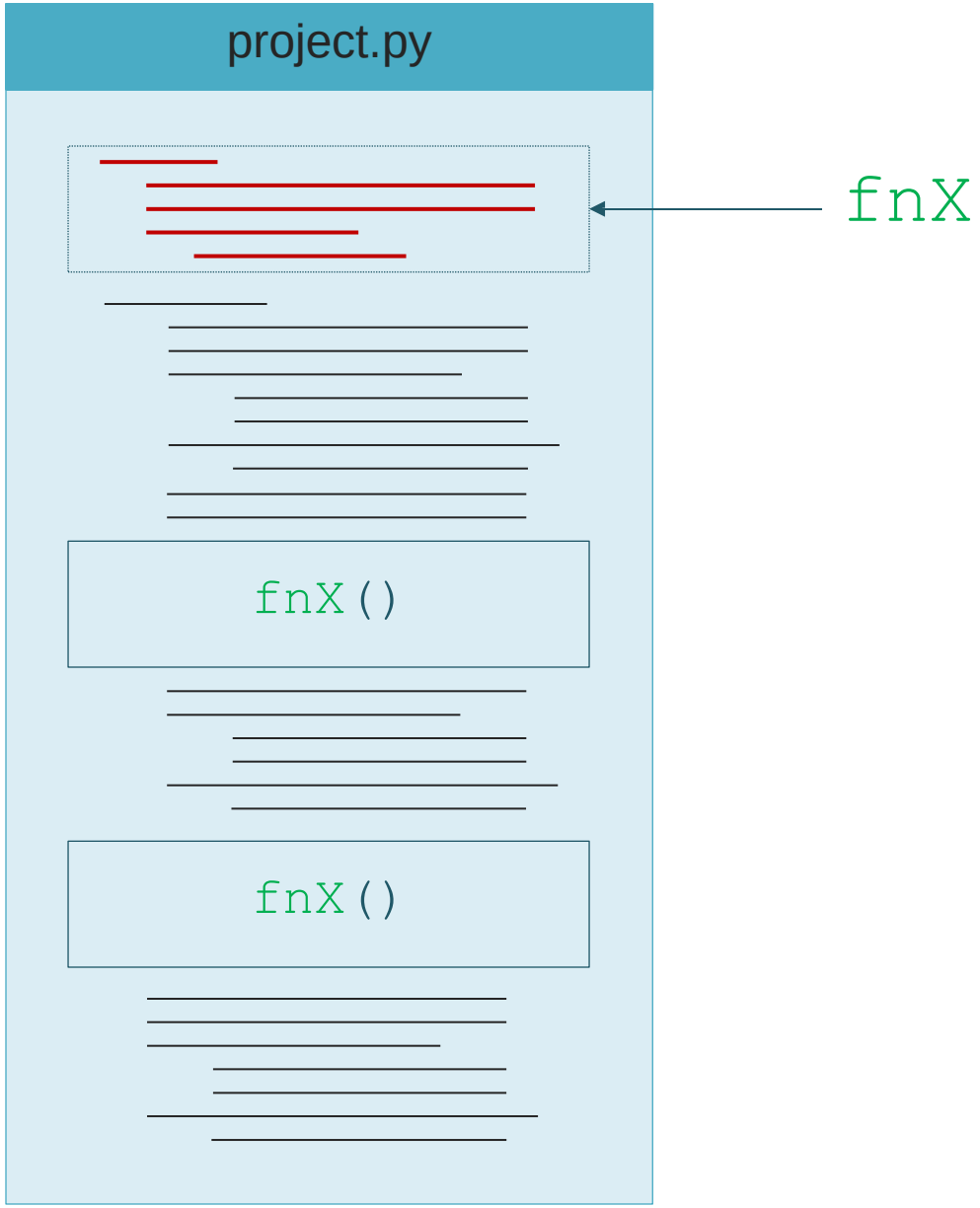
Functions

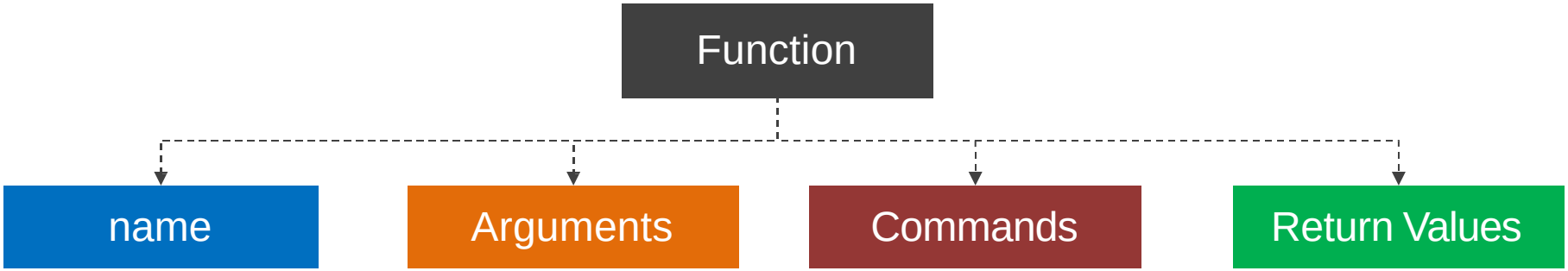
Make your code more generic

Intro



Defining

```
def fnName:
    pass
```




```
def measureTemp ( temp ):
    if temp < 37:
        return "Too Cold"
    elif temp > 37:
        return "Too Hot"
    return "Normal"
```

```
measureTemp(37)
# "Normal"
```

Default Arguments

```
def doSum(x, y = 2,z = 3):
    sum = x + y + z
    print(sum)
```

Calling It

```
doSum(2)            # output: 7
doSum(2,4)          # output: 9
doSum(2,4,10)       # output: 16
```

*arguments

```
def doSum(*args):
        sum = 0
        for i in args:
                sum += i;
        print(sum)
```

Calling It

```
doSum(2,6)              # output: 8

doSum(2,4,5,15)         # output: 26
```

**keywords

```
def doSum(**kwargs):
    for k in kwargs:
        print(kwargs[k])
```

Calling It

```
doSum(x = 2, y = 26)     # output: 2
                                  26
```

Scope

To know your limits

Lexical Scope

Output:

Lexical Scope

```
name = "Ahmed"
```

Output:

Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
```

Output:

Lexical Scope

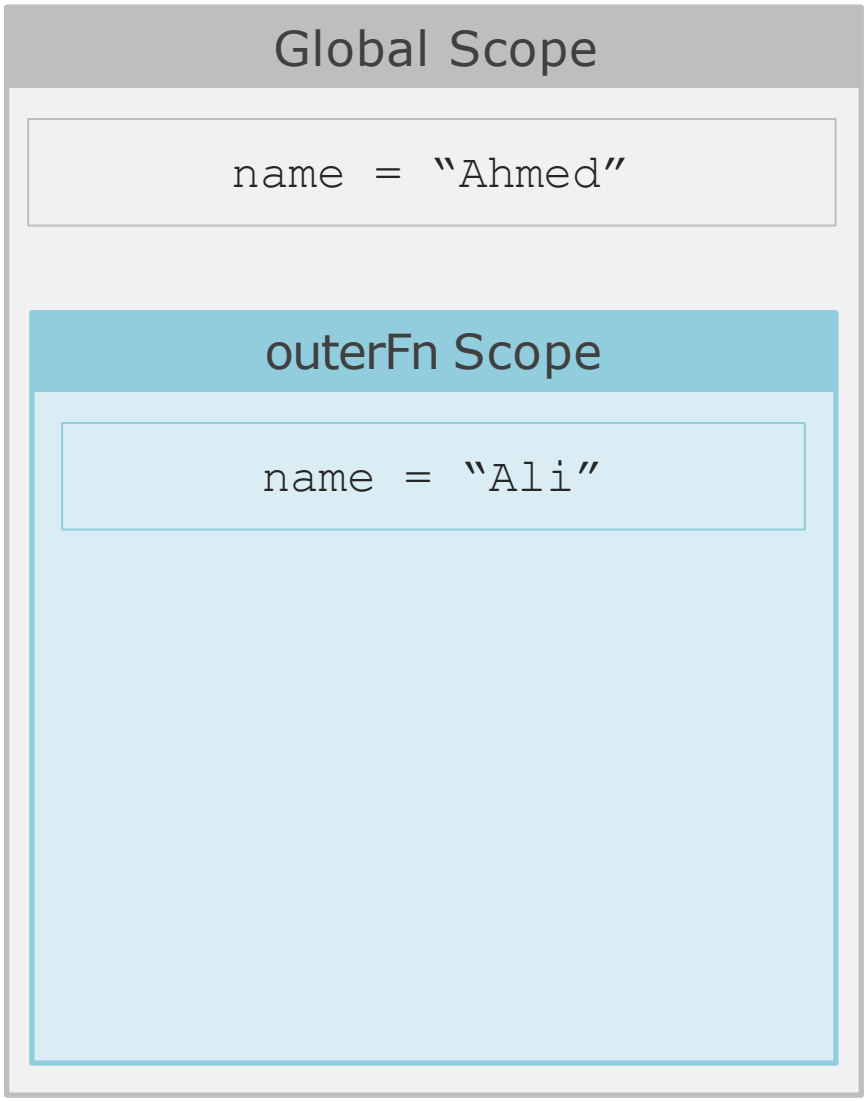
```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

Output:

