Python: The Easy Way

Lecture 2

Scope

To know your limits

Lexical Scope

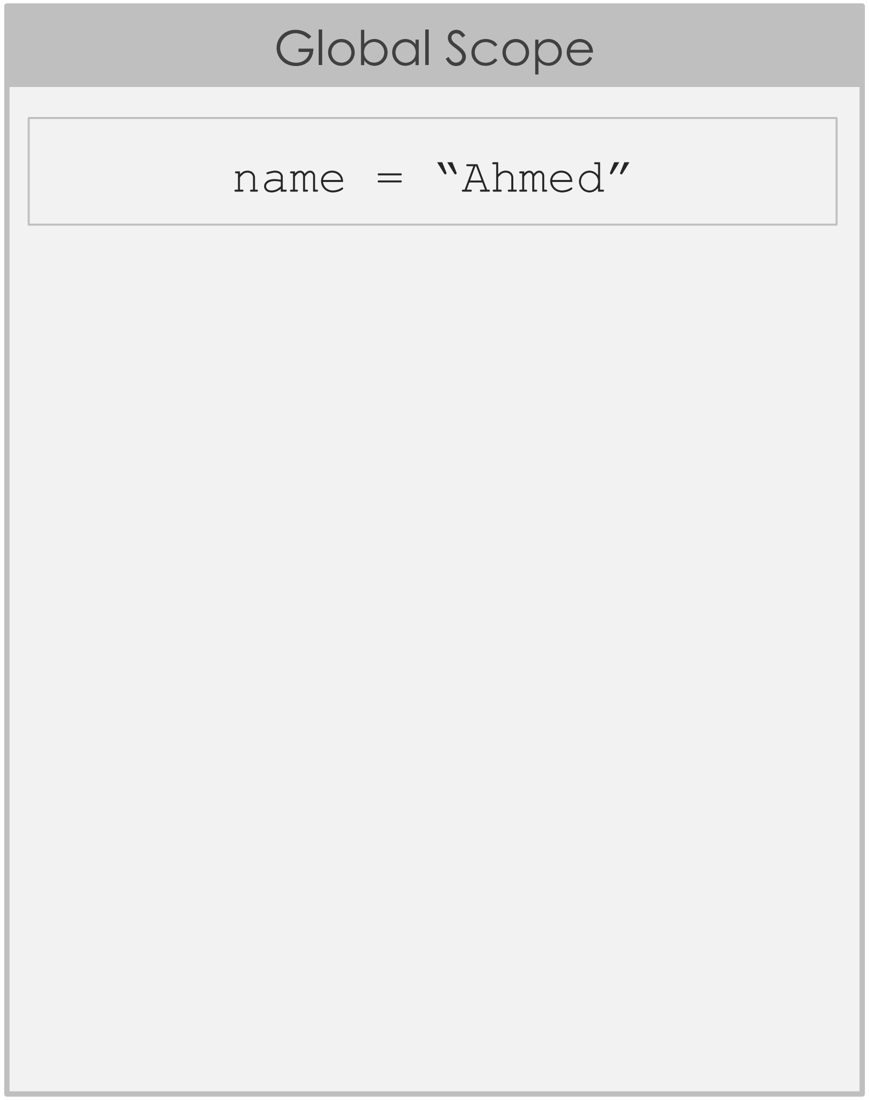
Global Scope

Output:

Lexical Scope

```
name = “Ahmed”
```

```
Output:
```

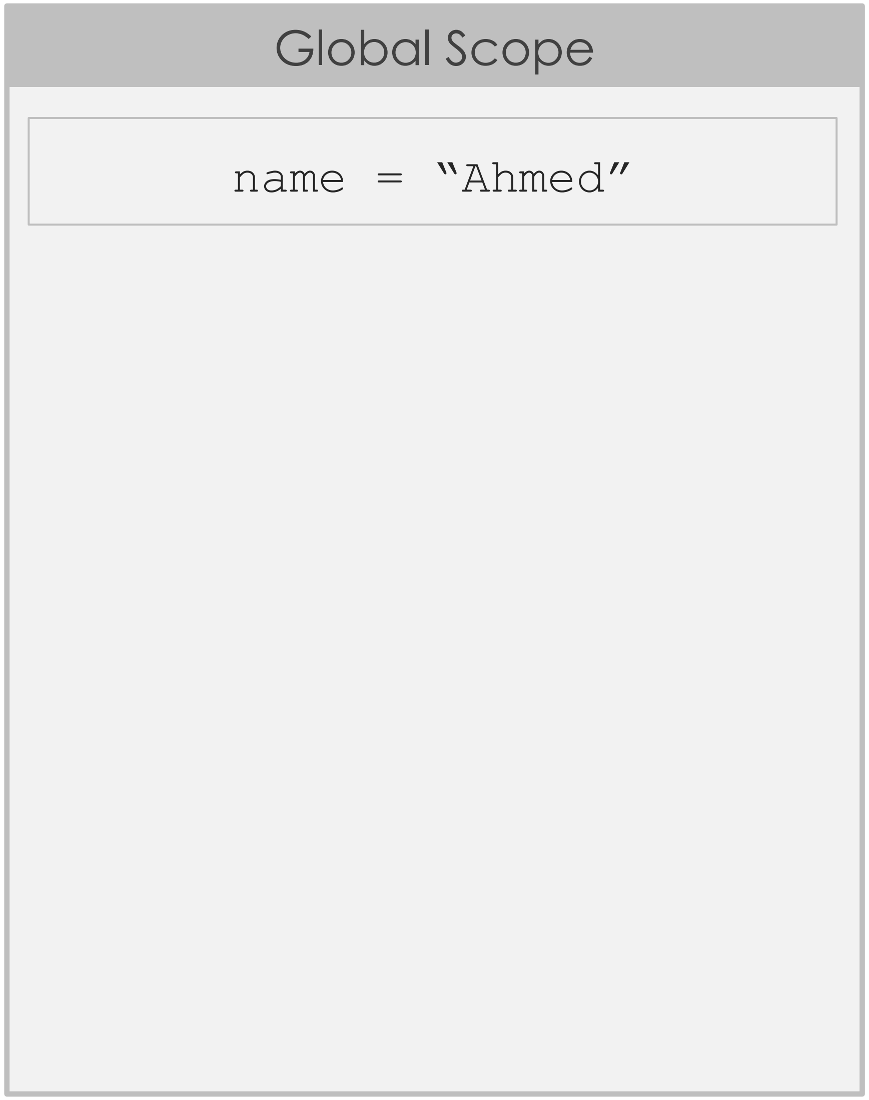


```
name = “Ahmed”
```

Lexical Scope

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

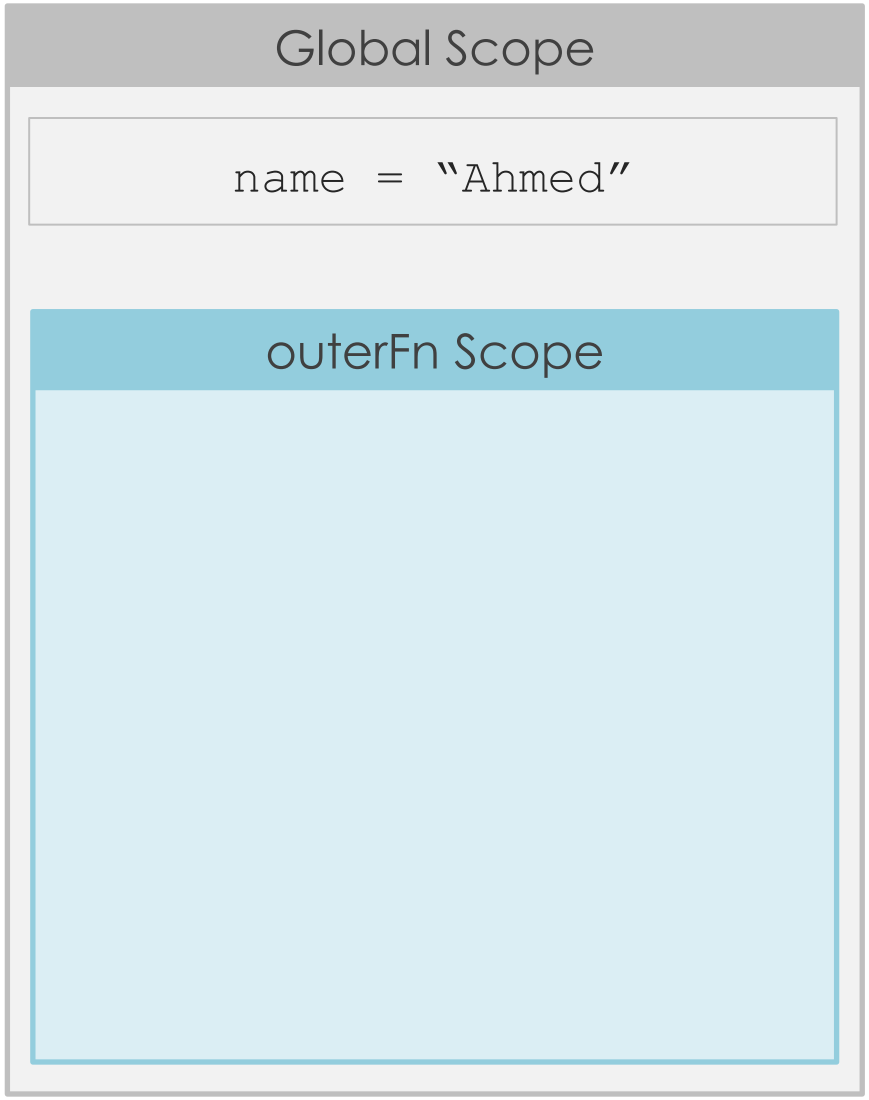
Output:



Lexical Scope

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

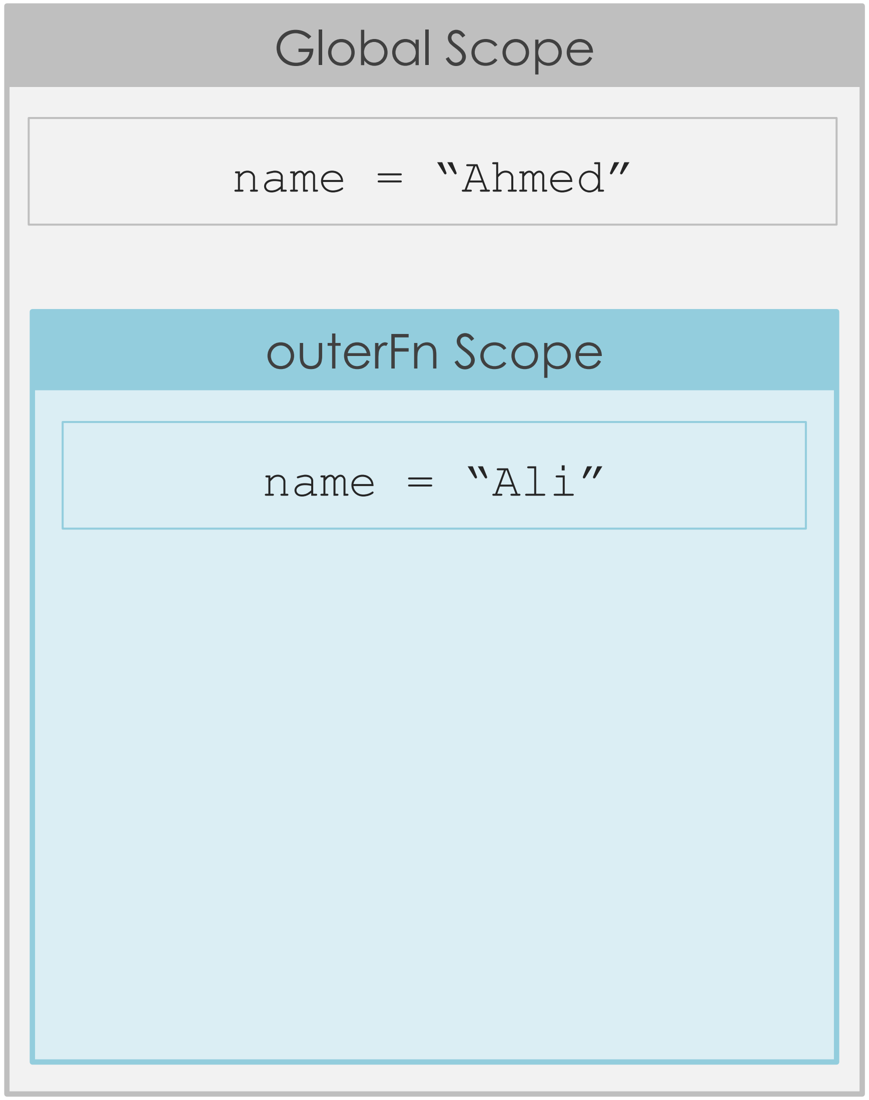
Output:



