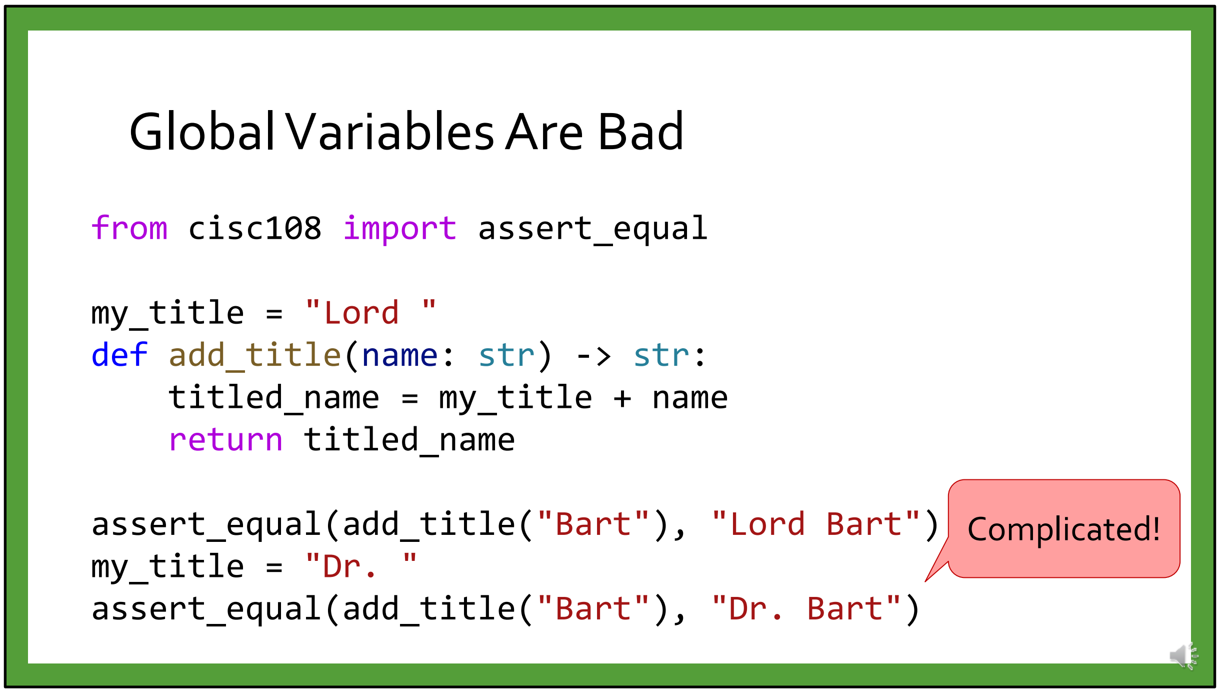
```python
def add1(number:int)->int:
    total = number + 1
    return total

total = 3
total = add1(total)
answer = 5
answer = add1(answer)
```

The local variables of `add1` are `number` and `total`

The global variables are `total` and `answer`

The local `total` and `global` total are different variables

Beginners will sometimes try to reuse a variable name
Any global variables with the same name are actually unrelated to t
he
variable inside the function.
On this slide, I have drawn squares around local variables, and cir
cles
around global variables.

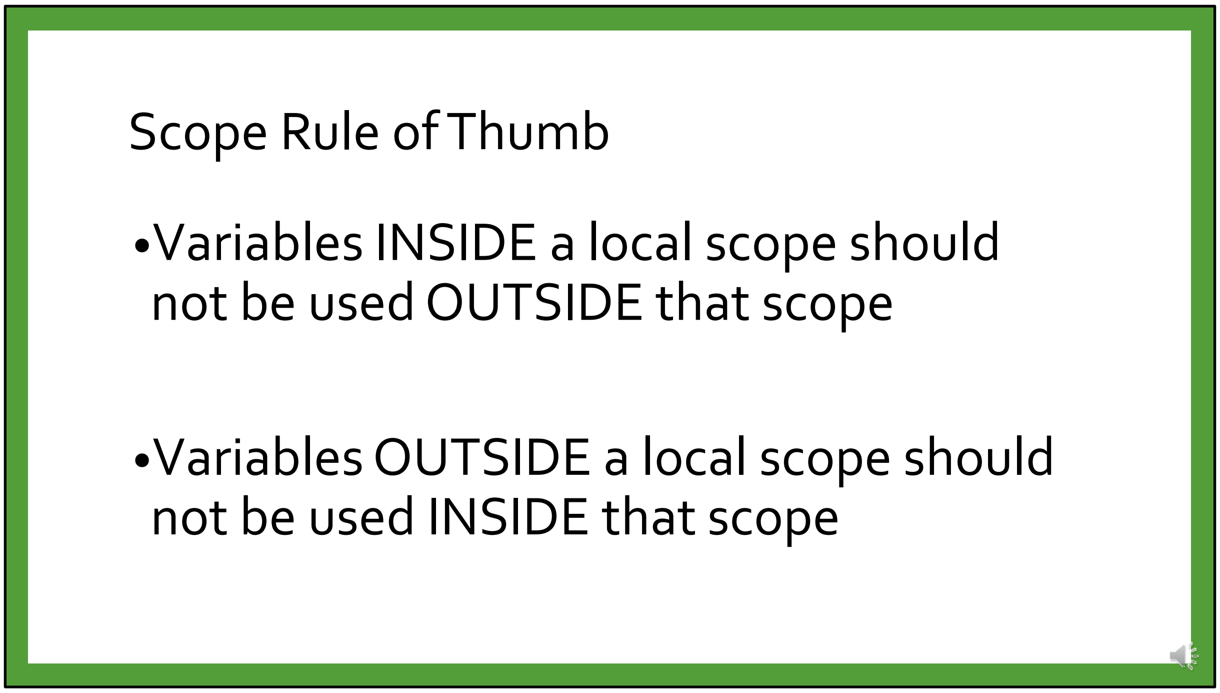


## Global Variables Are Bad

```python
from cisc108 import assert_equal

my_title = "Lord "
def add_title(name: str) -> str:
    titled_name = my_title + name
    return titled_name

assert_equal(add_title("Bart"), "Lord Bart")
my_title = "Dr. "
assert_equal(add_title("Bart"), "Dr. Bart")
```

Complicated!

It is technically possible to read a global variable inside a funct
ion.
However, you should not do so.
Every time you refer to global variables, your program becomes more
 complicated
 and you have to think about multiple levels of scope.
In this code example shown here, the unit tests would fail if we sw
apped
the order of the last two lines.
This may work out okay in smaller programs, but causes huge problem
s as you
start writing longer programs.
Whenever you feel the urge to use a global variable, stop and recon
sider.
The only exception is if you are 100% certain that the global varia
ble's value
will stay constant and never change.


## Scope Rule of Thumb

- Variables INSIDE a local scope should not be used OUTSIDE that scope
- Variables OUTSIDE a local scope should not be used INSIDE that scope

Here is a simple pair of rules for working with scope:
Variables inside a local scope should not be used outside that scope.
Variables outside a local scope should not be used inside that scope.
Keeping these two rules in mind will avoid many headaches.

Okay, are Global Variables really bad? Let's discuss further:
http://wiki.c2.com/?GlobalVariablesAreBad