Lexical Scope

```
name = "Ahmed"
def outerFn():
    → name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

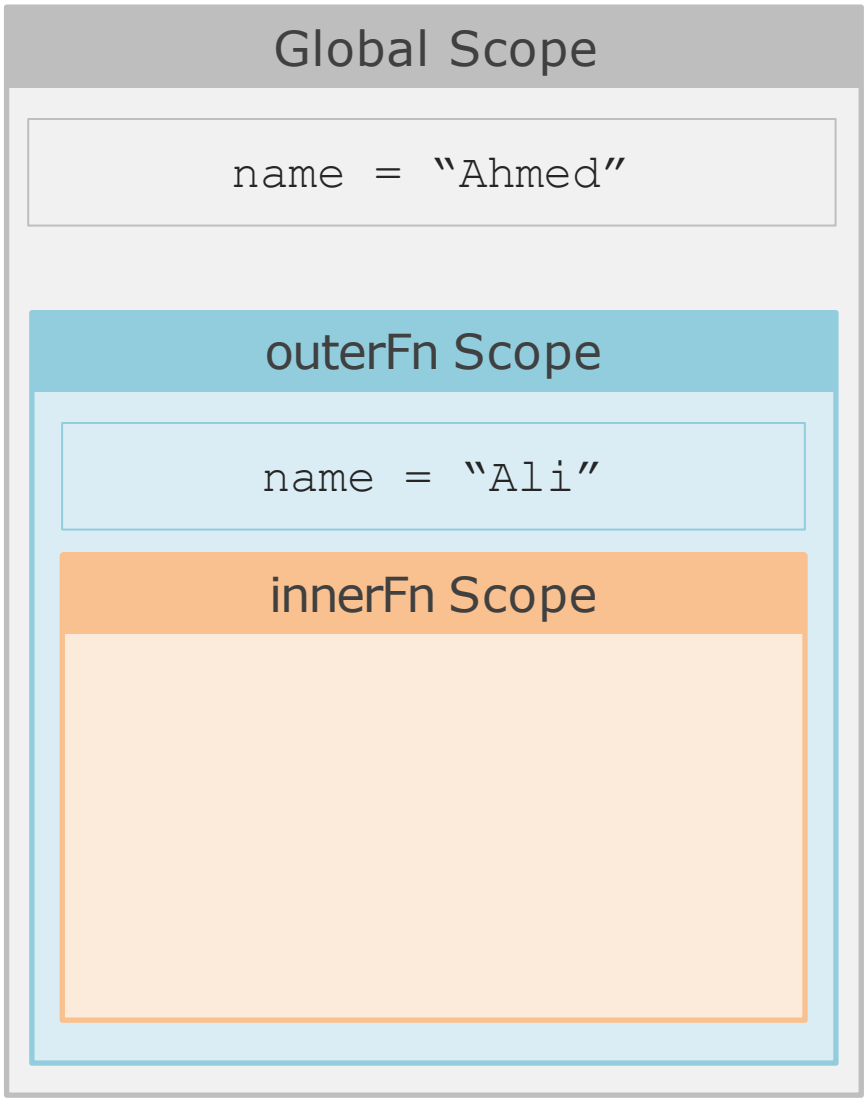
Output:



Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
    →   innerFn()
outerFn()
```

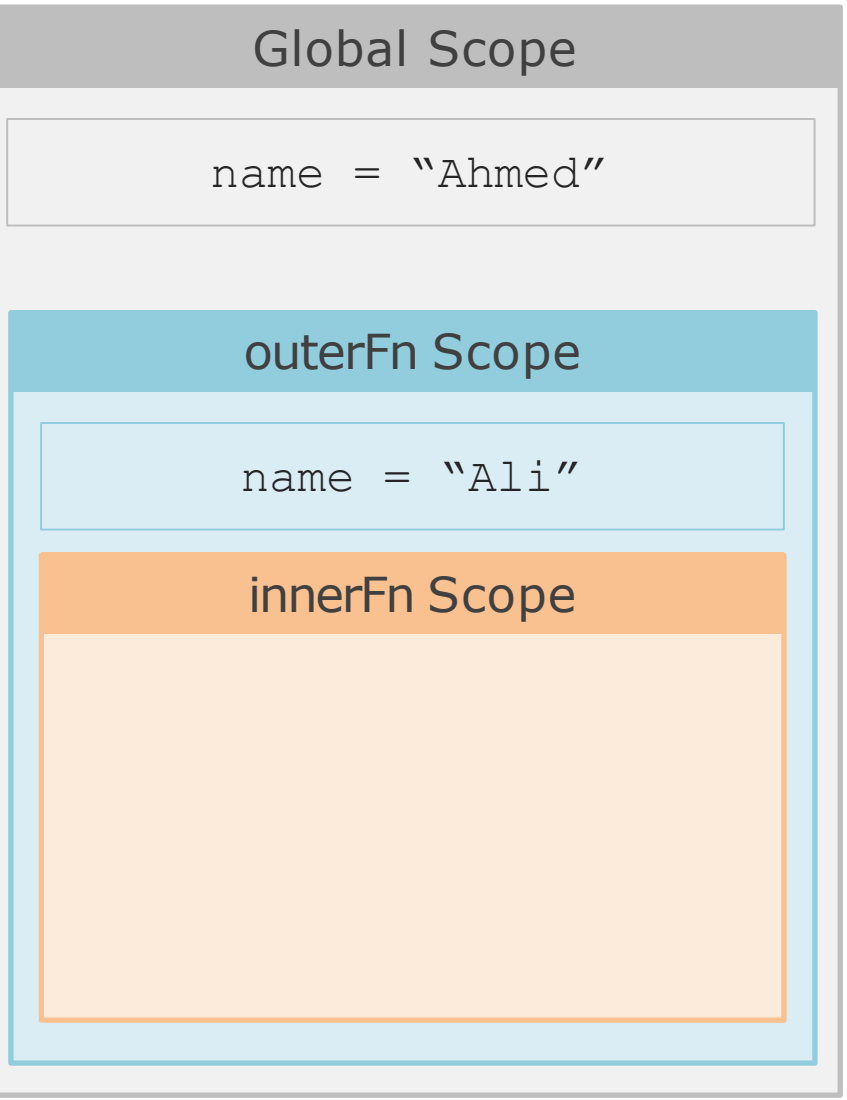
Output:



Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

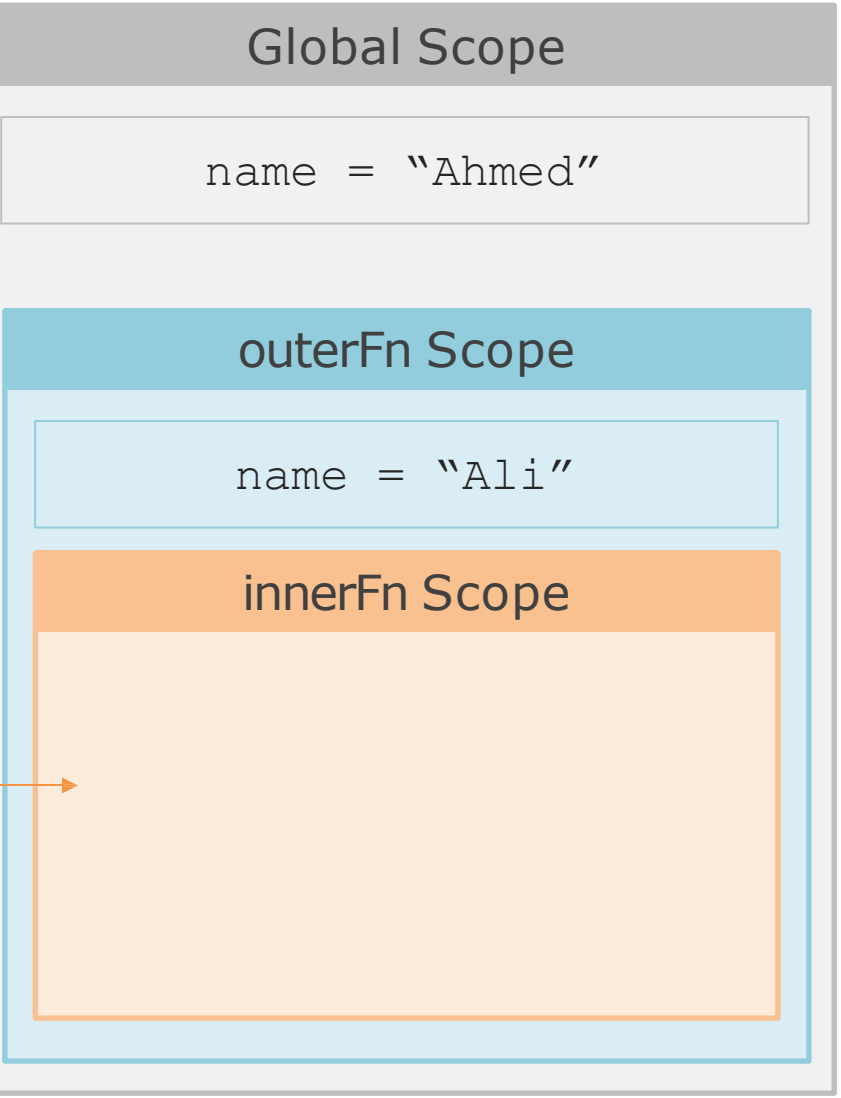


Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

name
???

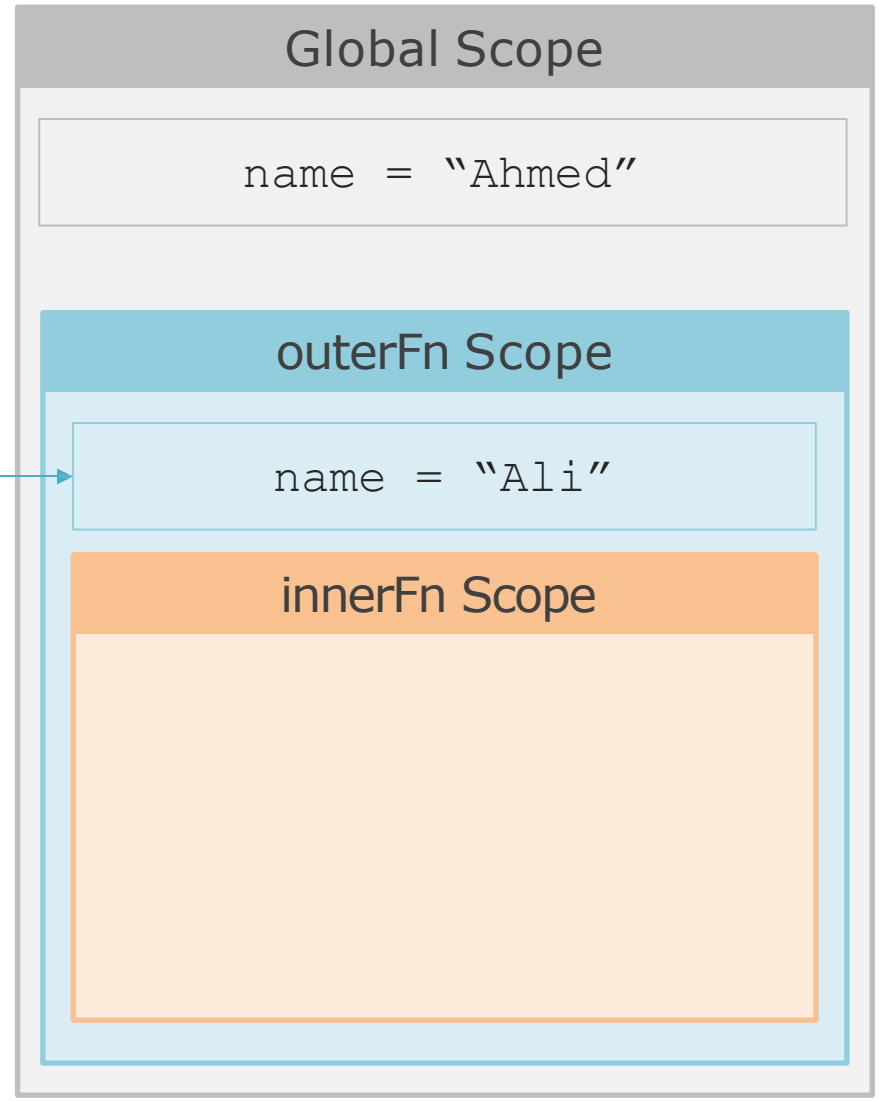


Lexical Scope

```
name = "Ahmed"
def outerFn():
    name = "Ali"
    def innerFn():
        print(name)
    innerFn()
outerFn()
```

Output:

name
???