Lexical Scope

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

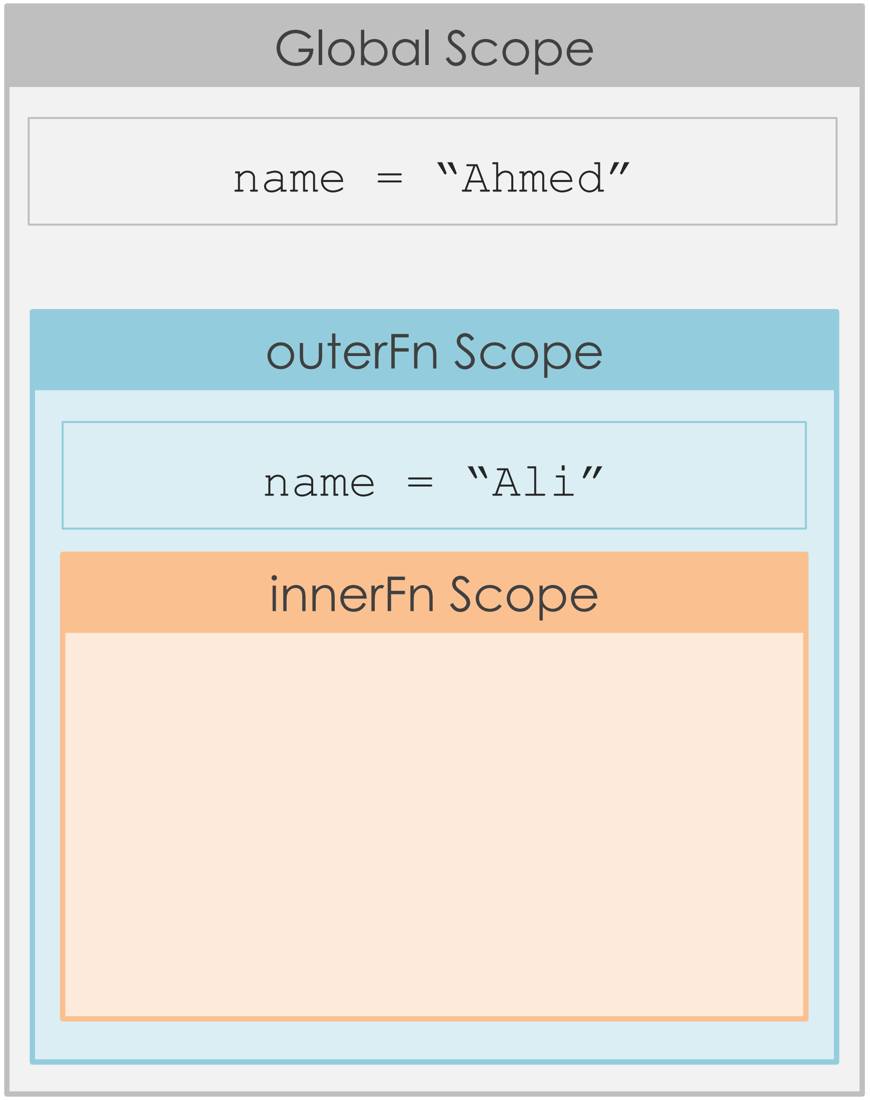
Output:



Lexical Scope

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

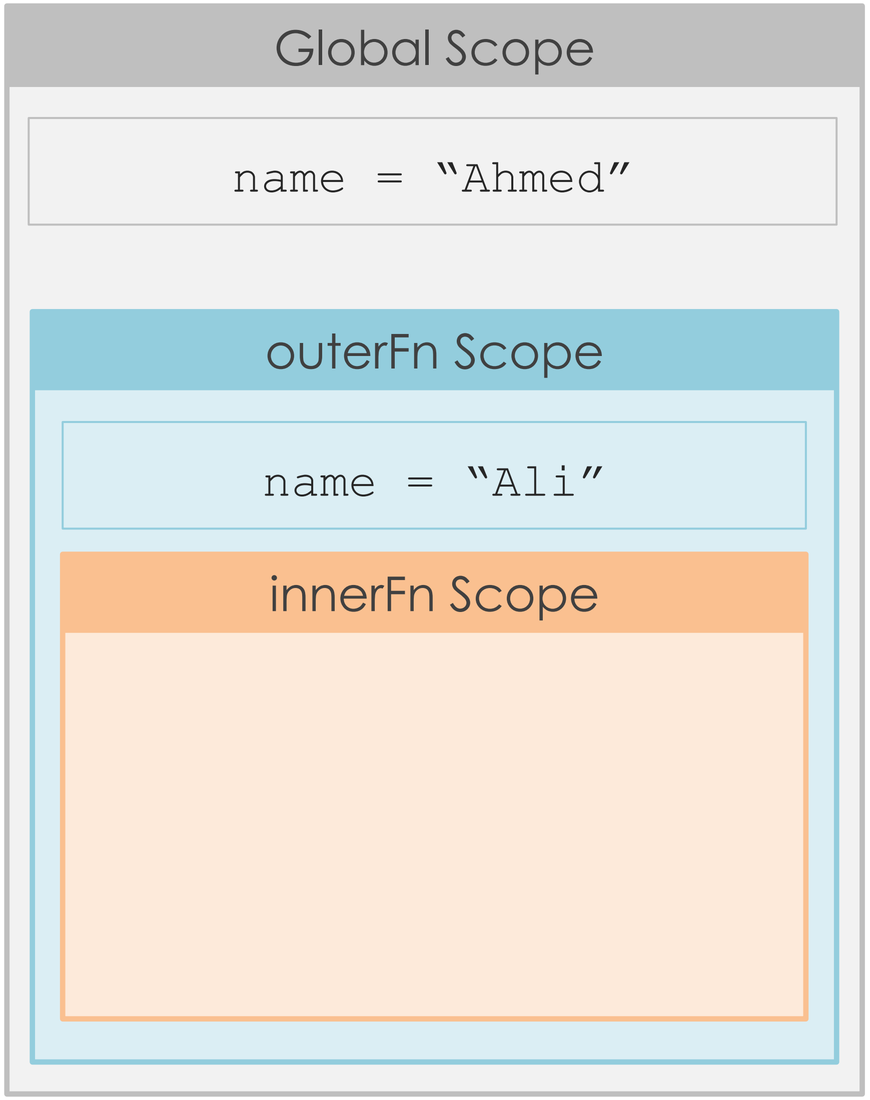
Output:



Lexical Scope

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

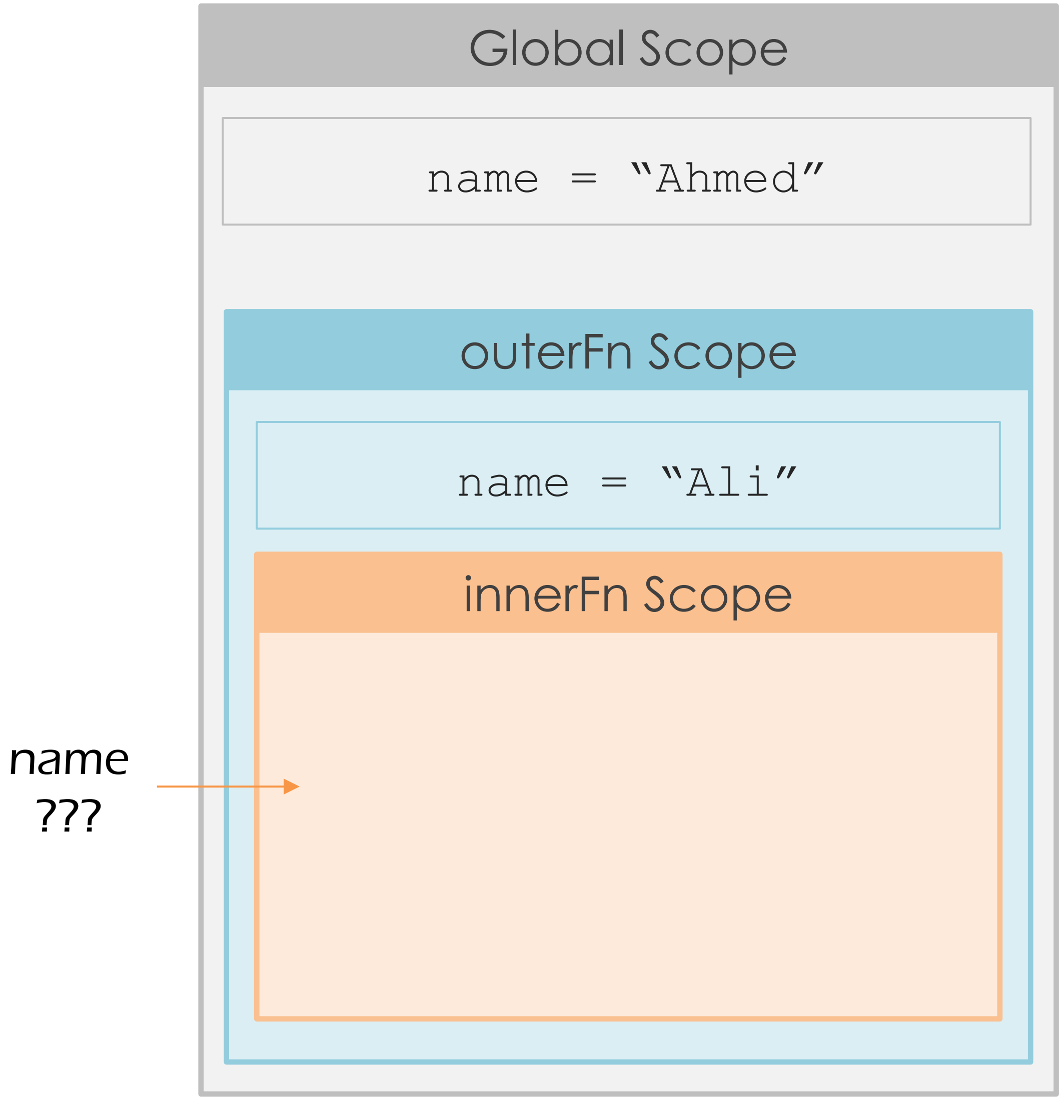
Output:



Lexical Scope

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

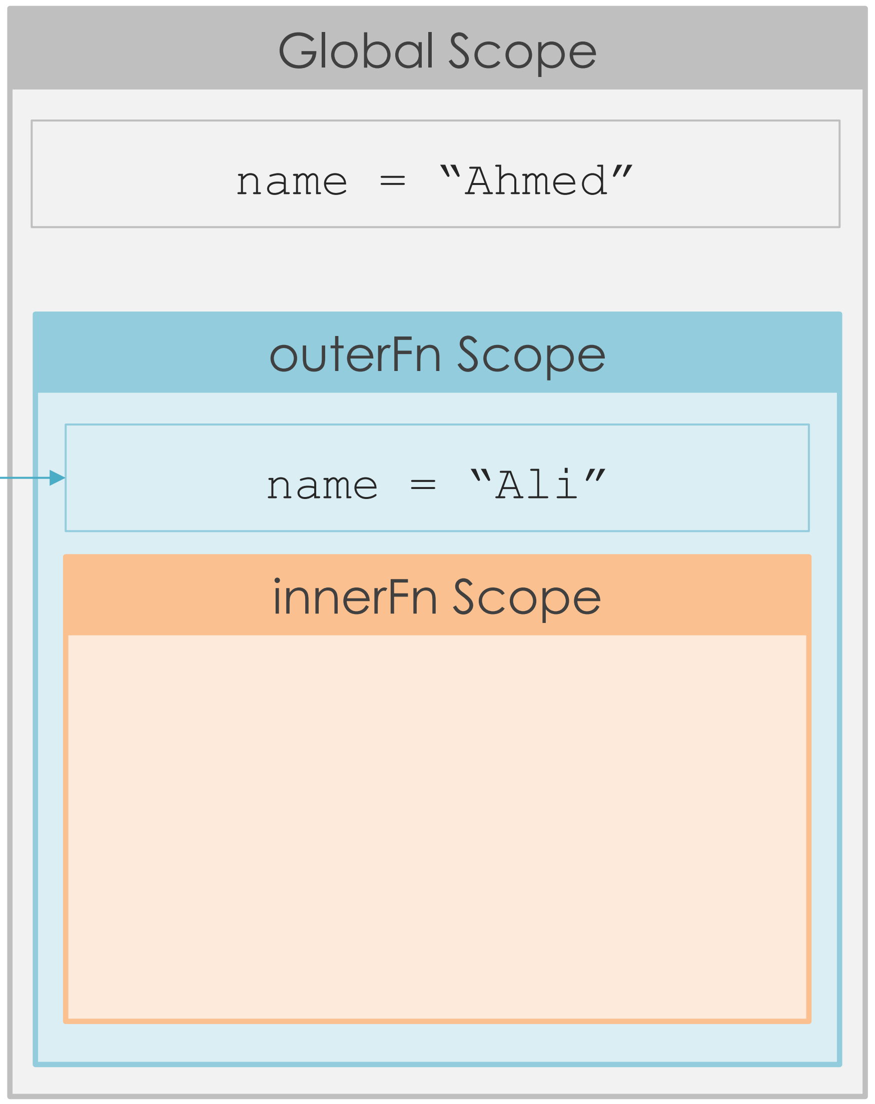
Output:



Lexical Scope

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

name
???