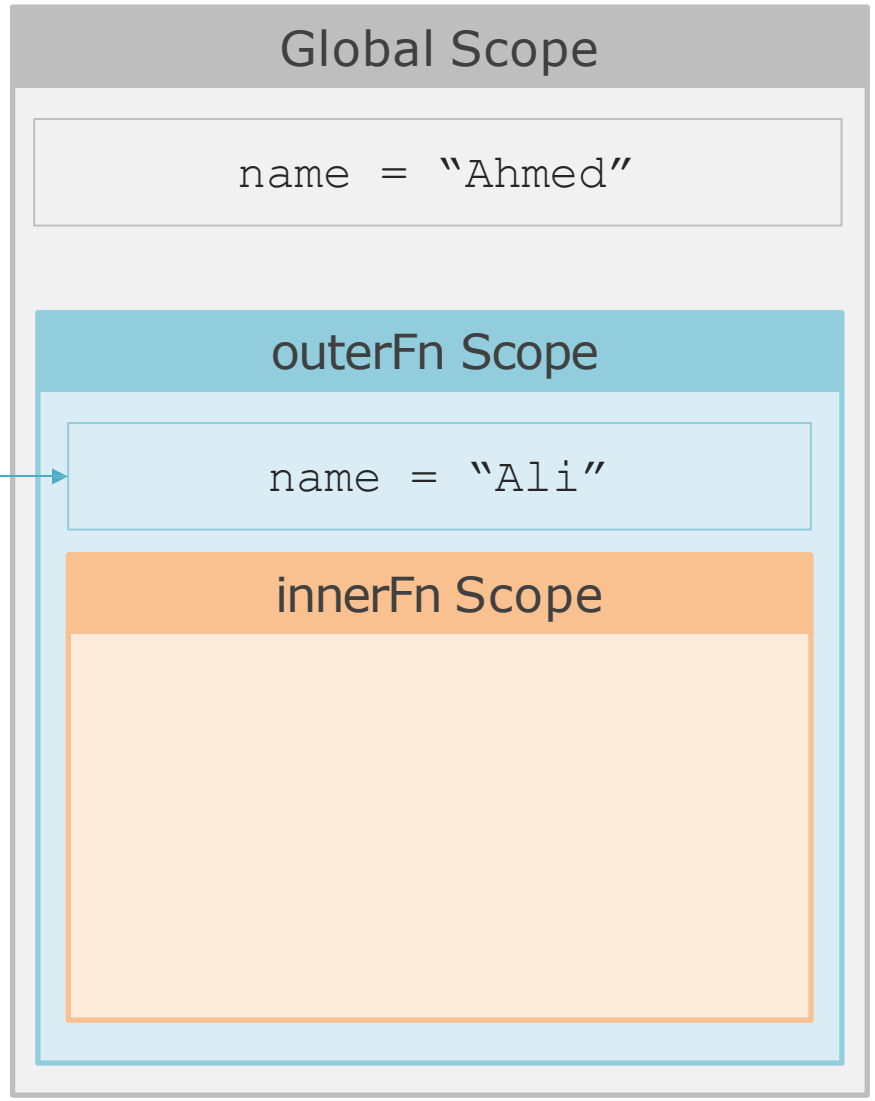
Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

```
Ali
```

name
???



Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
print(name)
```

Output:

```
Ali
```

Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
print(name)
```

Output:

```
Ali
```

name
???

Lexical Scope

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
print(name)
```

Output:

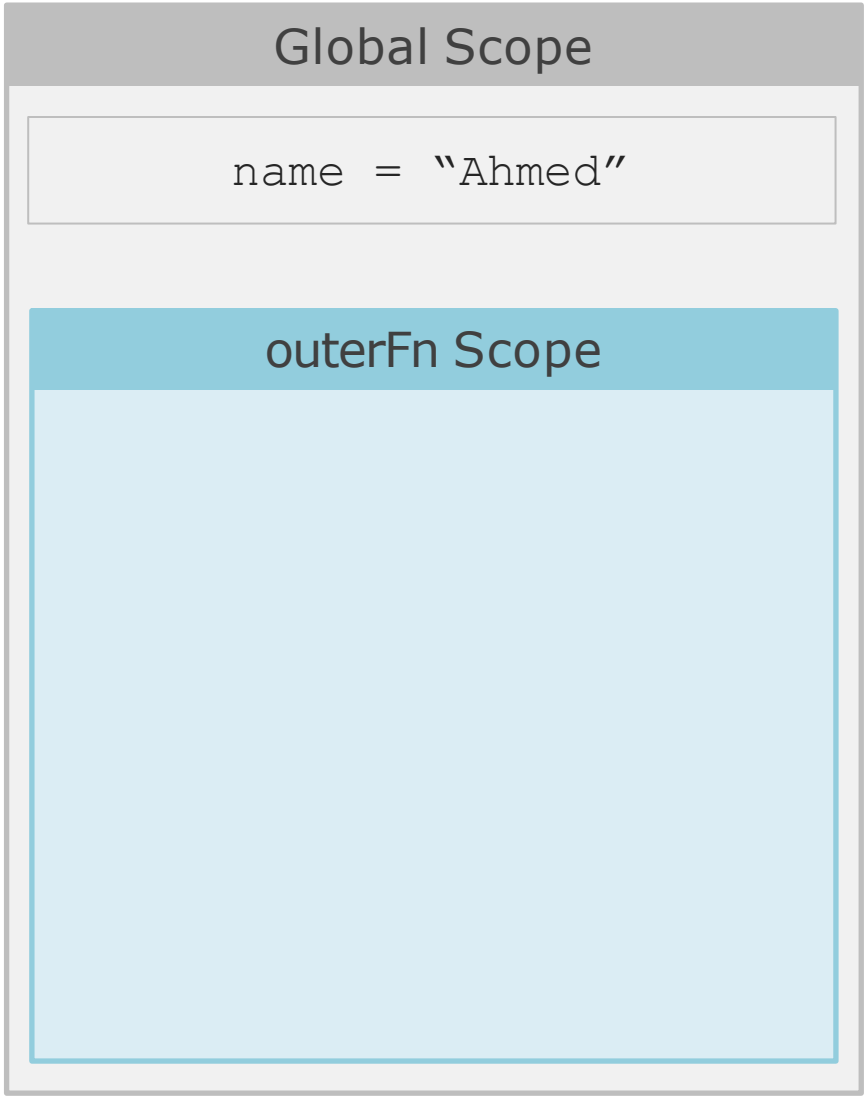
```
Ali
Ahmed
```

name
???

global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

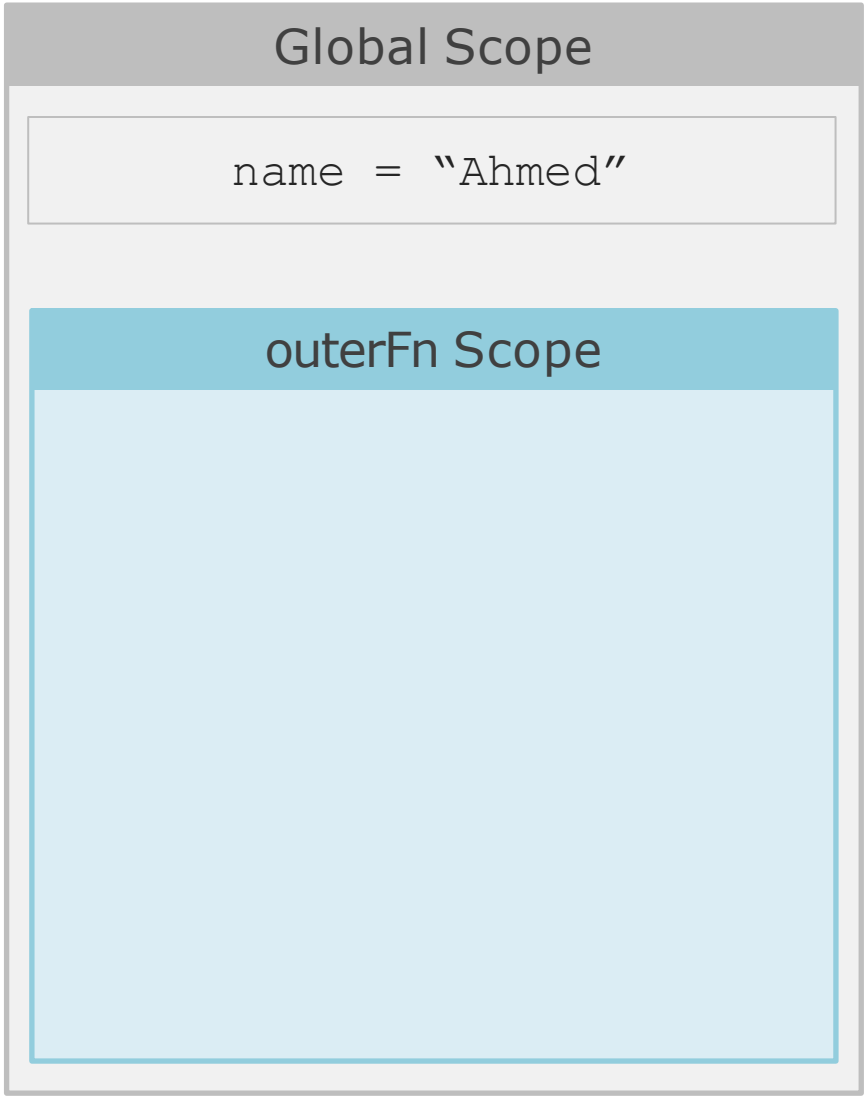
Output:



global Keyword