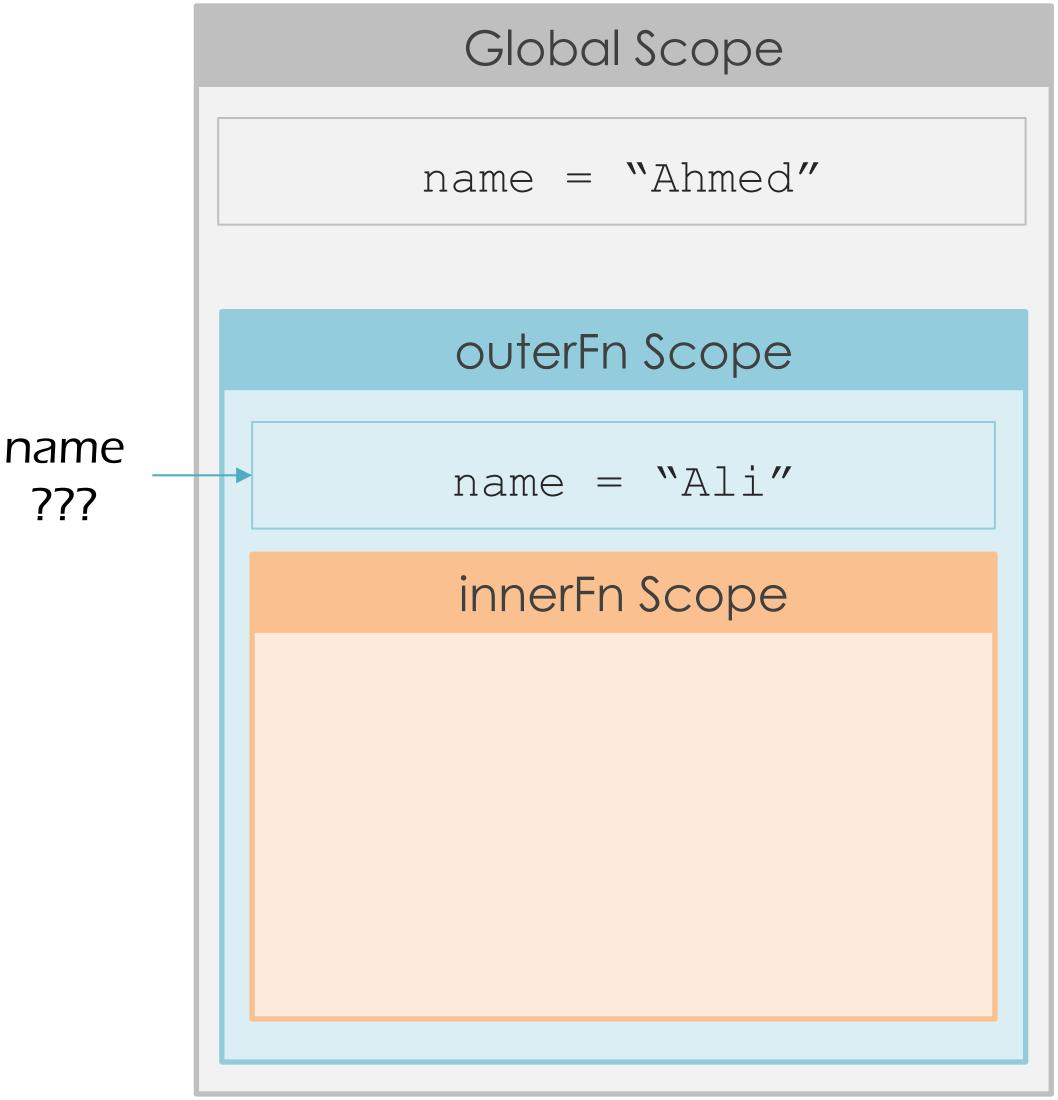


```
Output:
```

Lexical Scope

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

```
Output:
Ali
```

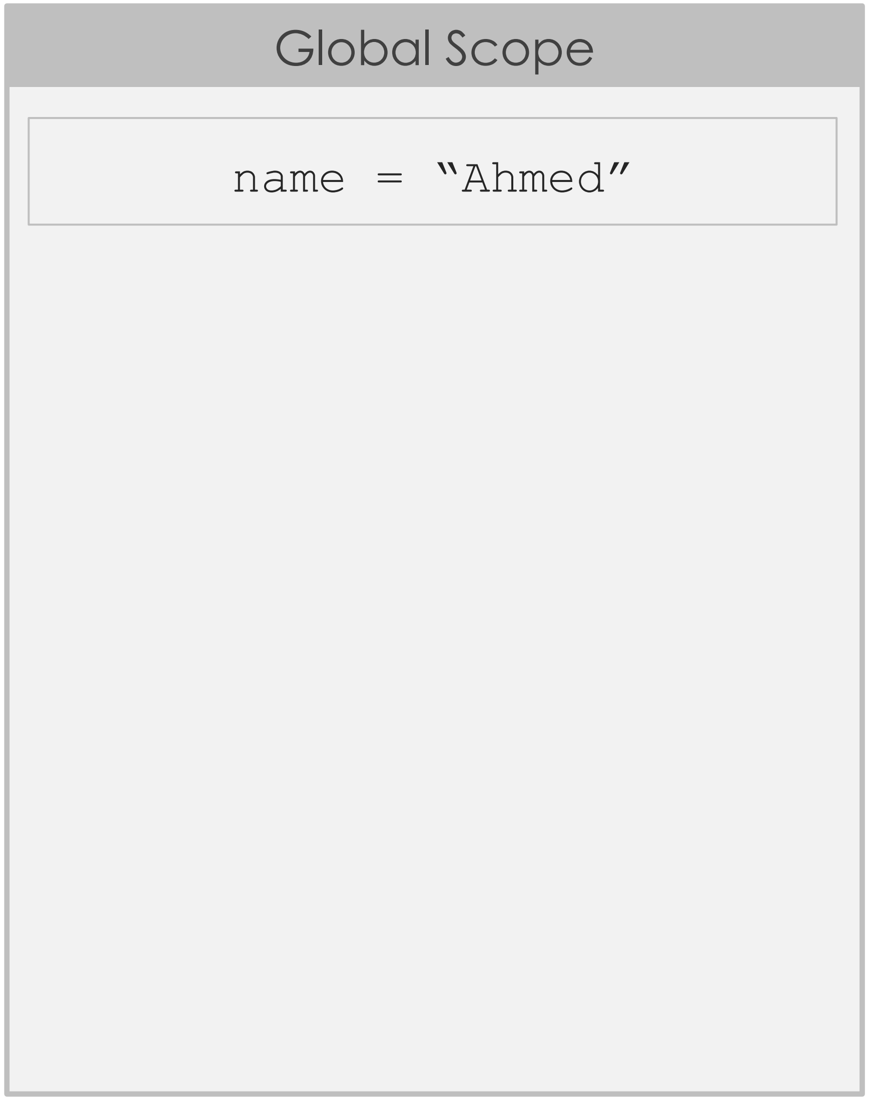


Lexical Scope

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

Output:

```
Ali
```

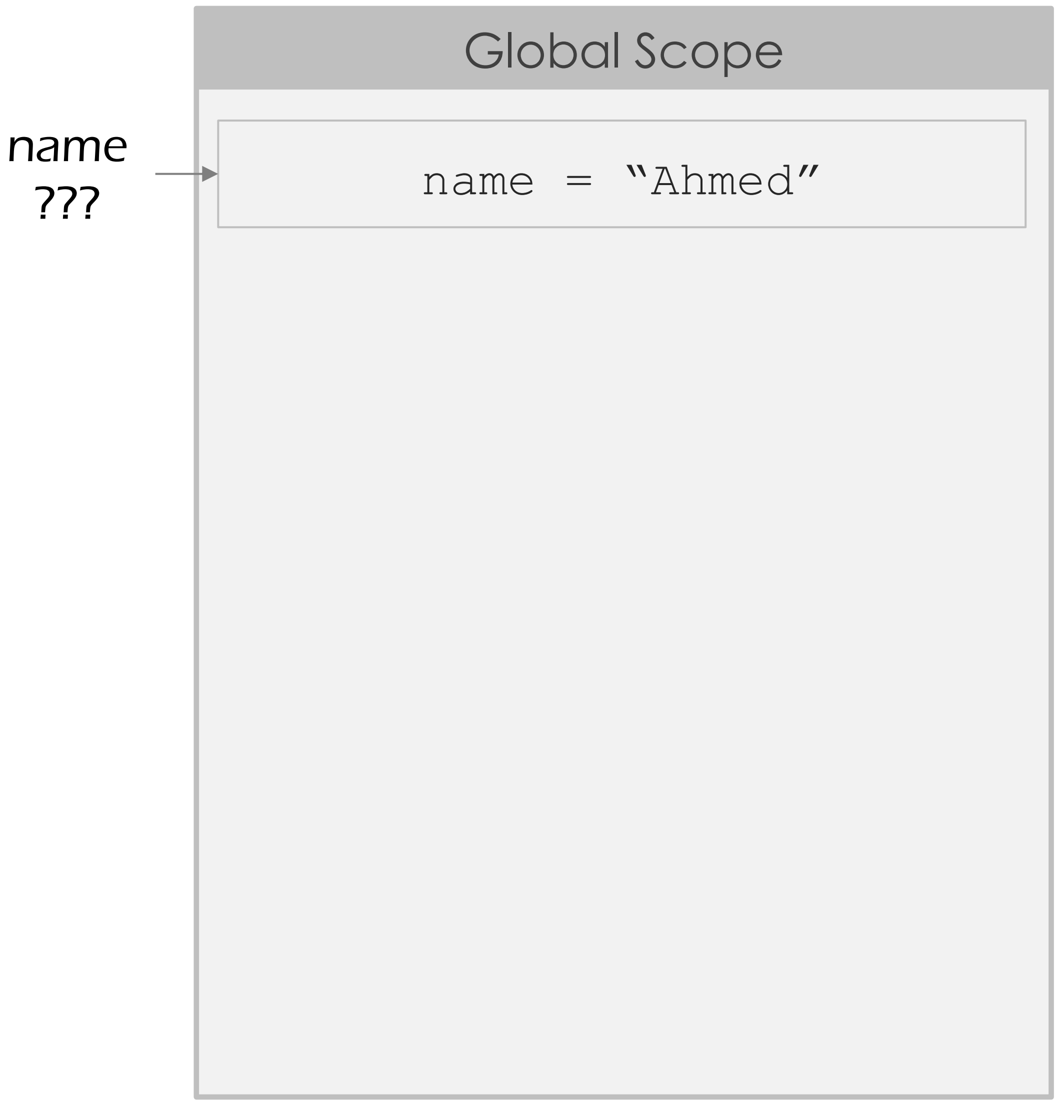


Lexical Scope

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

Output:

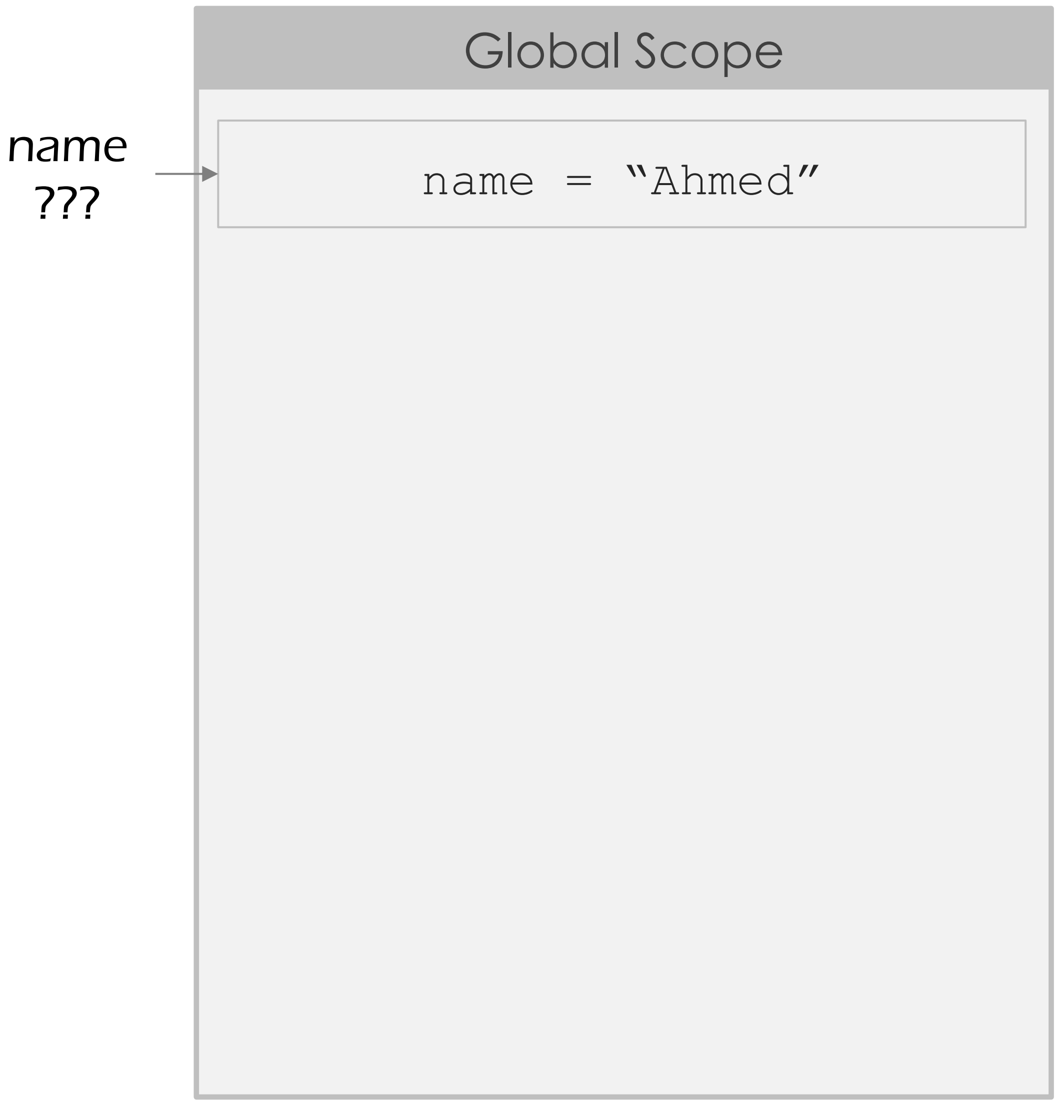
```
Ali
```



Lexical Scope

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

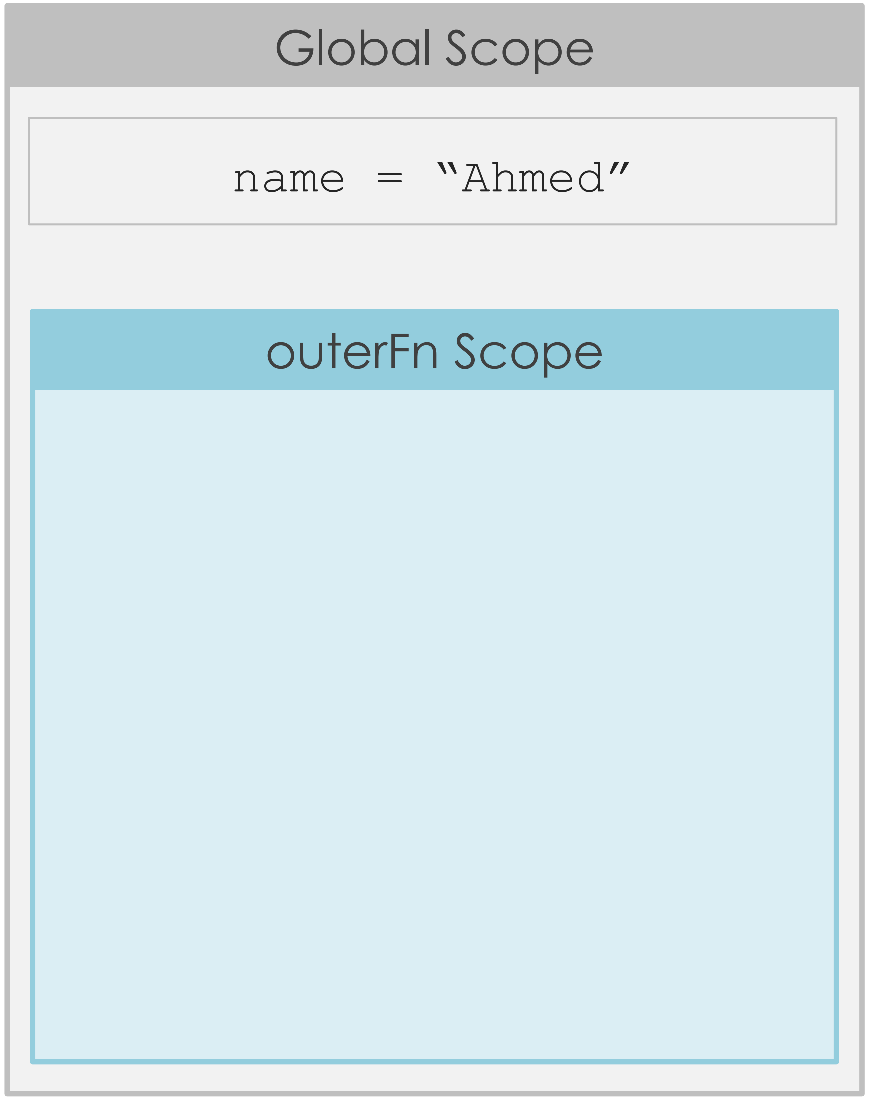
```
Output:
Ali
Ahmed
```



global Keyword

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

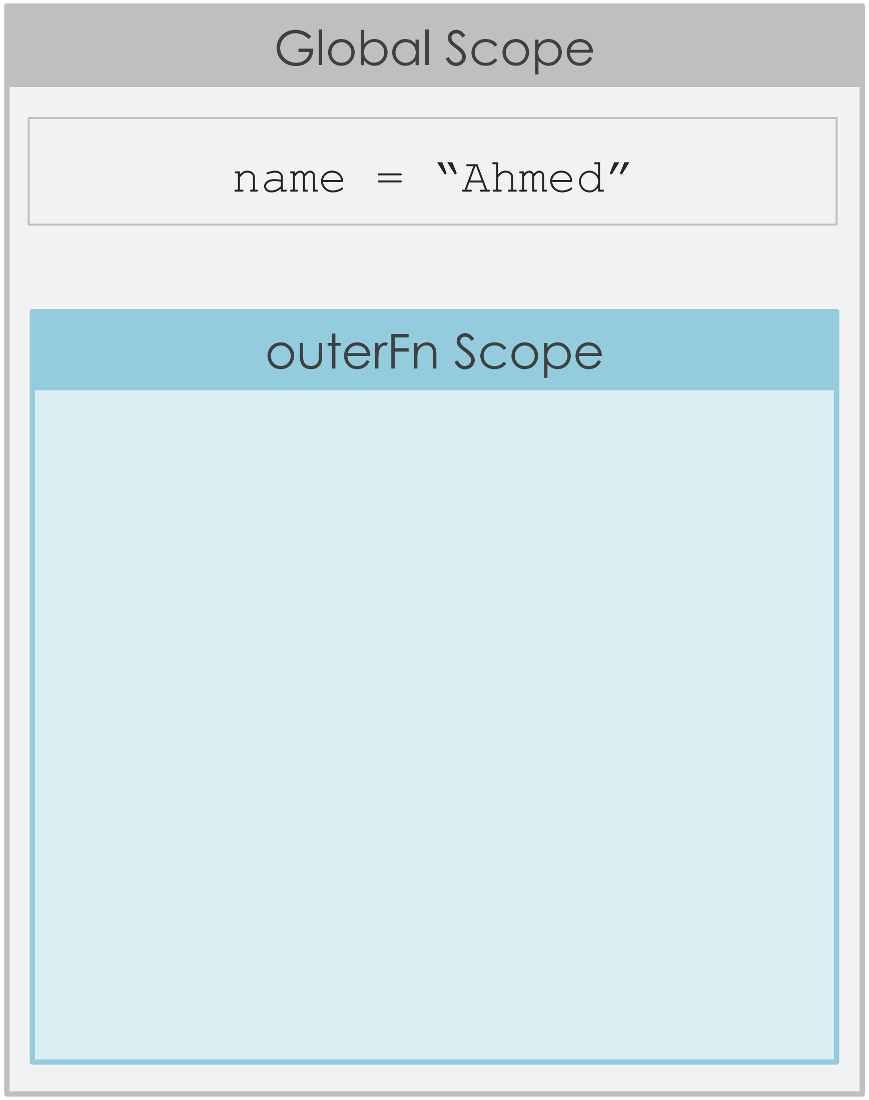
Output:



global Keyword

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

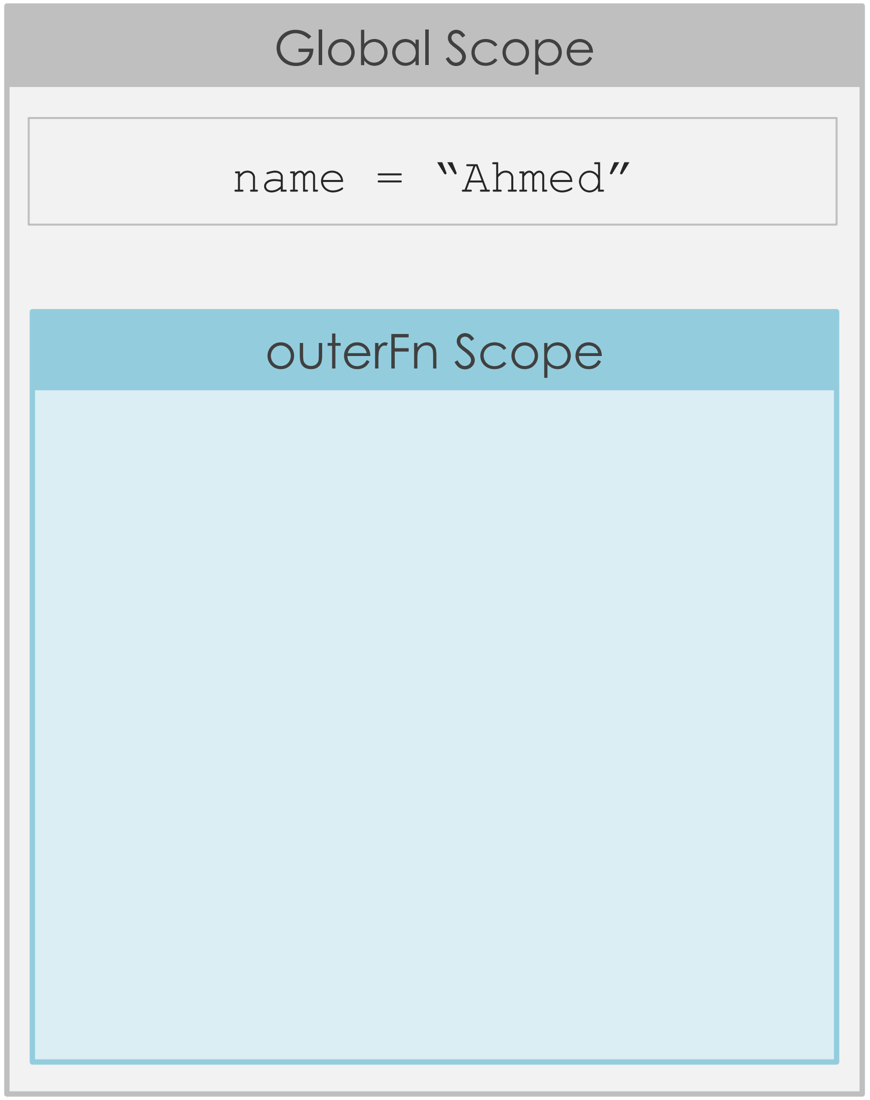
Output:



global Keyword

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

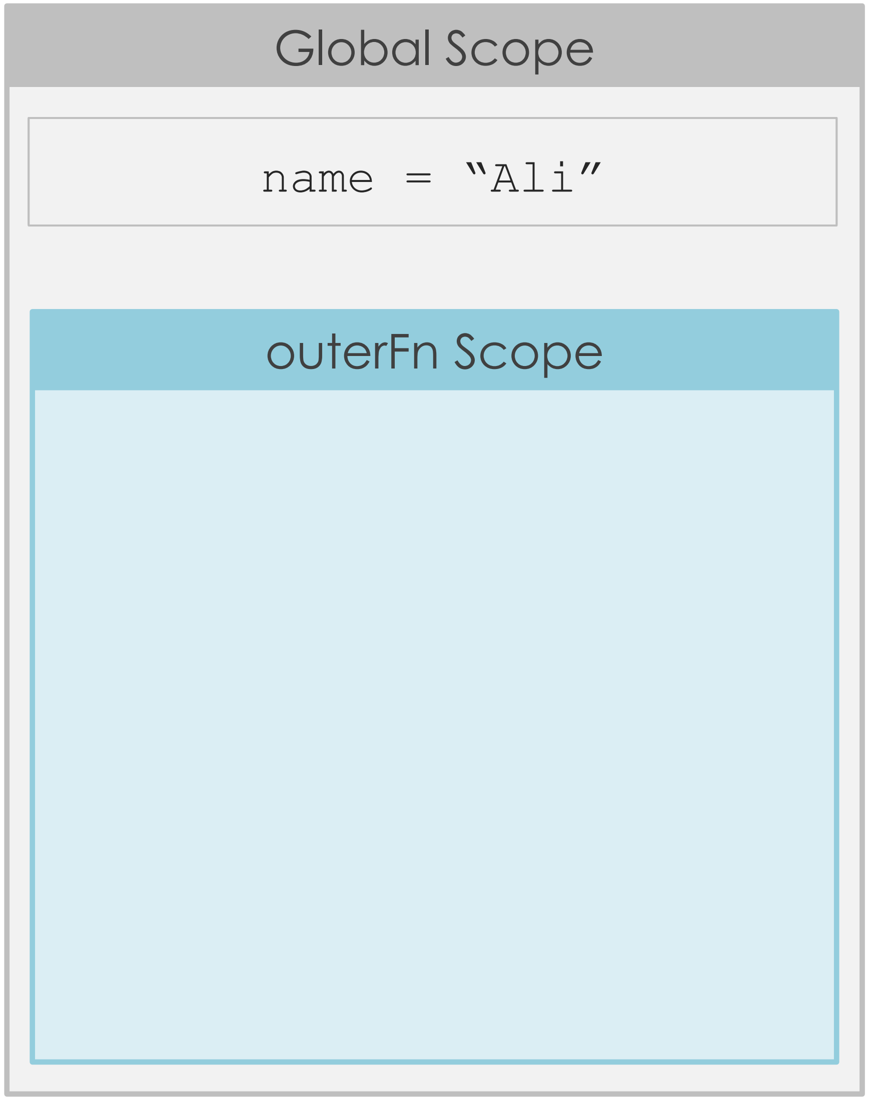
```
Output:
```



global Keyword

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

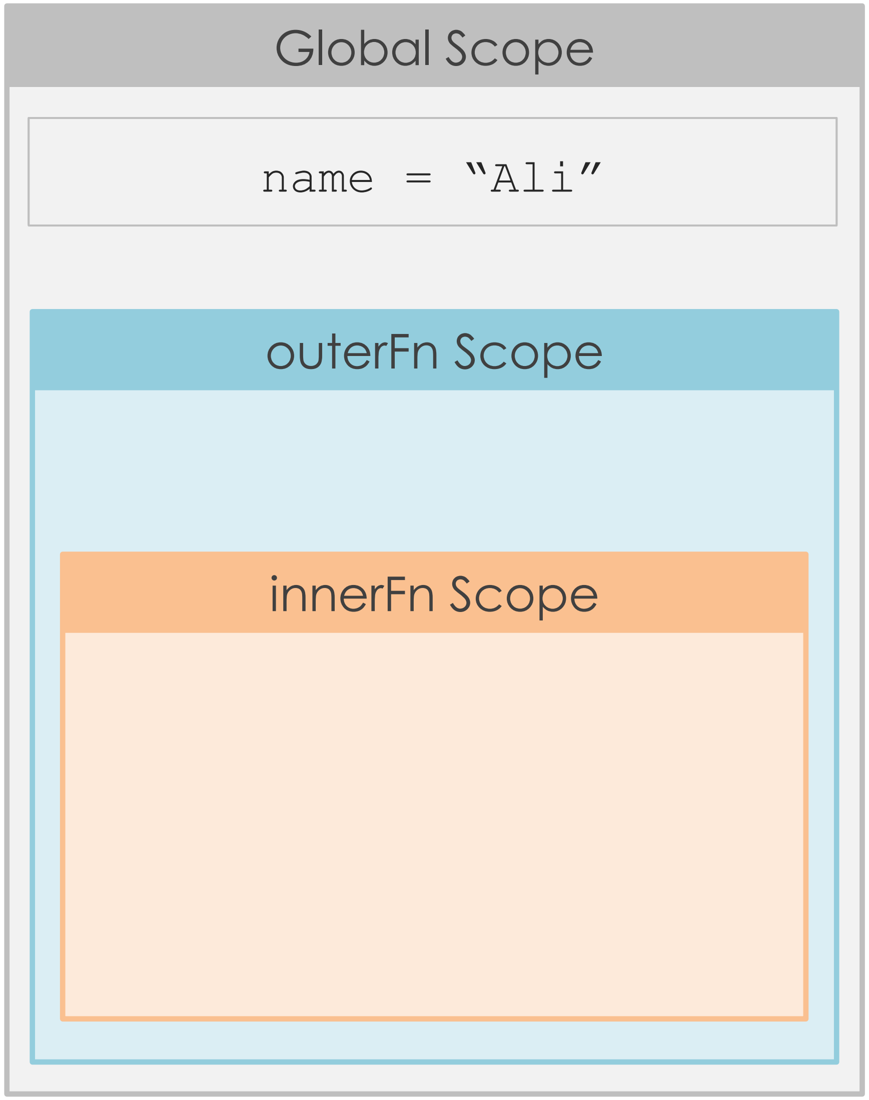
Output:



global Keyword

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

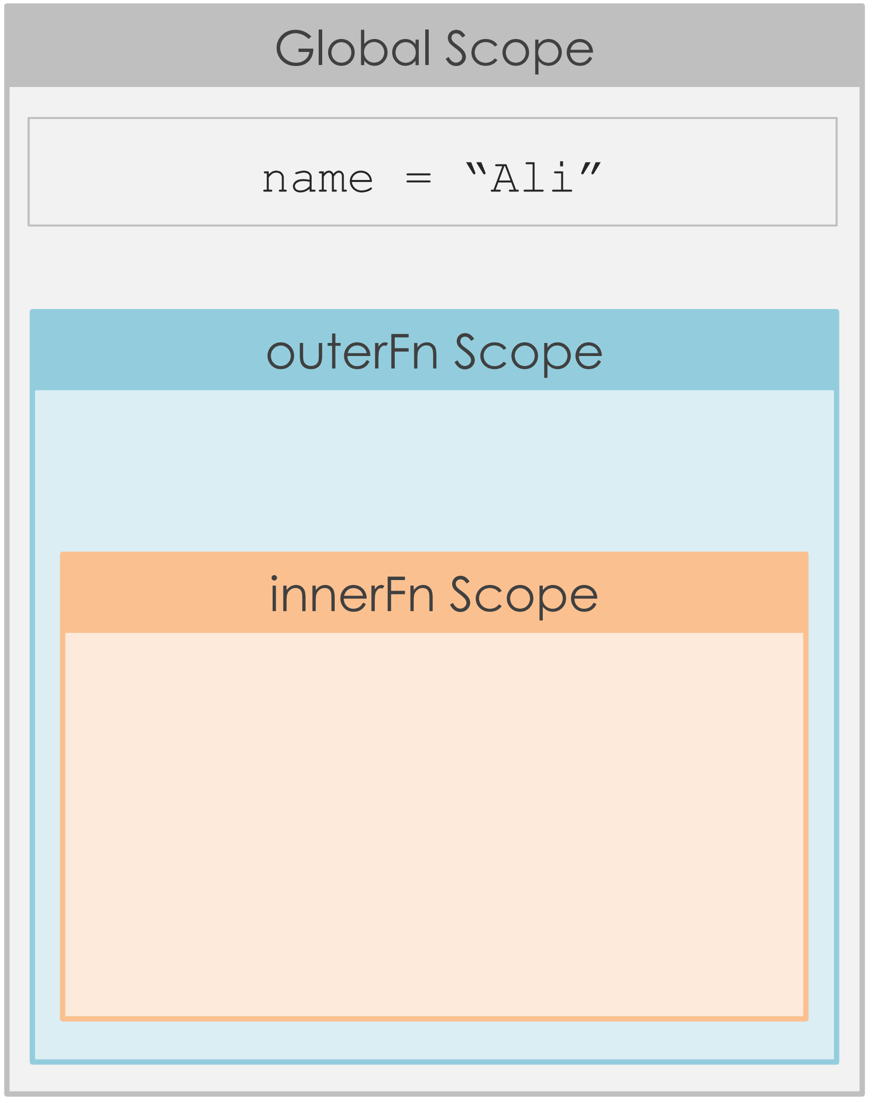
Output:



global Keyword

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

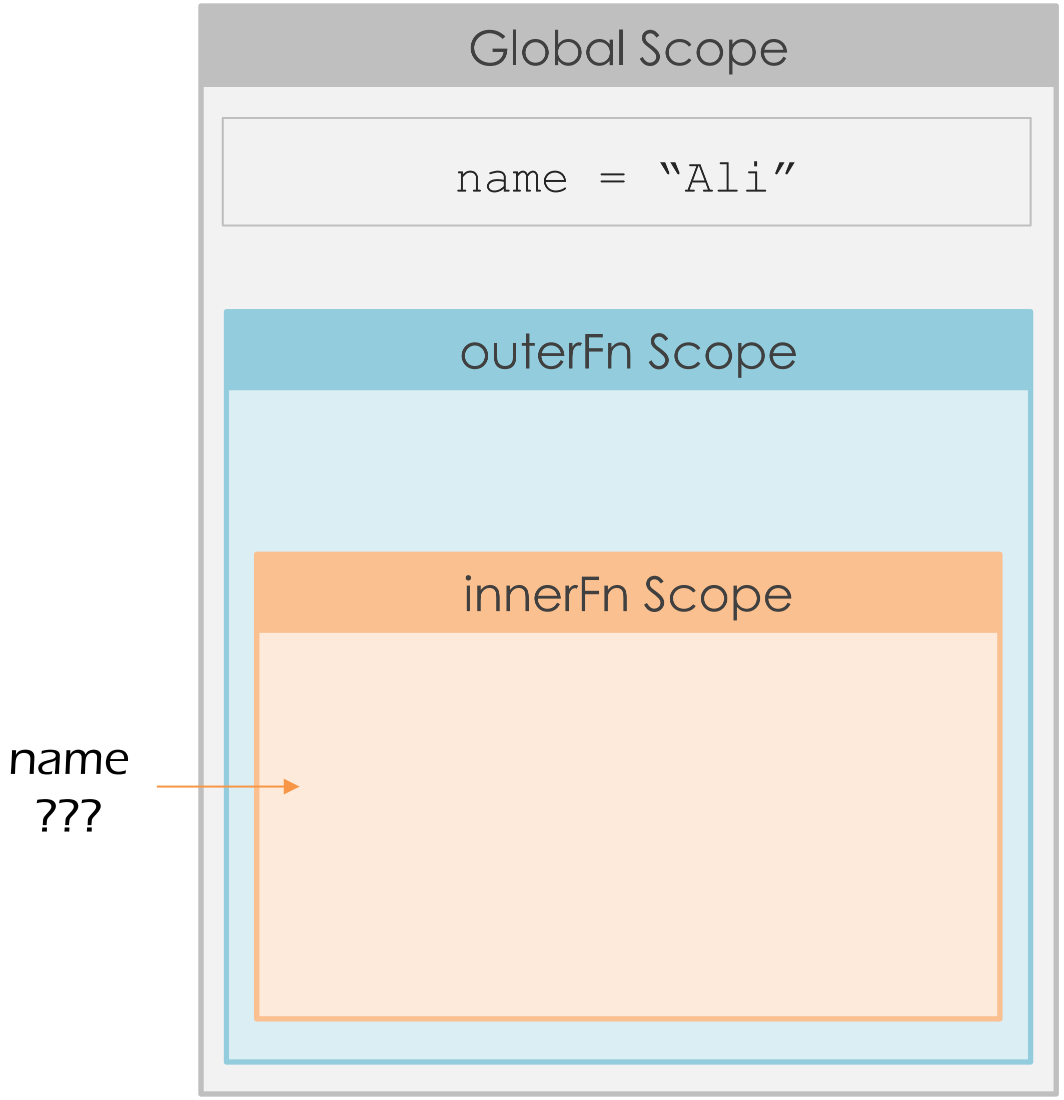
Output:



global Keyword

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

Output:

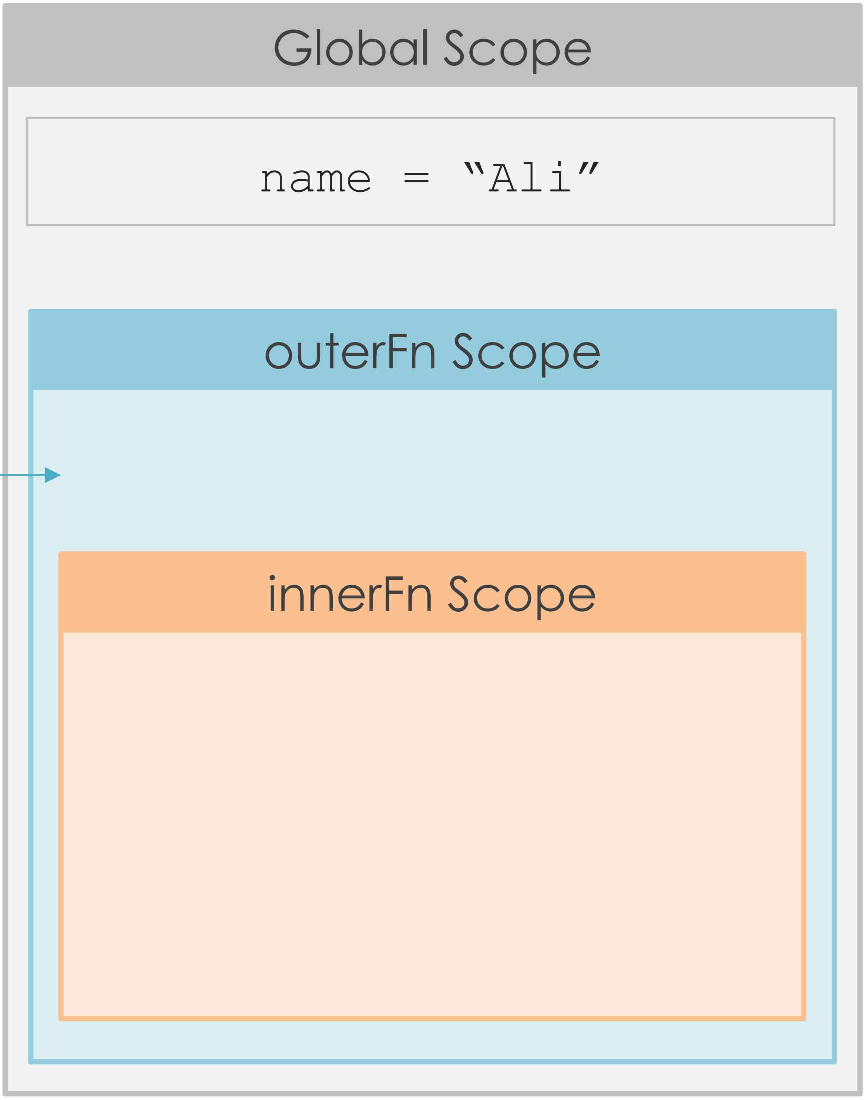


global Keyword

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

Output:

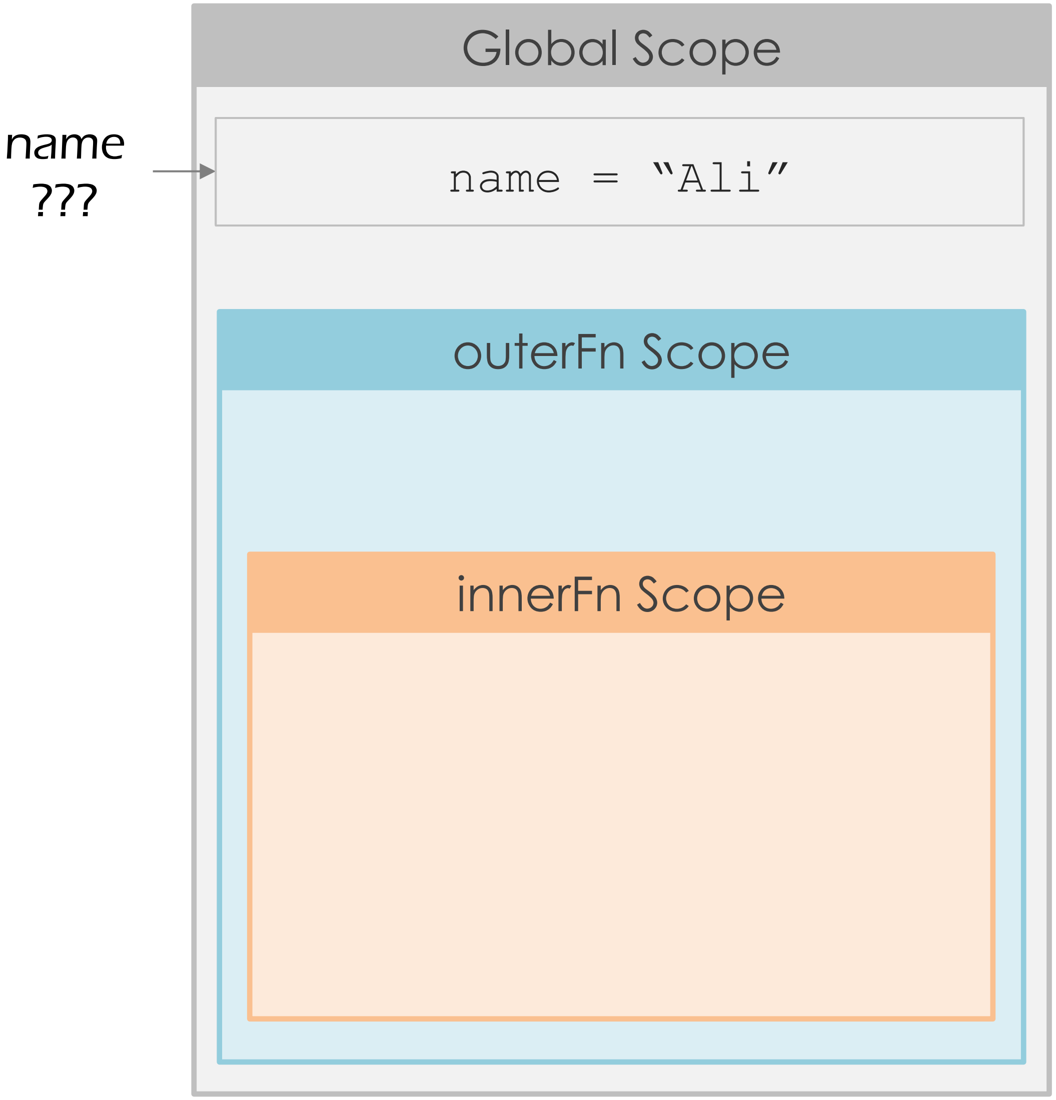
name
???