```
name = "Ahmed"
def outerFn():
    global name
    name = "Ali"
    def innerFn():
        print(name)
    innerFn()
outerFn()
```

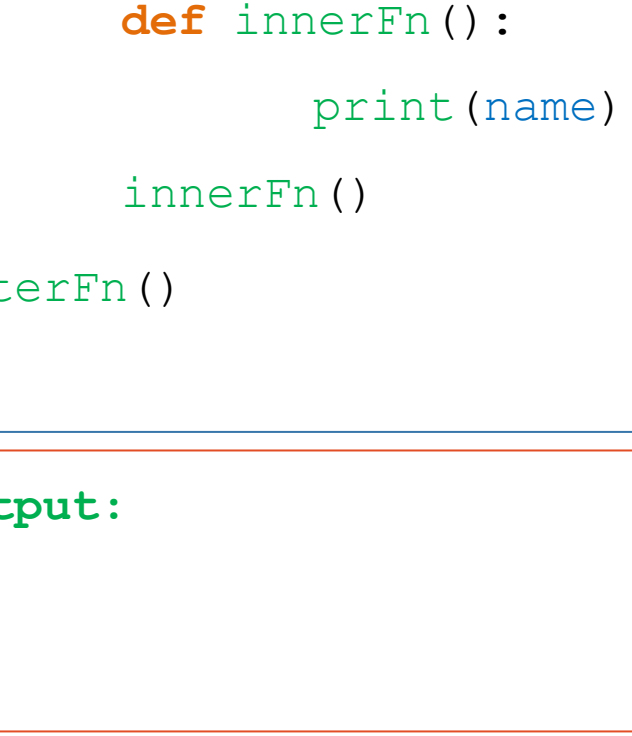
Output:



global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

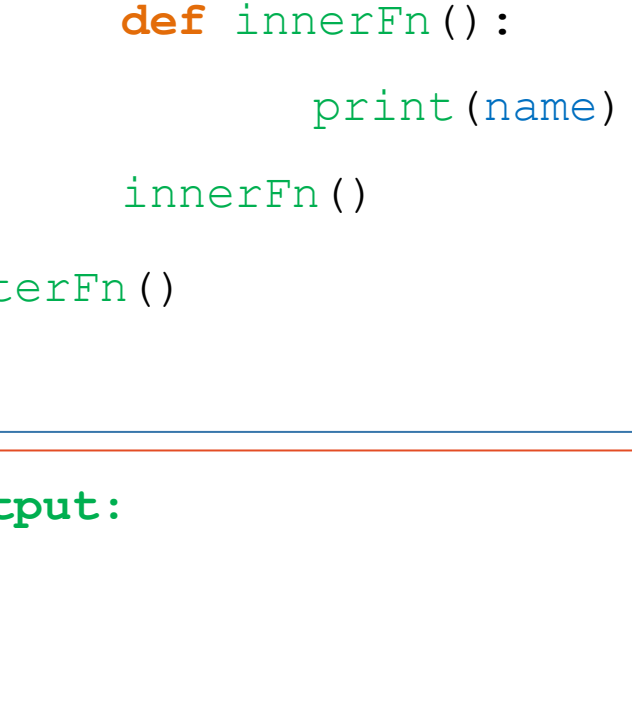
Output:



global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

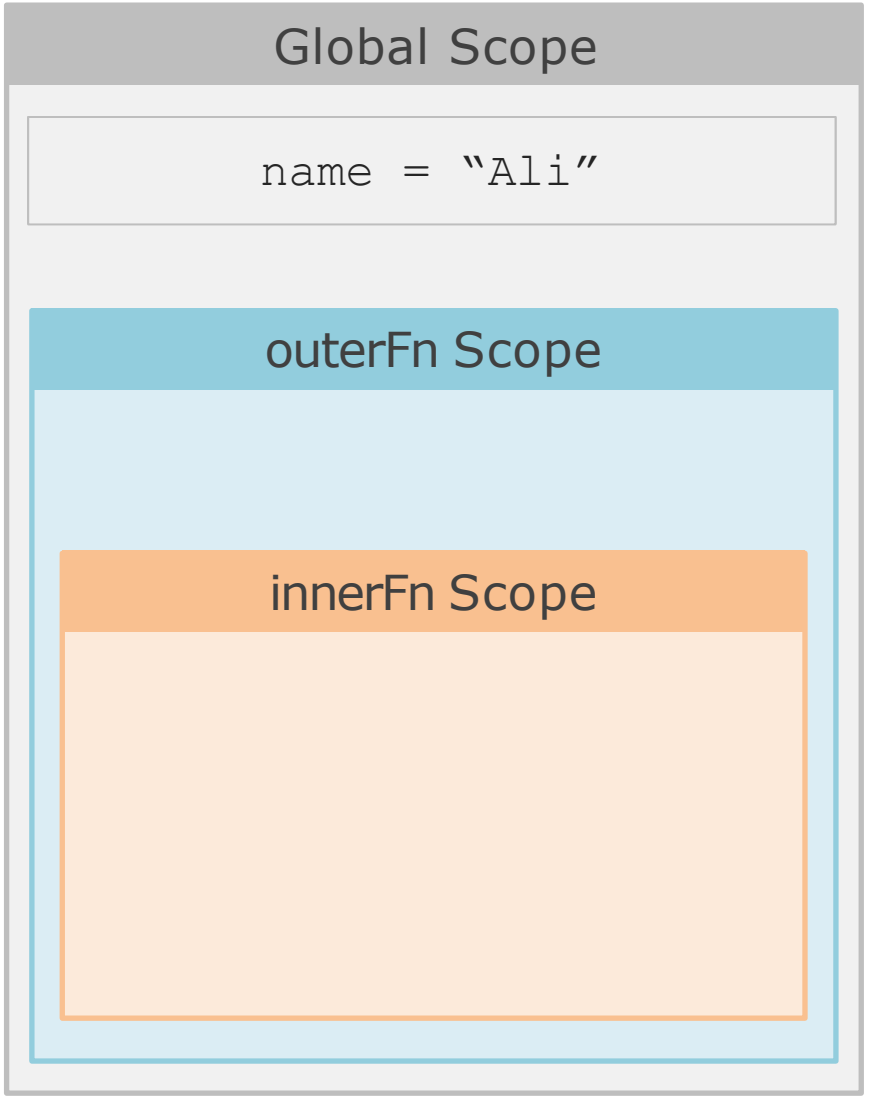
Output:



global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
    →   innerFn()
outerFn()
```

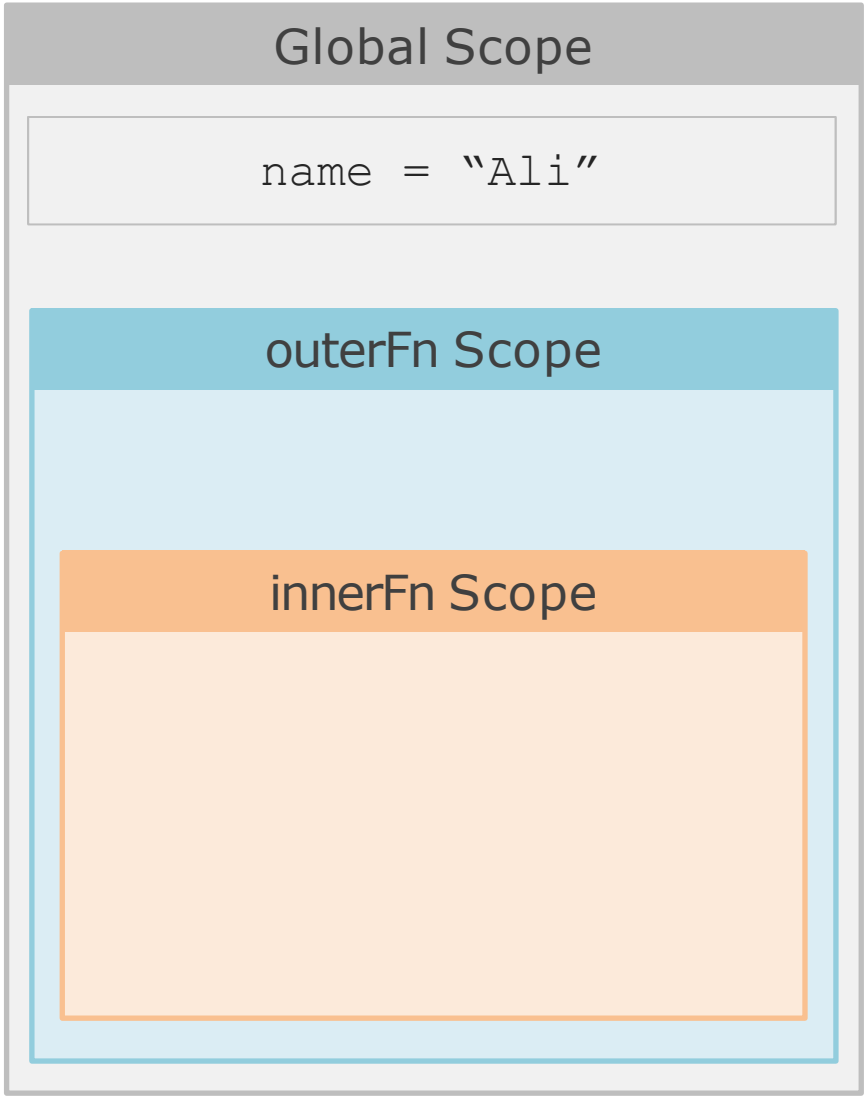
Output:



global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

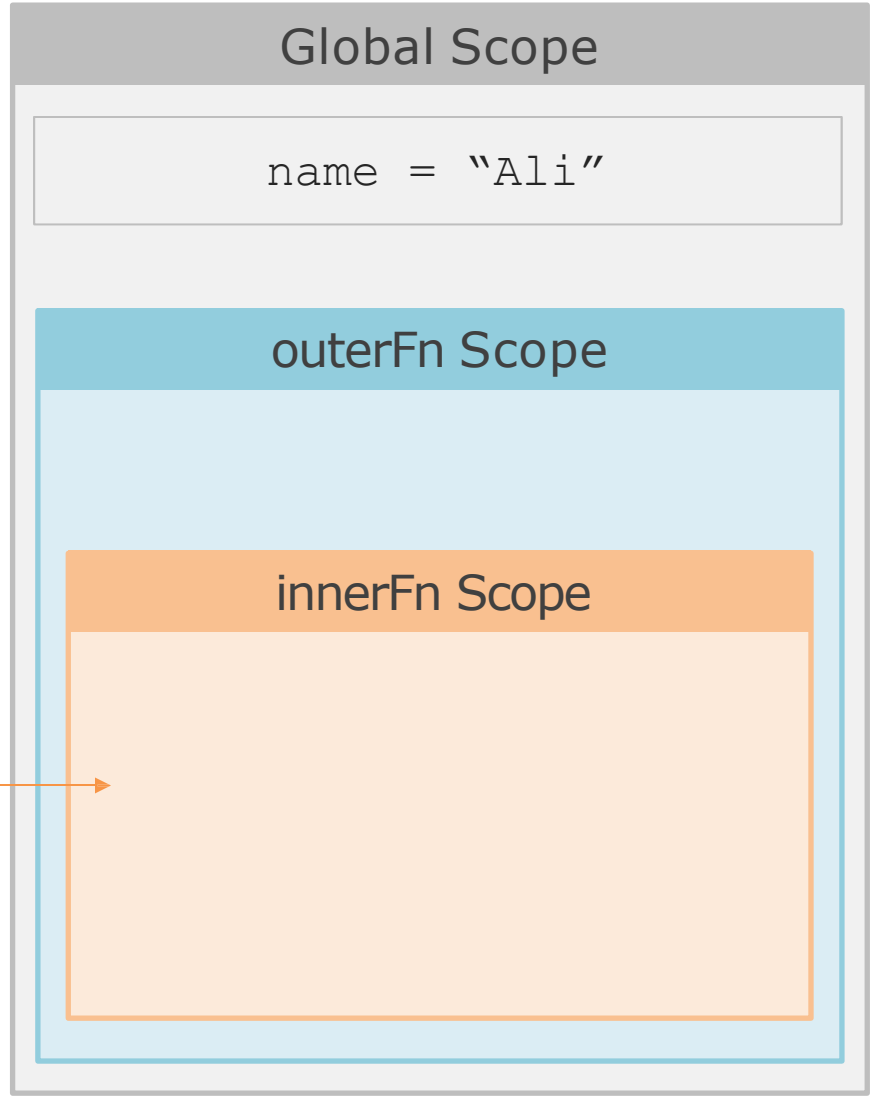


global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

name
???

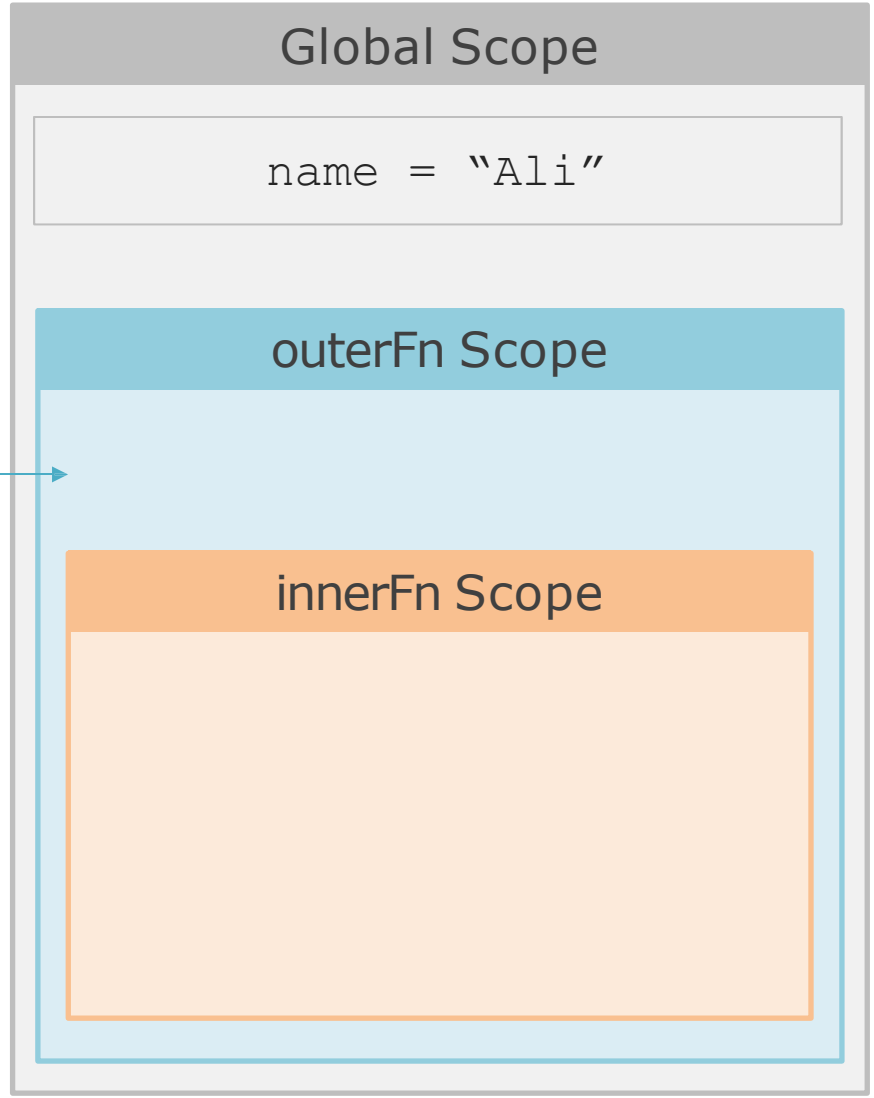


global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

name
???

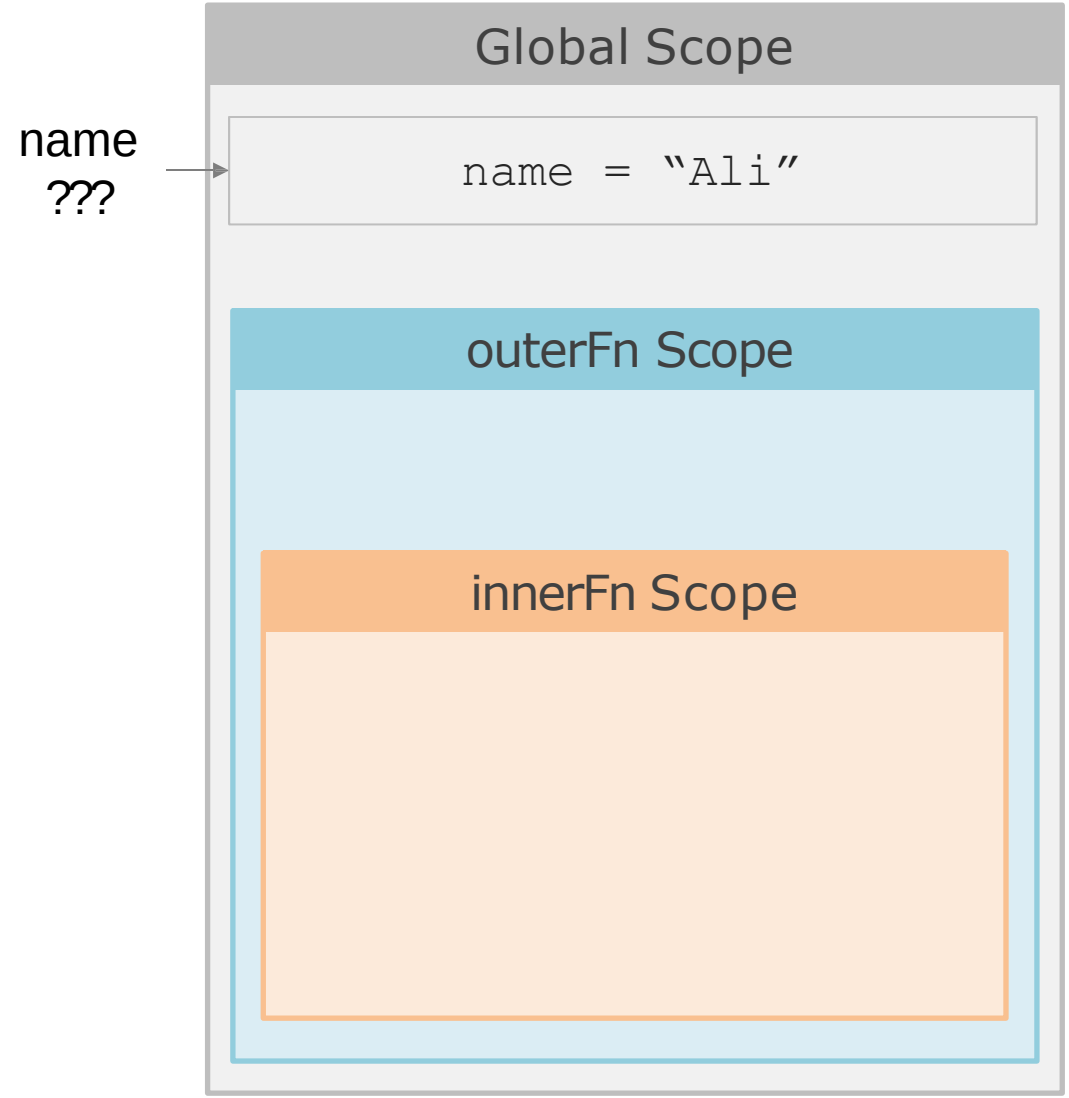


global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

name
???

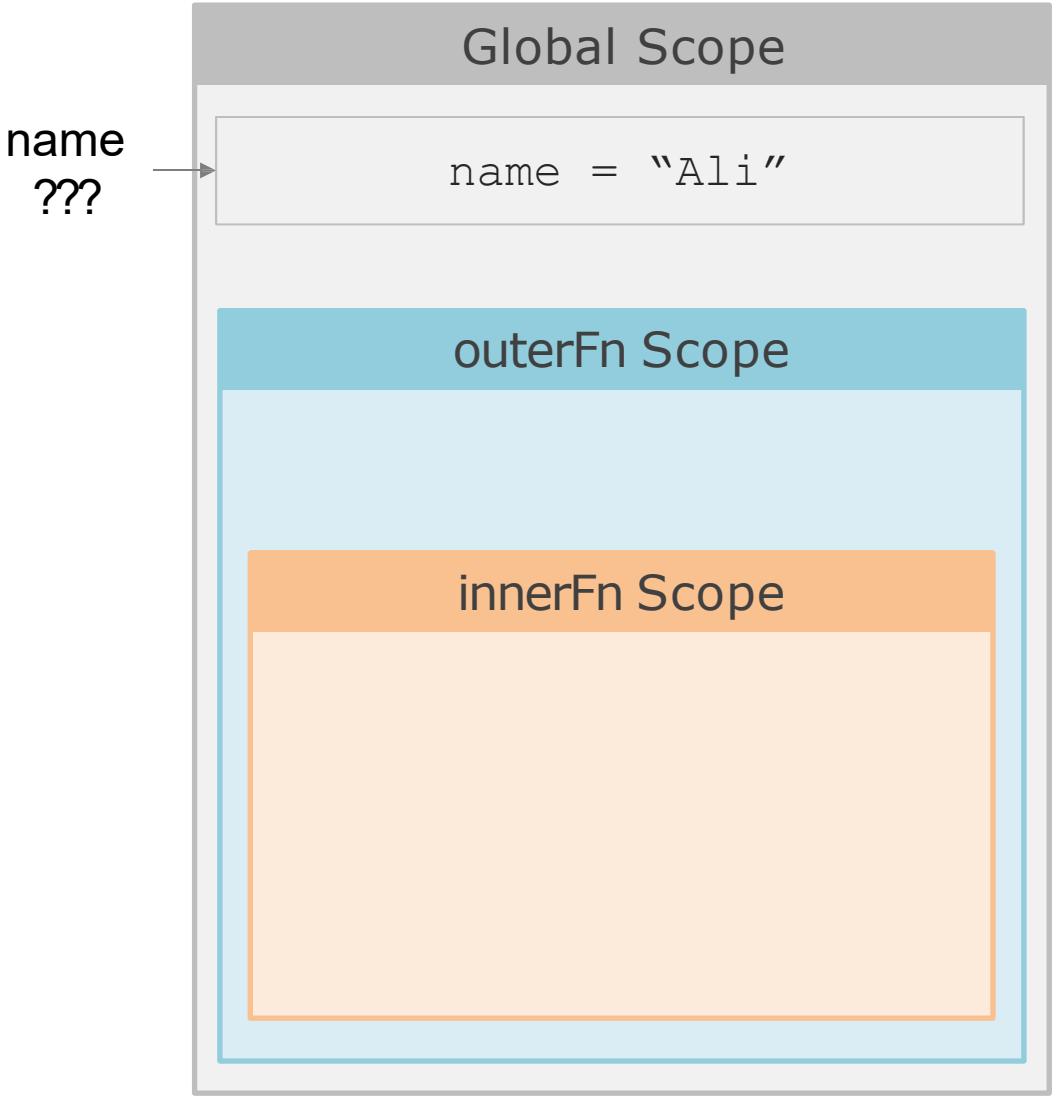


global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
            print(name)
        innerFn()
outerFn()
```

Output:

```
Ali
```



global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

Output:

```
Ali
```

name
???

global Keyword

```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
print(name)
```

Output:

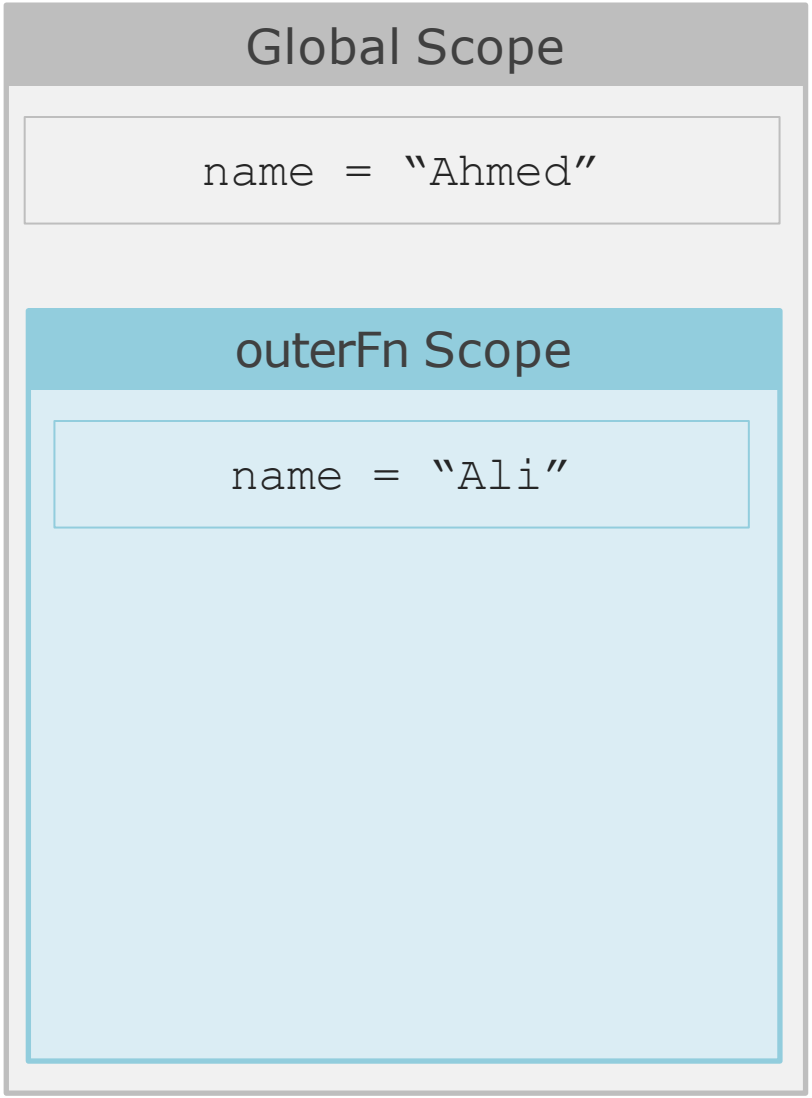
```
Ali
Ali
```

name
???

nonlocal Keyword

```
name = "Ahmed"
def outerFn():
    name = "Ali"
    def innerFn():
        nonlocal name
        print(name)
        name = "Sara"
    innerFn()
    print(name)
outerFn()
```

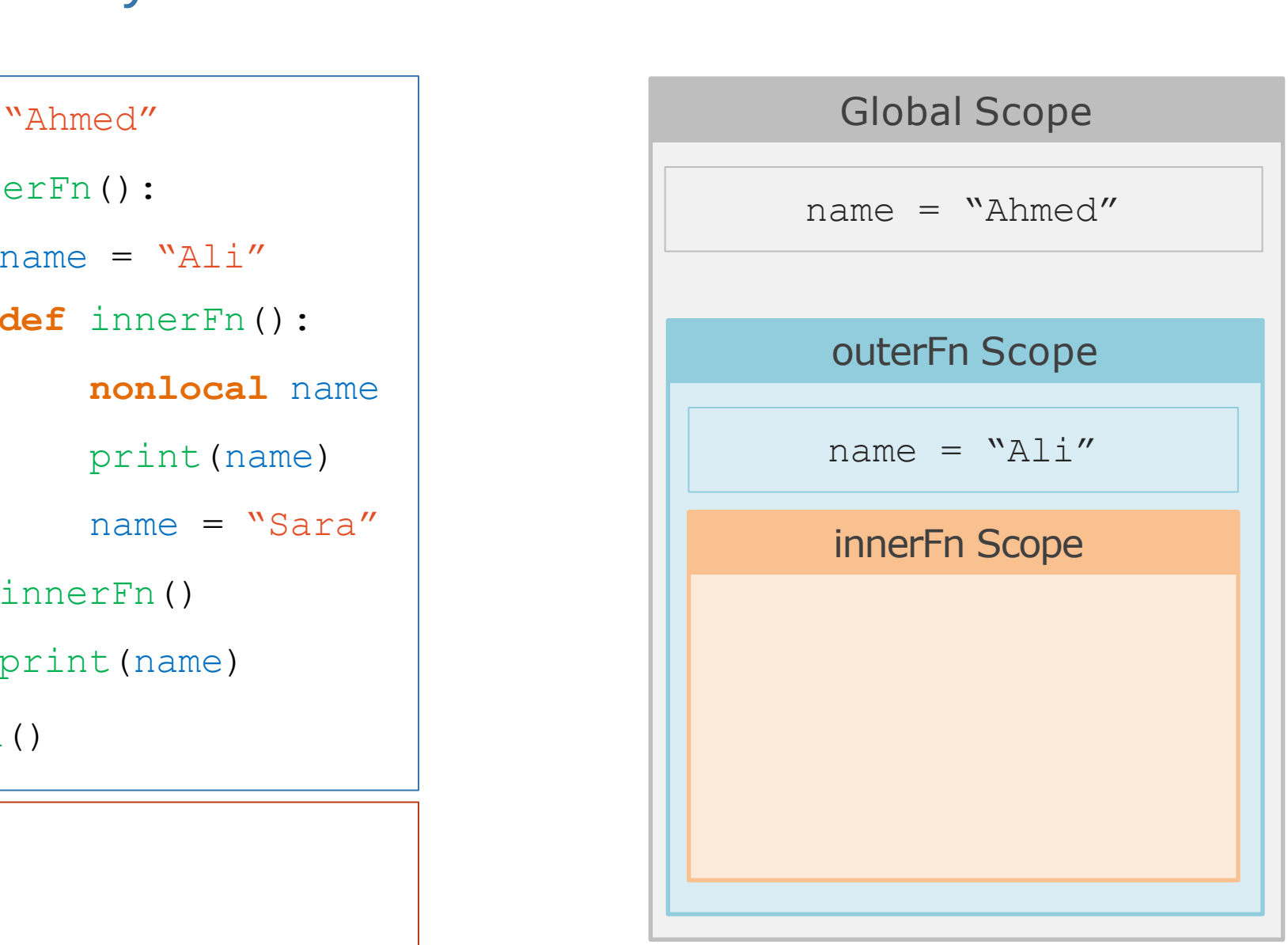
Output:



nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
    →   innerFn()
        print(name)
outerFn()
```

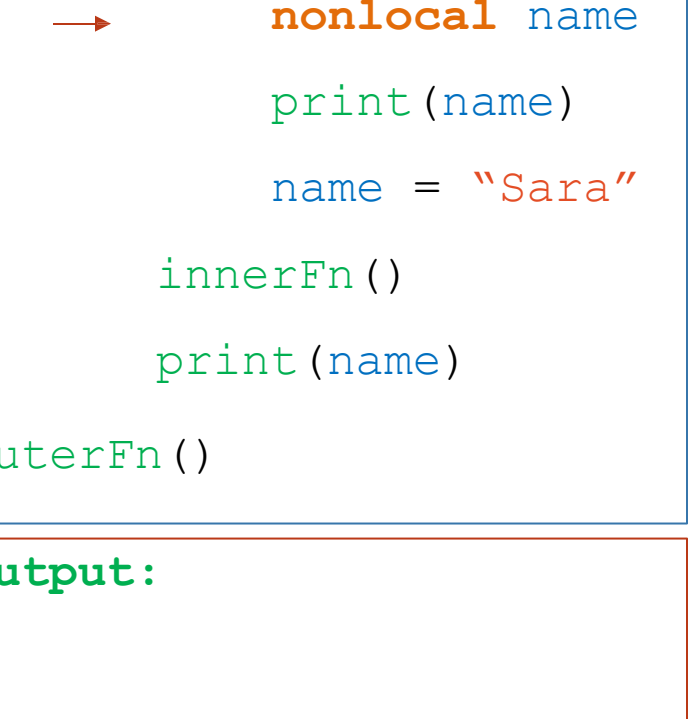
Output:



nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

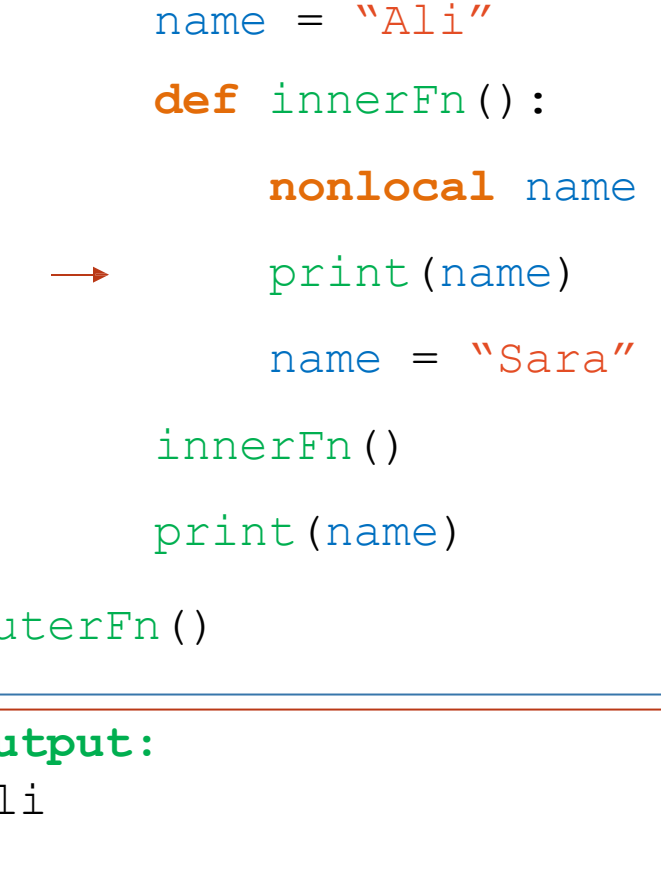


nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

```
Ali
```

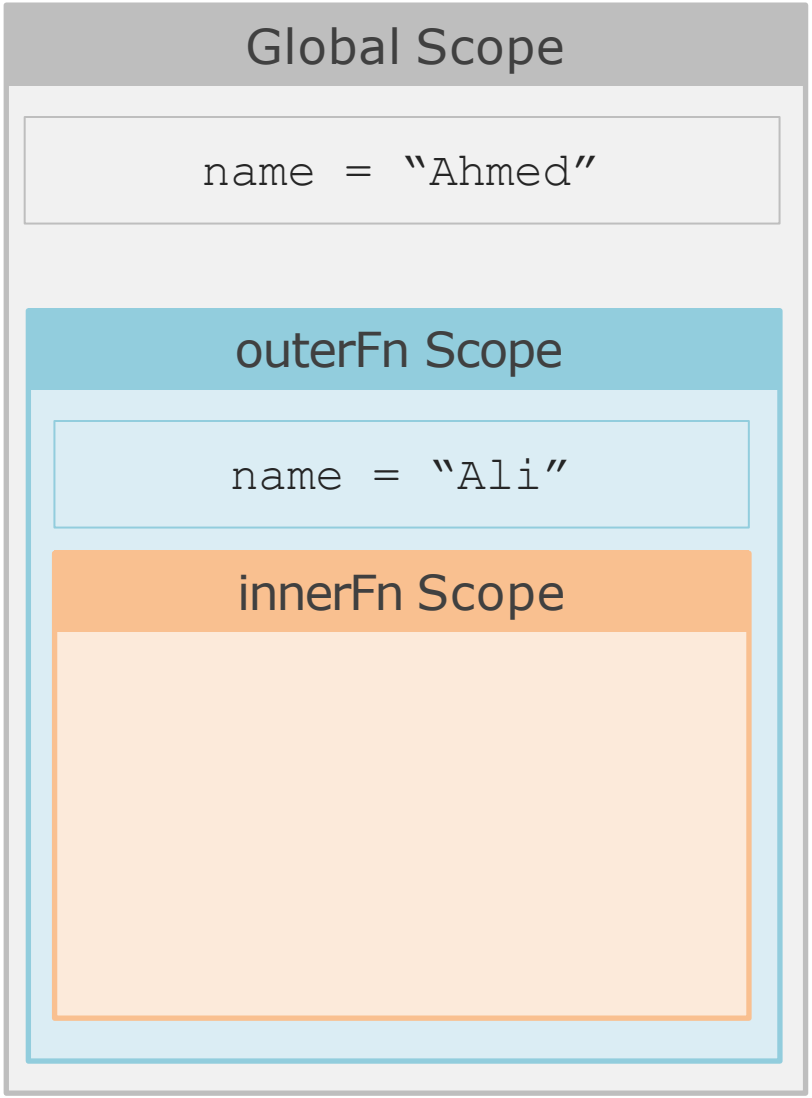


nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

```
Ali
```

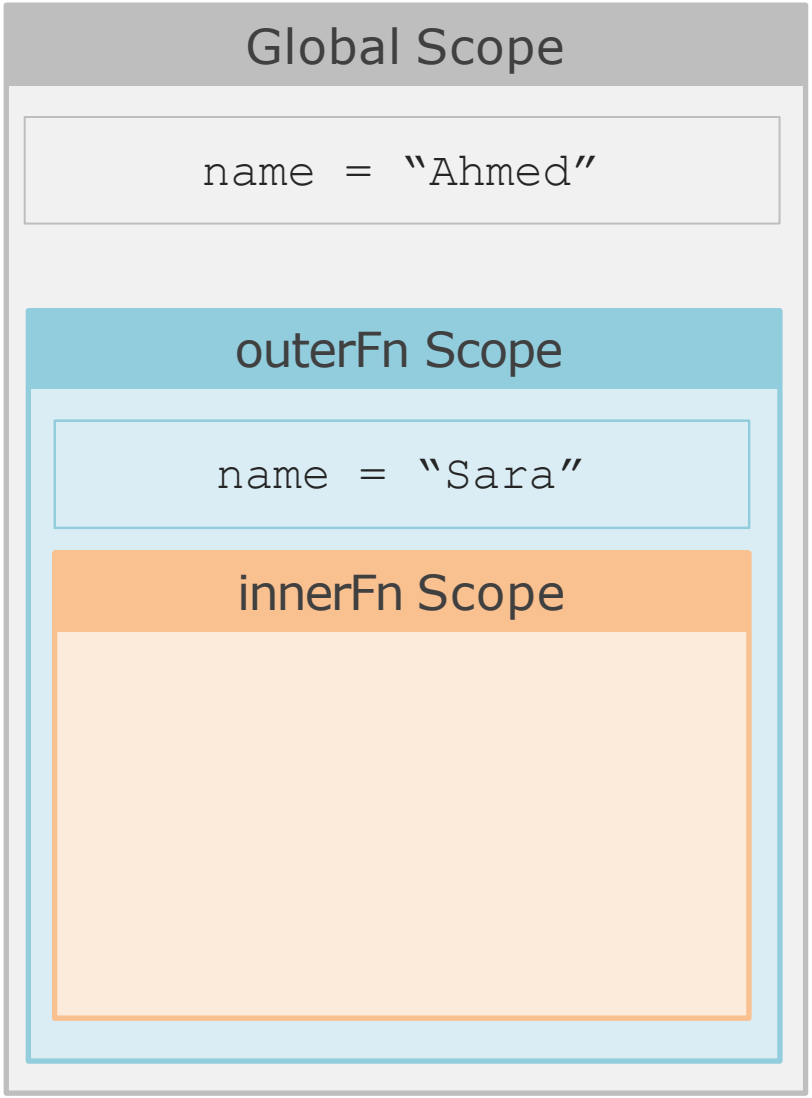


nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

```
Ali
```

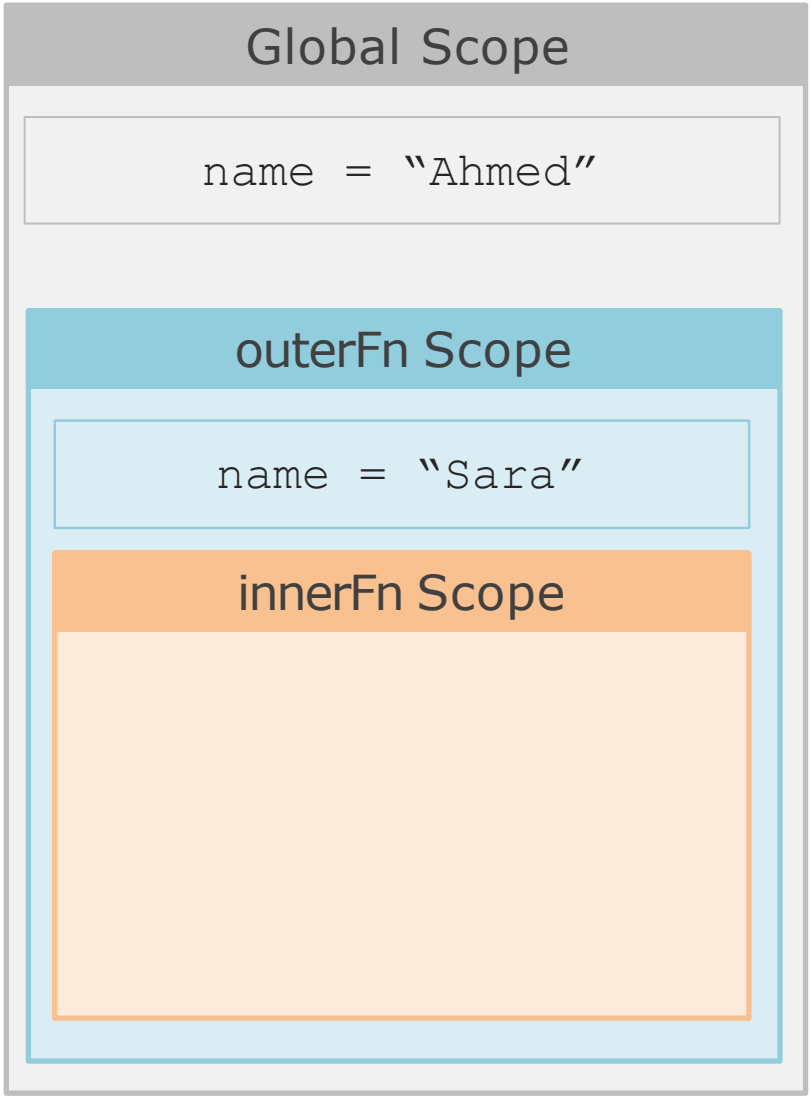


nonlocal Keyword

```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

```
Ali
```

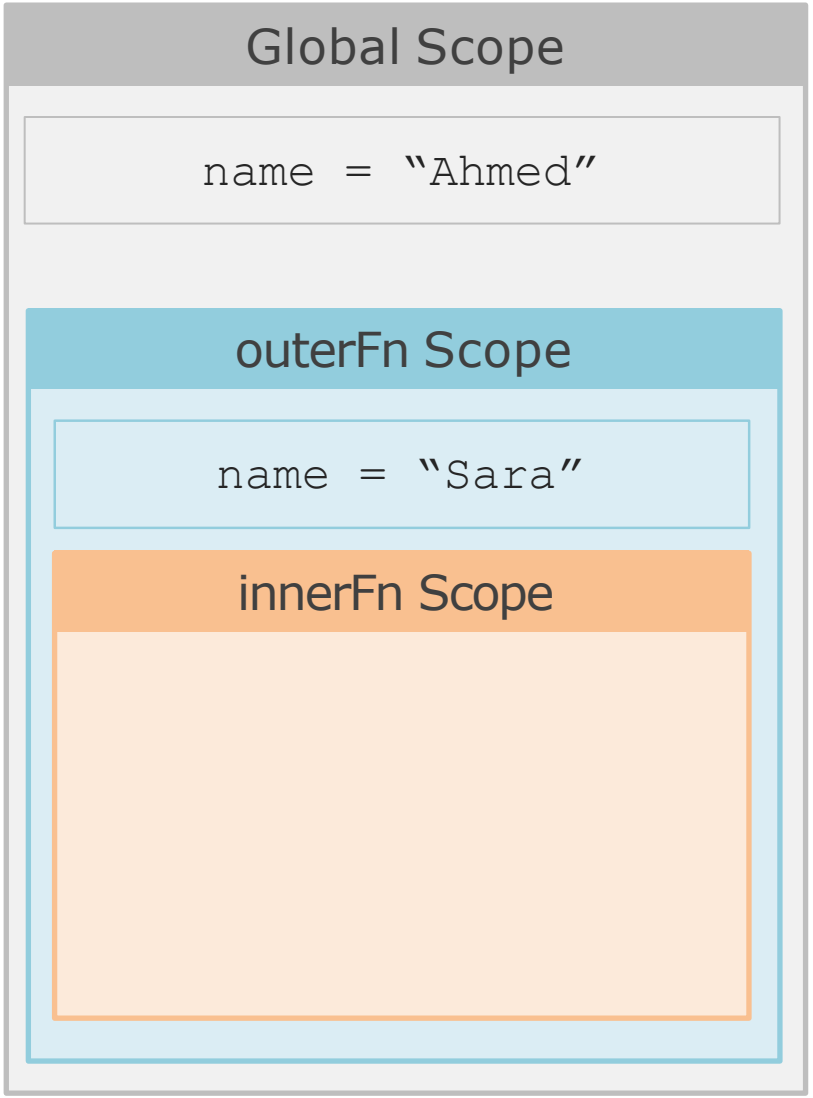


nonlocal Keyword

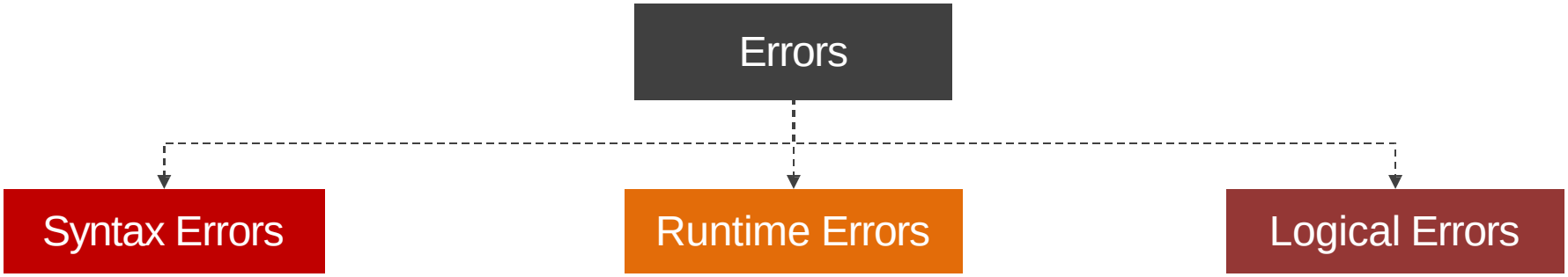
```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
            nonlocal name
            print(name)
            name = "Sara"
        innerFn()
        print(name)
outerFn()
```

Output:

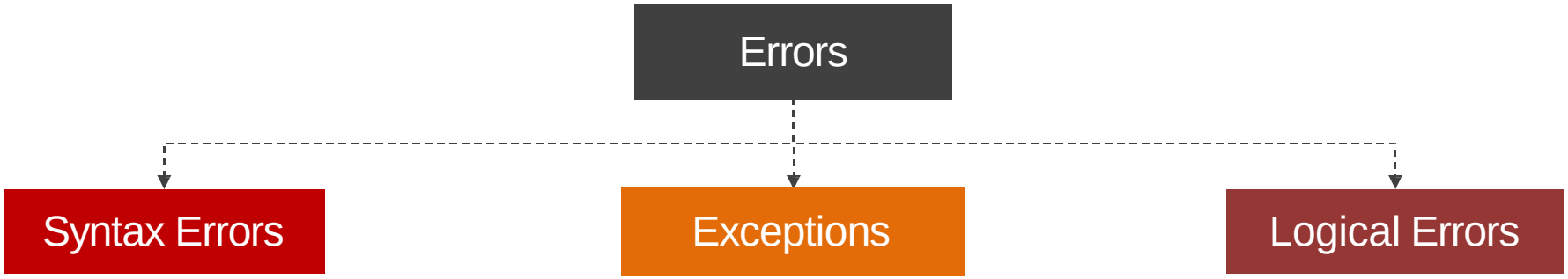
```
Ali
Sara
```



Intro



Intro



Syntax Errors

Errors that will show up if you doesn't follow Python Syntax Rules

```
print("You missed the closing round braces "
```

```
print("You missed the closing round braces "
                                            ^
SyntaxError: invalid syntax
```

Exceptions

Errors detected during execution are called **Exceptions**

```
print(firstname);
```

```
NameError: name 'firstname' is not defined
```

Handling Exceptions

```
try:
    doTry()
except:
    doExcept()
finally:
    doFinally()
```

Put the code that you want to handle its exceptions

Handle the exception if it raised in the try clause

Put the code that you want to run always if there is an exception or not.

Handling Exceptions

```
try:
        for i in range(3):
            print(3/i)
except ValueError:
        print(“Value Error")
except ZeroDivisionError:
        print("you divided by 0")
finally:
        print("this will print no matter what")
```

Raising Exceptions

```
raise ErrorName(error_message)
```

i.e. raise NameError(“It’s Not a name”)

File Input & Output

File Authoring

Open Files

```
open(file_name, mode)
```

mode	Job description
r	Open Files for reading only
w	Open Files for writing only *
a	Open Files for appending *
r+	Open Files for reading and writing *
rb	Open Files for reading binary files
rb+	Open Files for reading and writing binary files *

* If the file not exist , It will create it.

Read Files

```
fl = open("some_file.txt", 'r')
fl.read()
#output: Some text on line 1.
         Other text on line 2.
fl.read(4)
#output: Some
fl.readline()
#output:  text on line 1.
fl = open("some_file.txt", 'r')
for line in f1:
       print(line)
#output: Some text on line 1.
         Other text on line 2.
```

some_file.txt

Some text on line1.

Other text on line2.

Write on Files

```
fl = open("some_file.txt", 'w')
```

some_file.txt

Some text on line 1.

Other text on line 2.

Write on Files

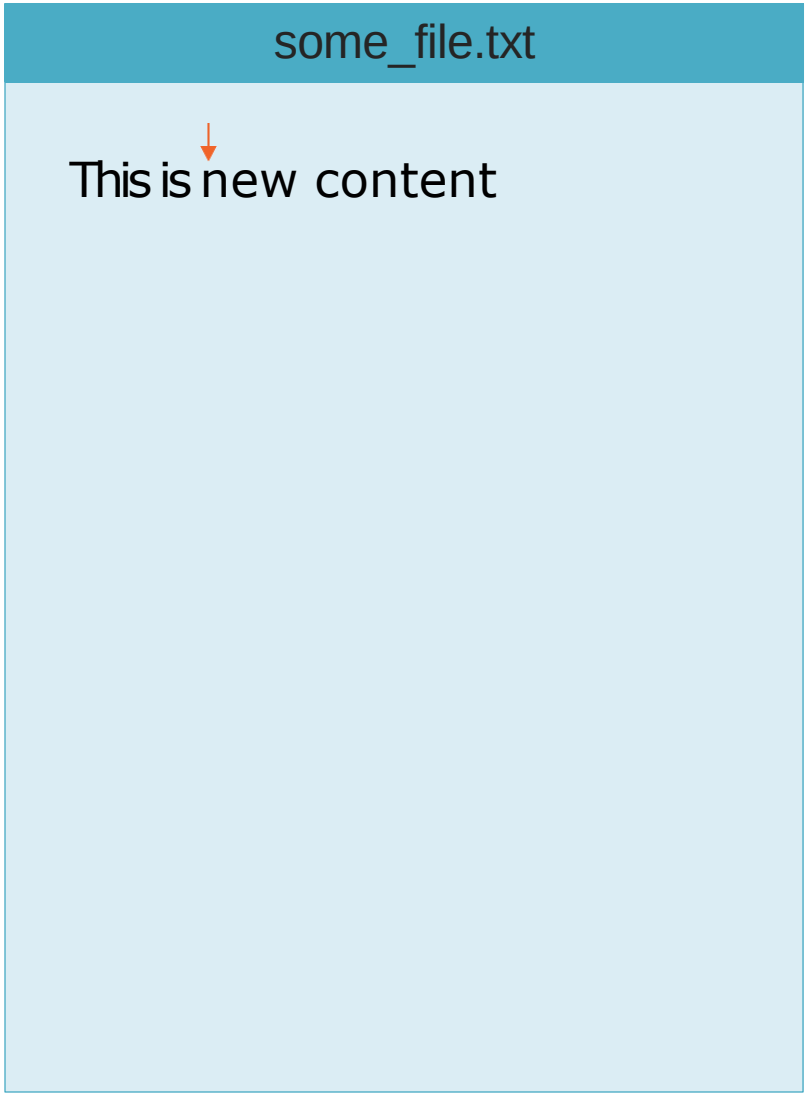
```
fl = open("some_file.txt", 'w')
fl.write("This is new content")
```

some_file.txt

This is new content

Write on Files

```
fl = open("some_file.txt", 'w')
fl.write("This is new content")
fl.seek(8)
```



Write on Files

```
fl = open("some_file.txt", 'w')
fl.write("This is new content")
fl.seek(8)
fl.write("old")
```

some_file.txt

This is old content

Write on Files

```
fl = open("some_file.txt", 'w')
fl.write("This is new content")
fl.seek(8)
fl.write("old")
fl.close()
fl = open("some_file.txt", 'a')
fl.write("\n content is appended")
```

some_file.txt

This is old content

content isappended