global Keyword

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

Output:

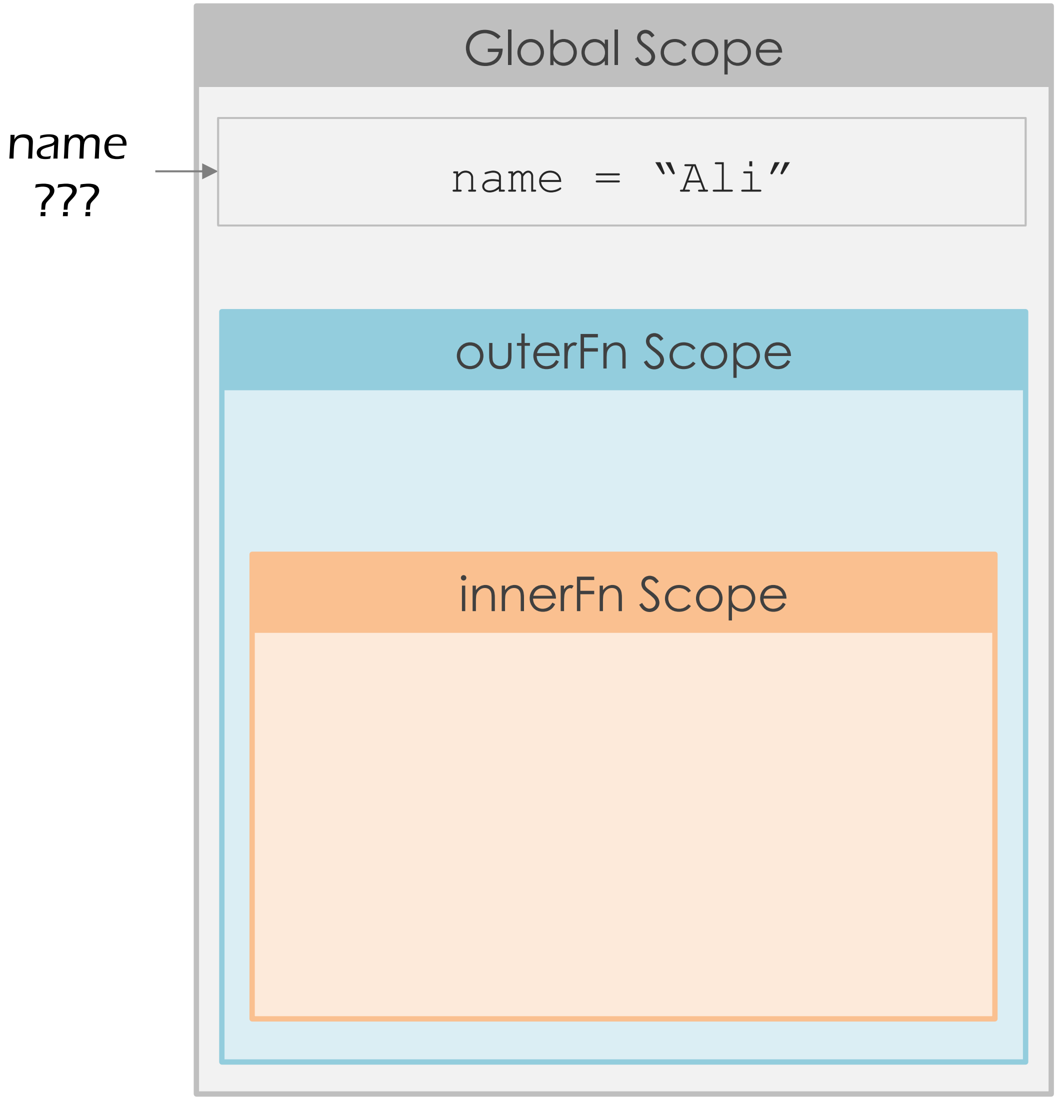


global Keyword

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

Output:

```
Ali
```

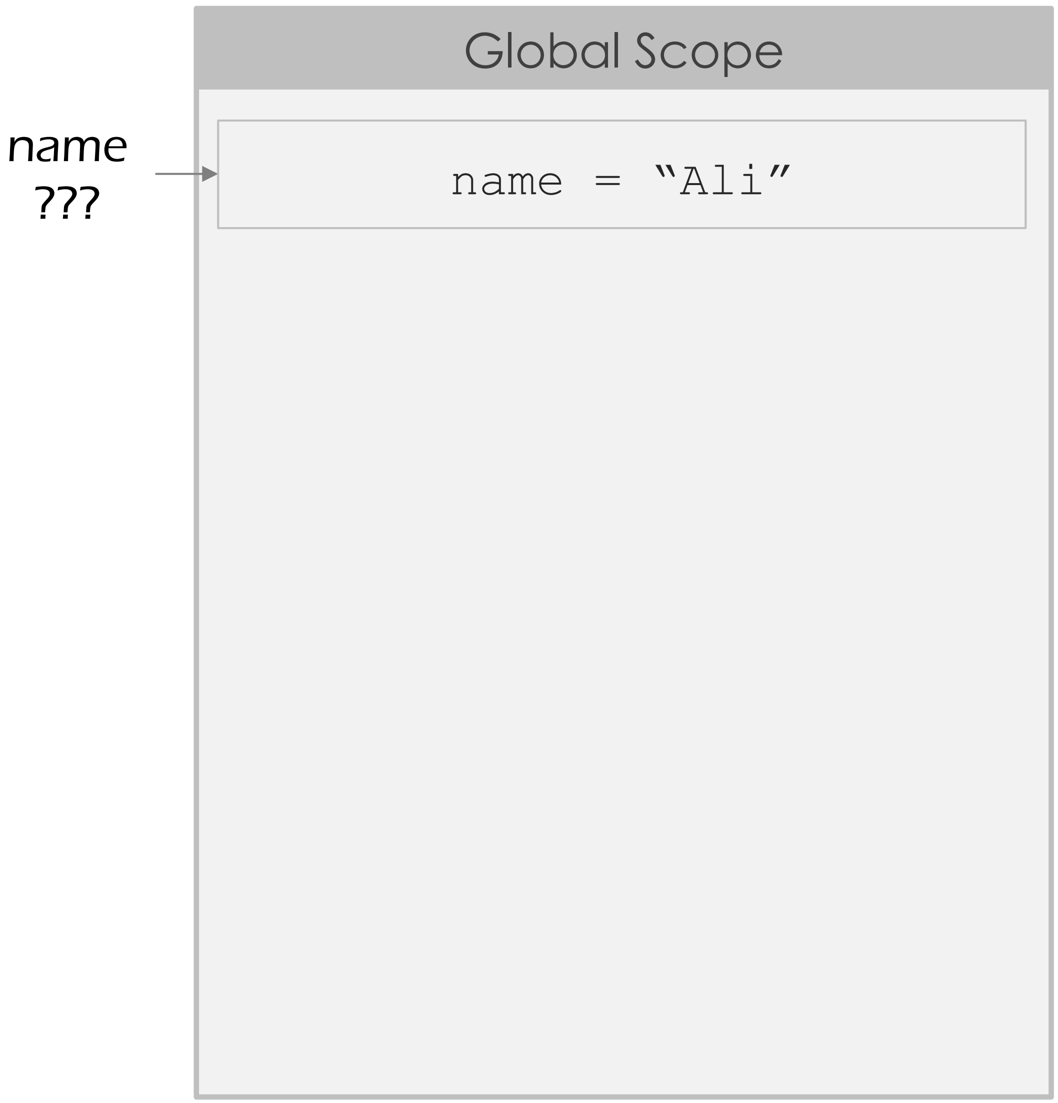


global Keyword

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

Output:

```
Ali
```

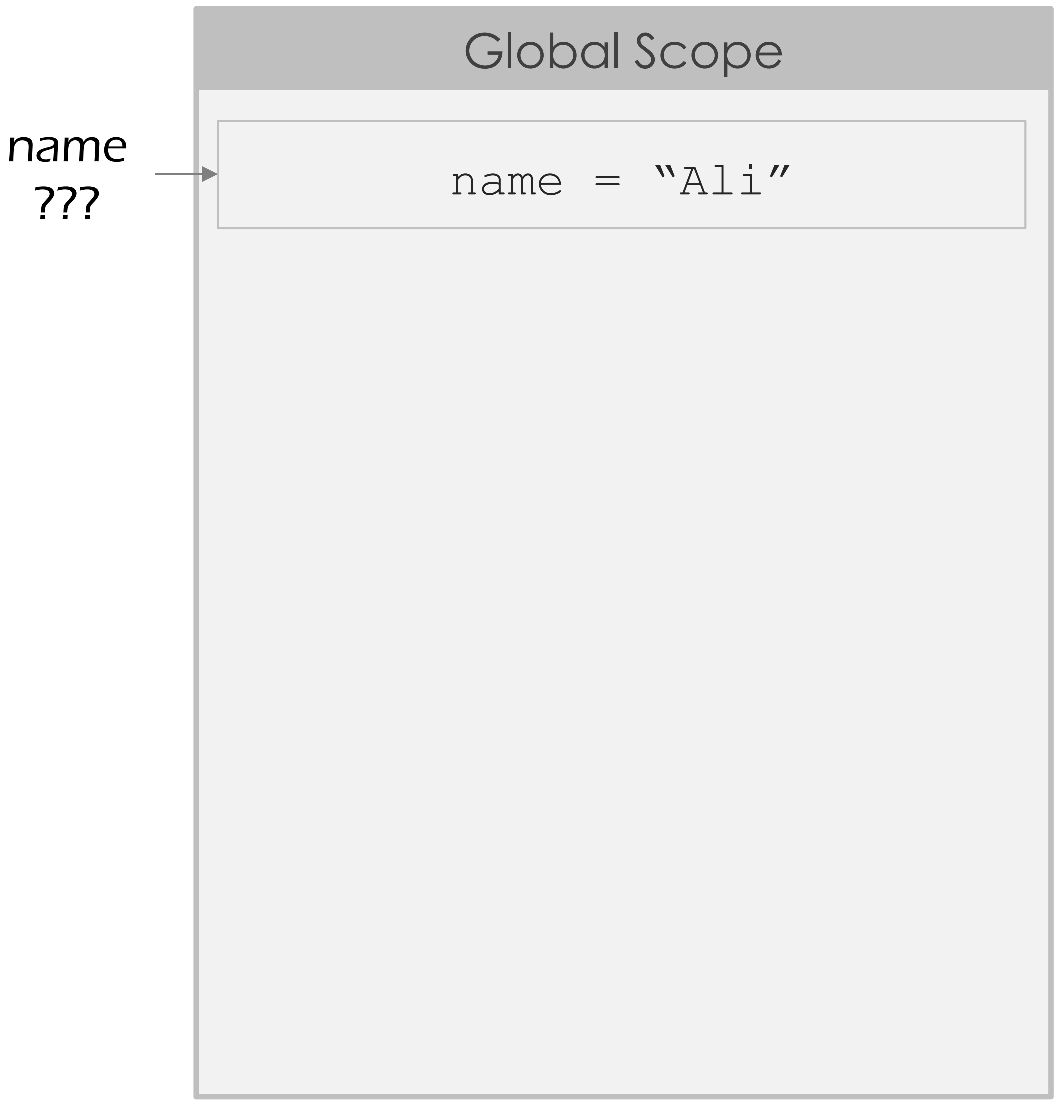


global Keyword

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

Output:

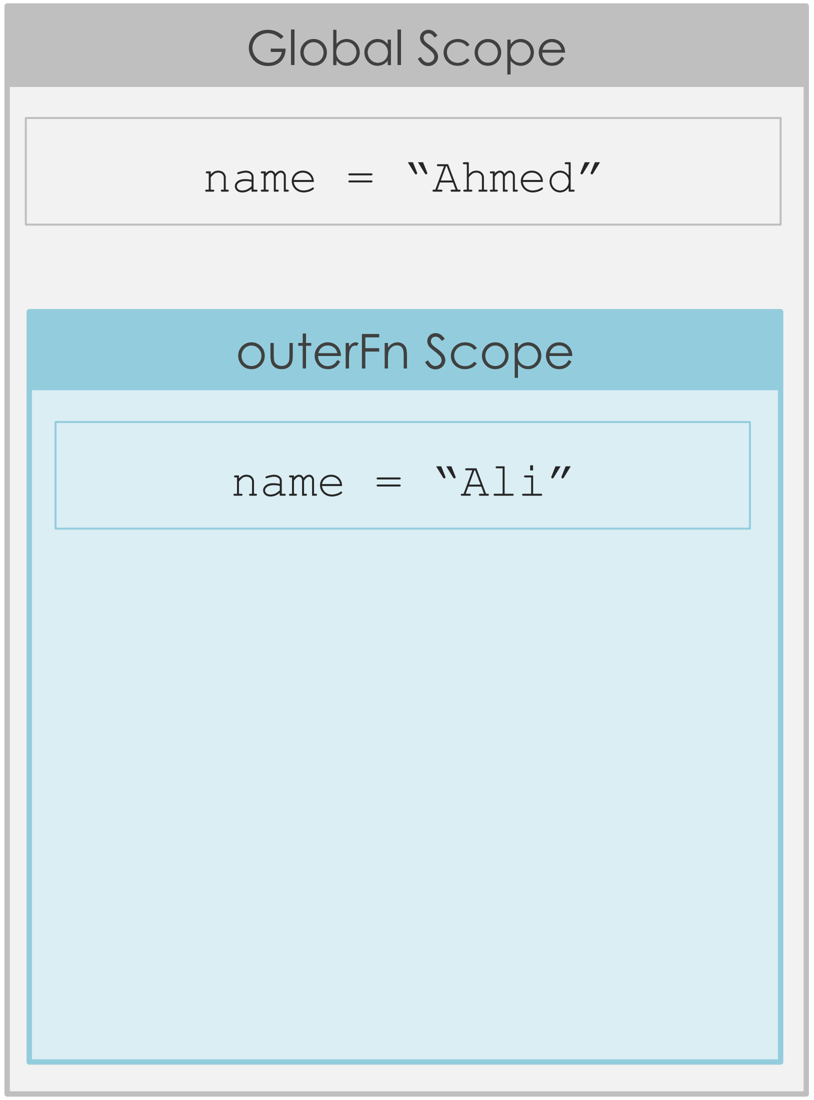
```
Ali
Ali
```



nonlocal Keyword

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

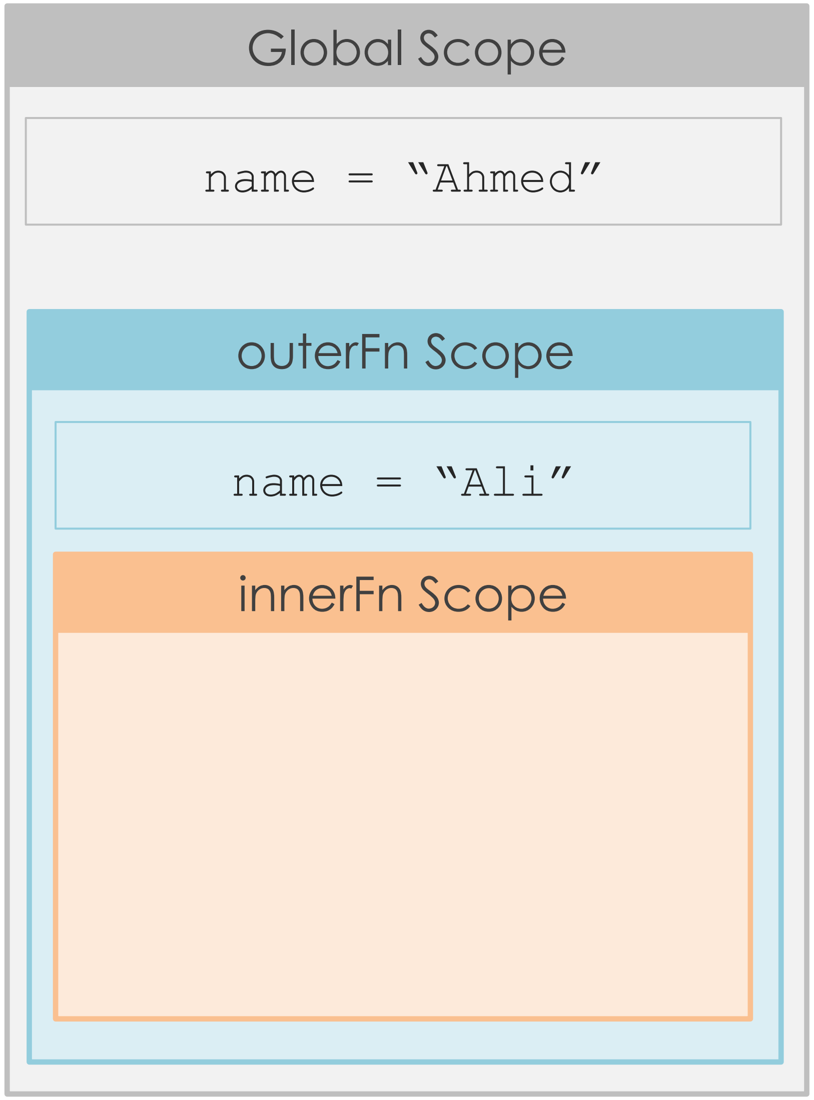
Output:



nonlocal Keyword

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

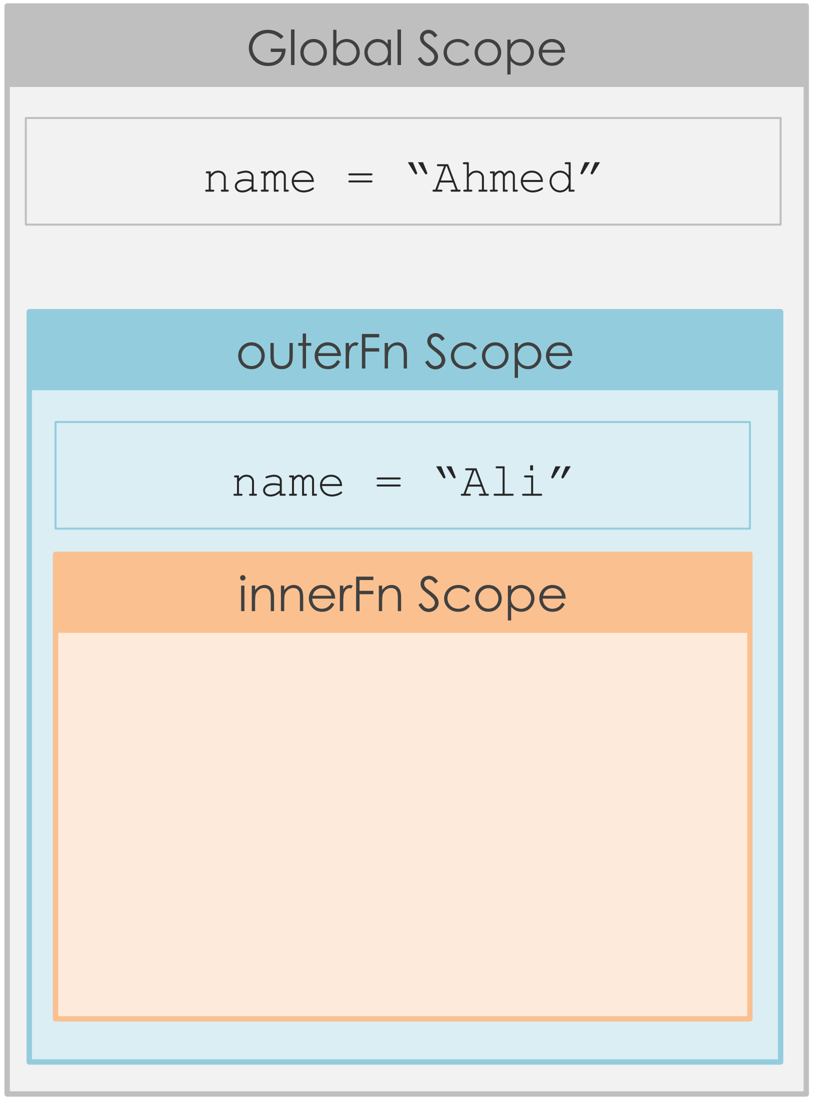
Output:



nonlocal Keyword

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

Output:

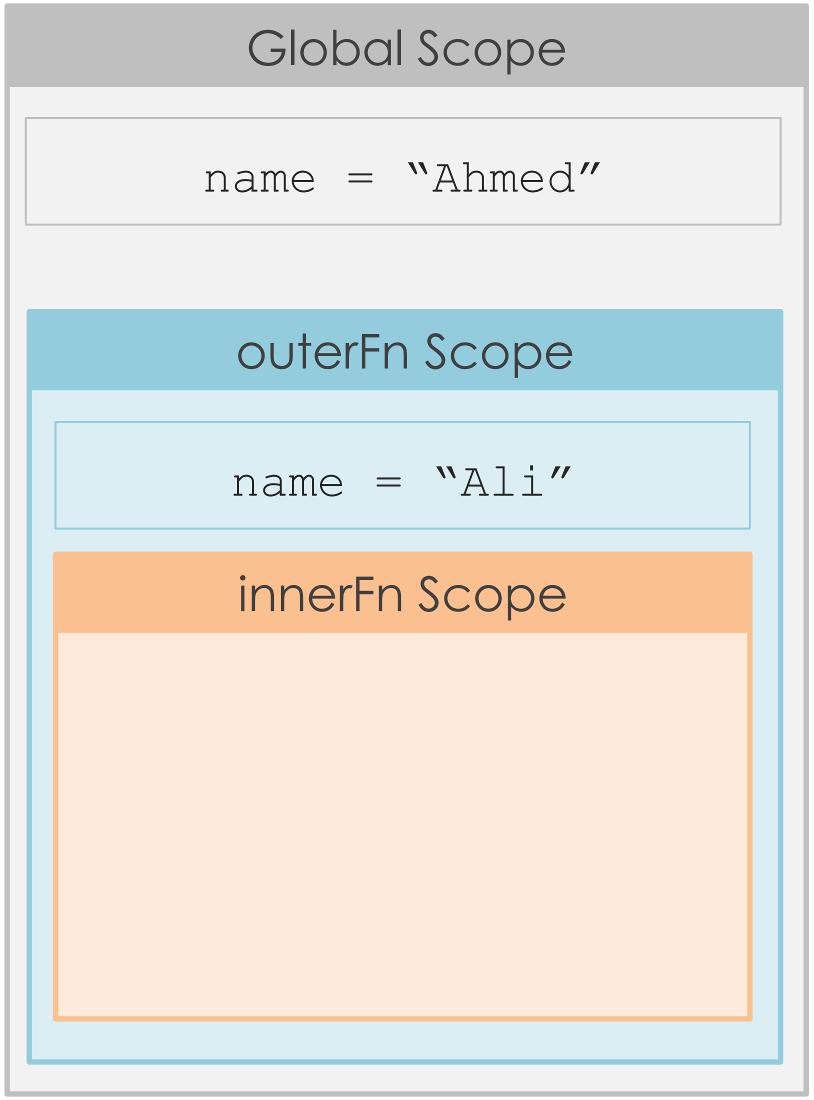


nonlocal Keyword

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

Output:

```
Ali
```

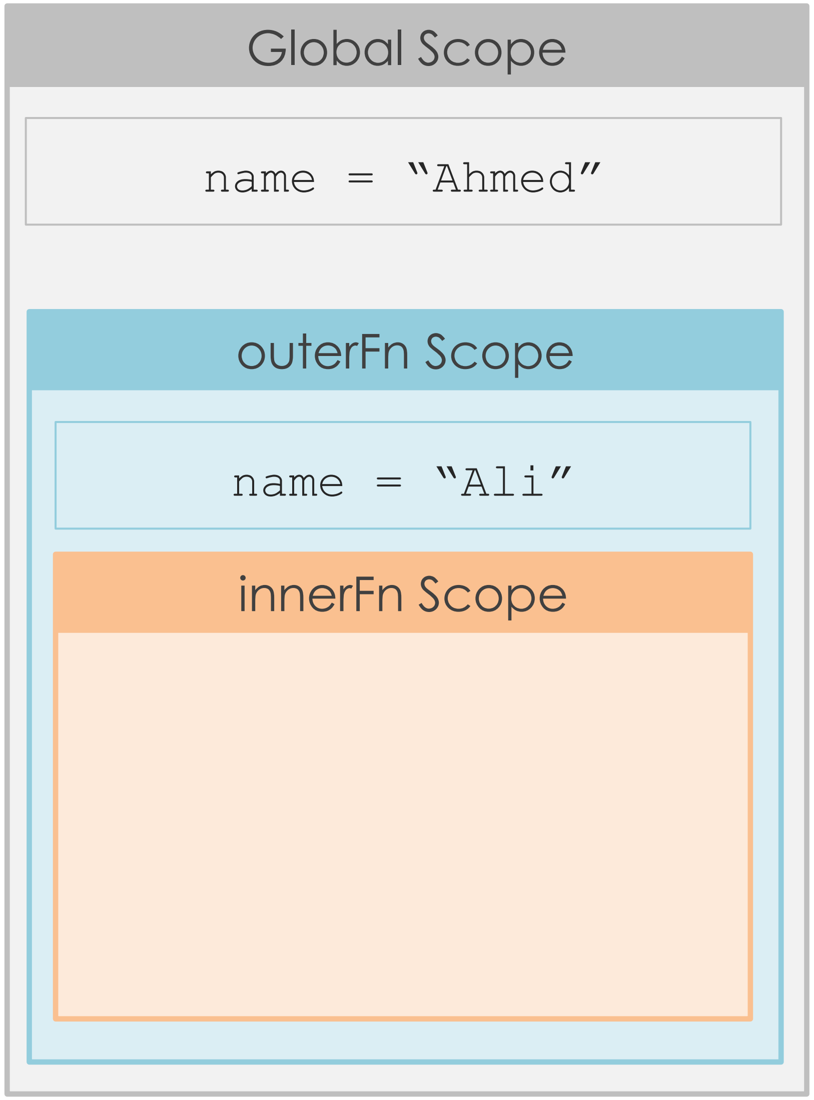


nonlocal Keyword

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

Output:

```
Ali
```

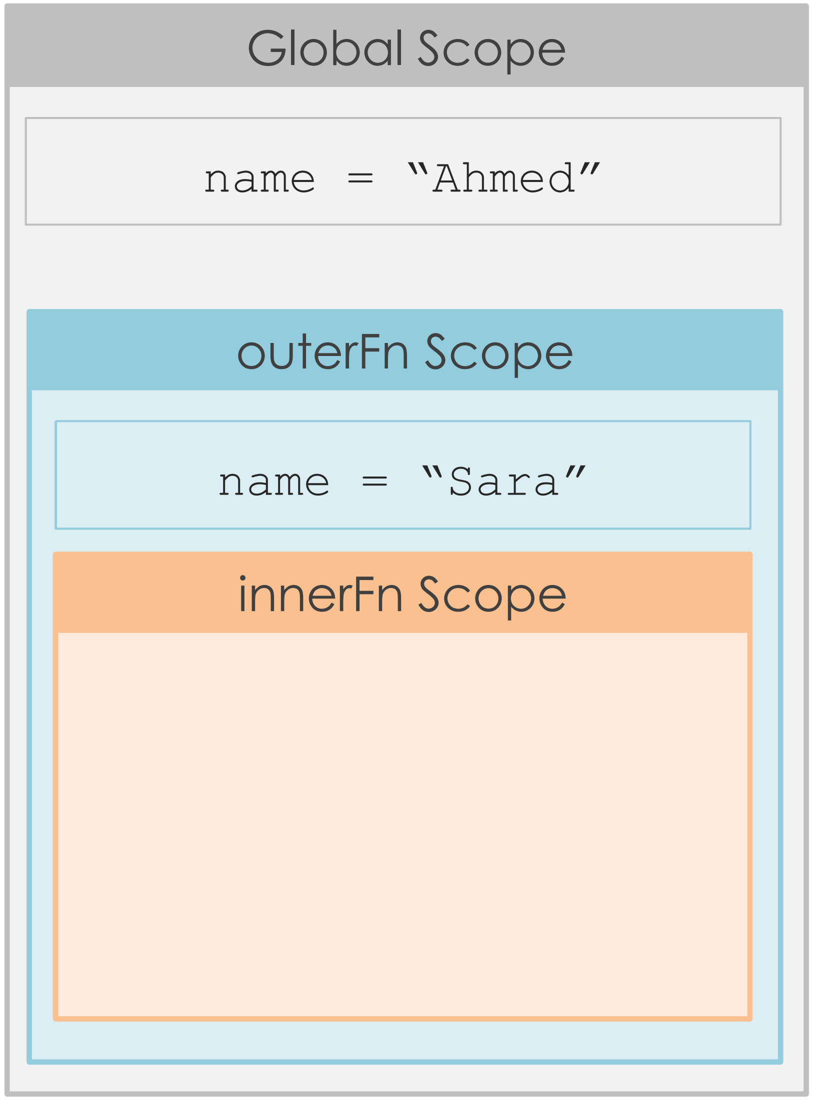


nonlocal Keyword

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

Output:

```
Ali
```

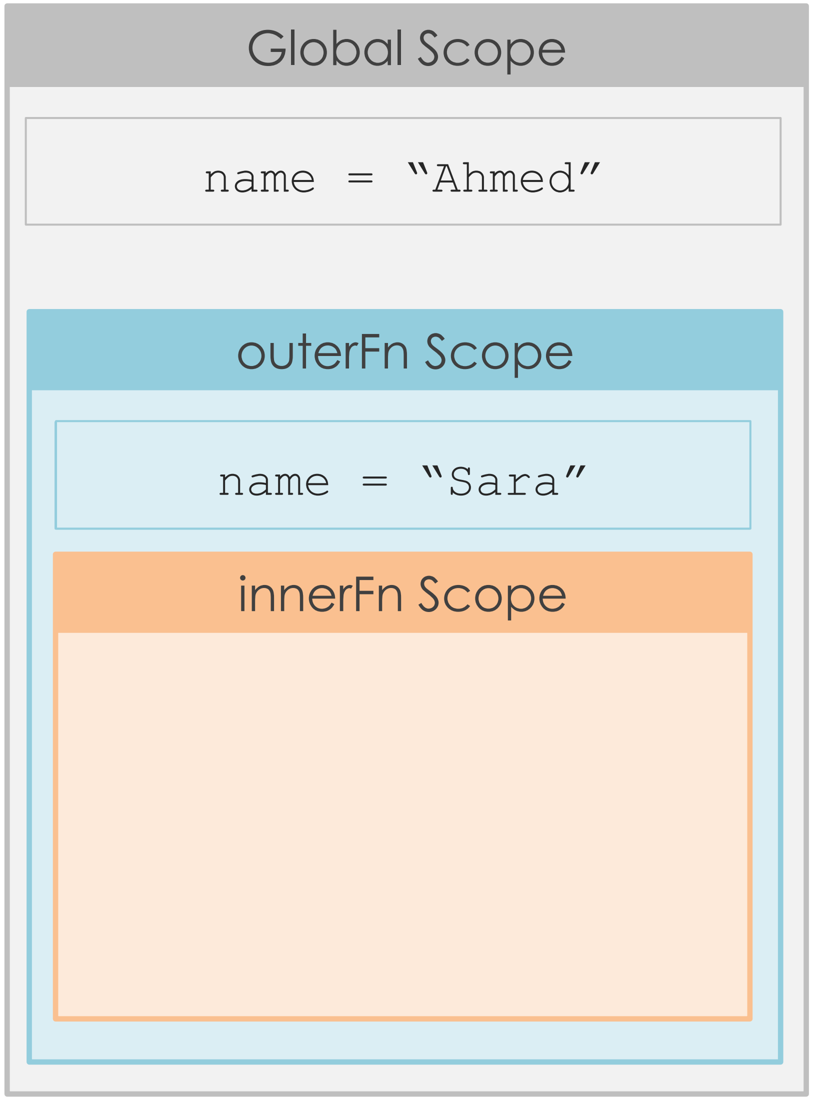


nonlocal Keyword

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

Output:

```
Ali
```

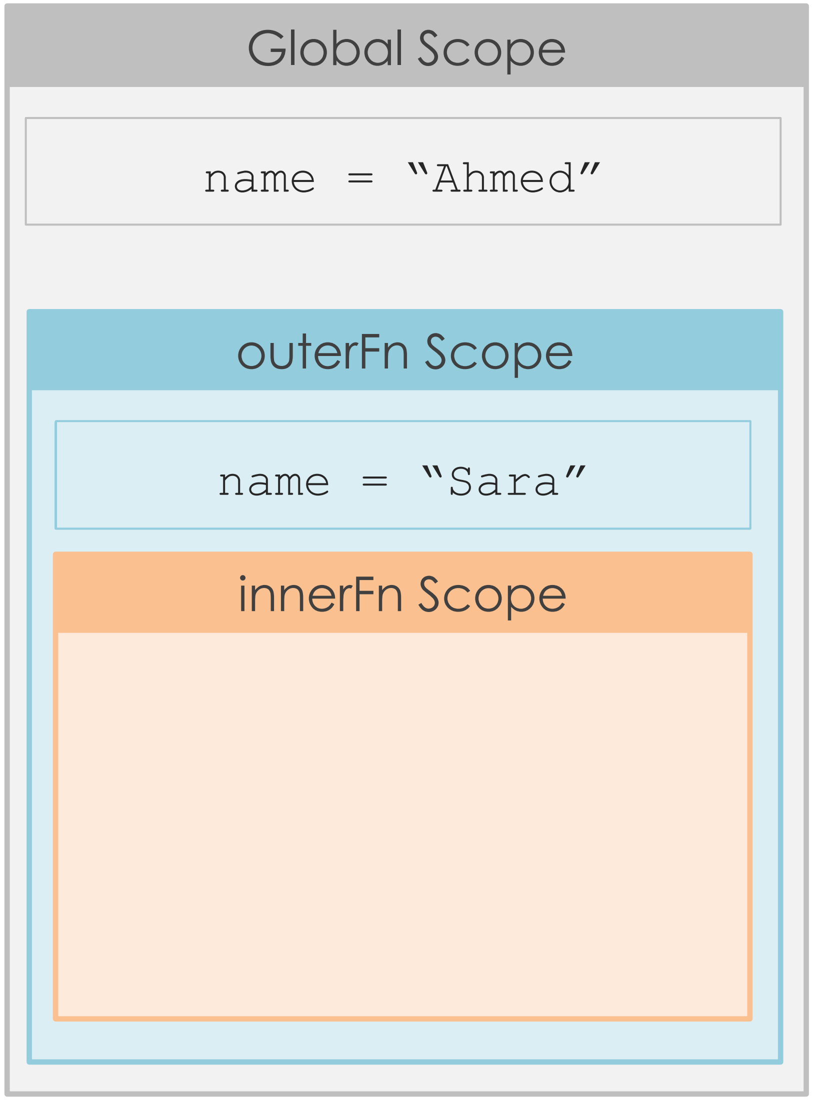


nonlocal Keyword

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

Output:

```
Ali
Sara
```

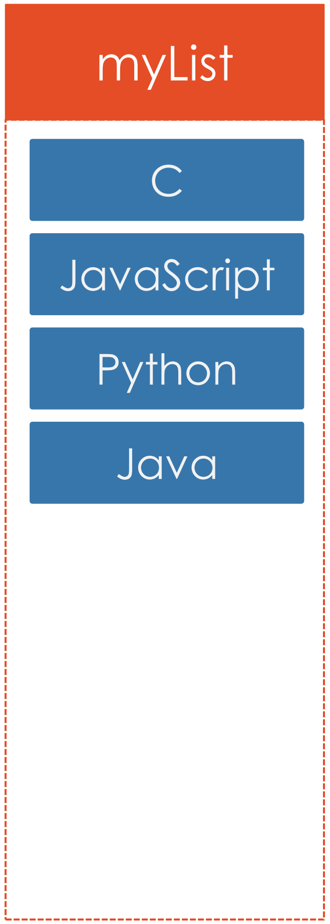


More in Lists

More and More

Methods

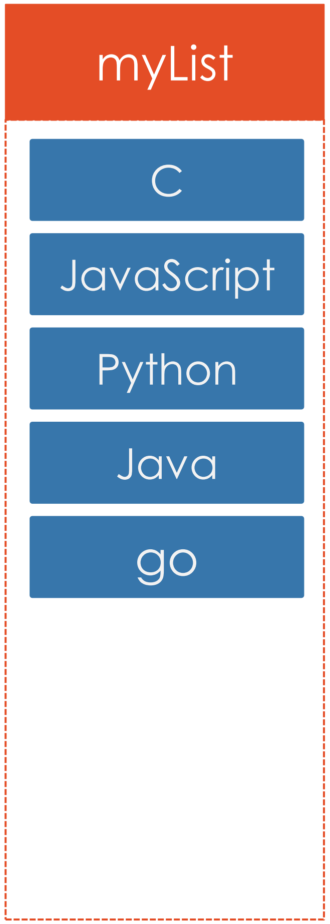
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```



```
myList.pop(4)
```

Methods

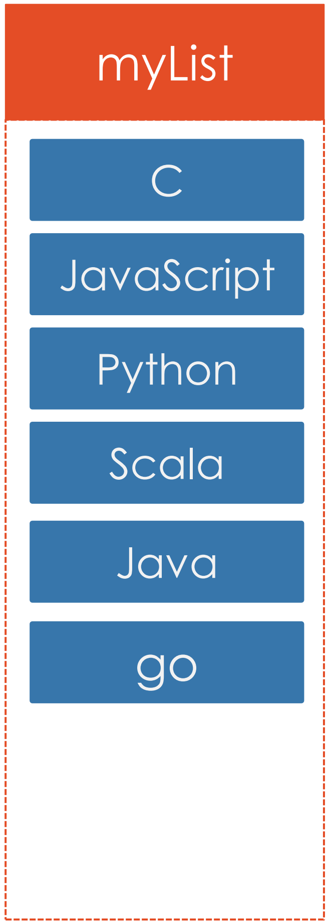
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```



```
myList.pop(4)

myList.append("go")
```

Methods

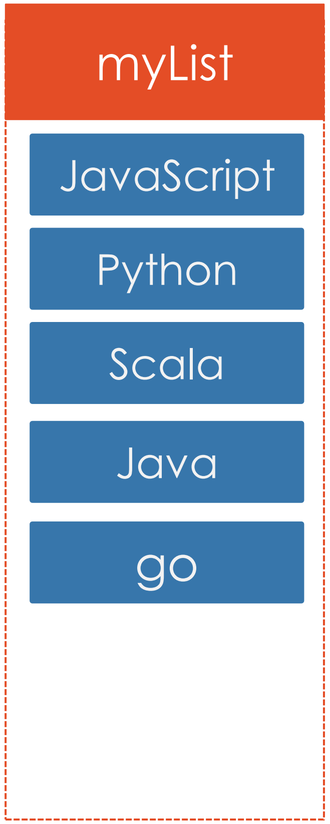
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```



```
myList.pop(4)

myList.append("go")

myList.insert(3, 'Scala')
```

Methods

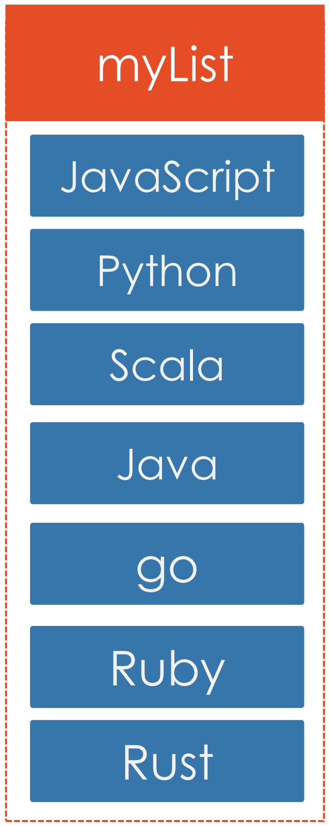
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```



```
myList.pop(4)

myList.append("go")

myList.insert(3, 'Scala')

myList.remove("C")
```

Methods

```
myList = [“C”, “JavaScript”, “Python”, “Java”, “php”];
```



```
myList.pop(4)

myList.append(“go”)

myList.insert(3, ‘Scala’)

myList.remove(“C”)

yourList = [“Ruby”, “Rust”];

myList.extend(yourList)
```

Tuples

Immutable Lists

Intro

Same as Lists but Tuples are immutable

```
newTuple = ()
```

```
t = (1, “hi”, True)
t[1]
# hi
t[1] = 4
TypeError: 'tuple' object does not support item assignment
```

Dictionaries

Key/value Pairs

Intro

A key: value **comma** seperated elements Data Structure

```
newDict = {}
```

```
d = {name: “Ahmed”, track: “OS”}
d[name]
# Ahmed
d[name] = “Ali”
# {name: “Ali”, track: “OS”}
```

Methods

```
infoDict = {'track': 'OS', 'name': 'Ahmed', 'age': 17}

infoDict.keys() # dict_keys(['track', 'name', 'age'])

'name' in infoDict # True

infoDict.items()

# dict_items([('track', 'OS'), ('name', 'Ahmed'), ('age', 17)])

addInfoDict = {'track': 'SD', 'branch': "Smart"}

infoDict.update(addInfoDict)

#{'track': 'SD', 'name': 'Ahmed', 'age': 17 , 'branch': "Smart"}
```

**keywords

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

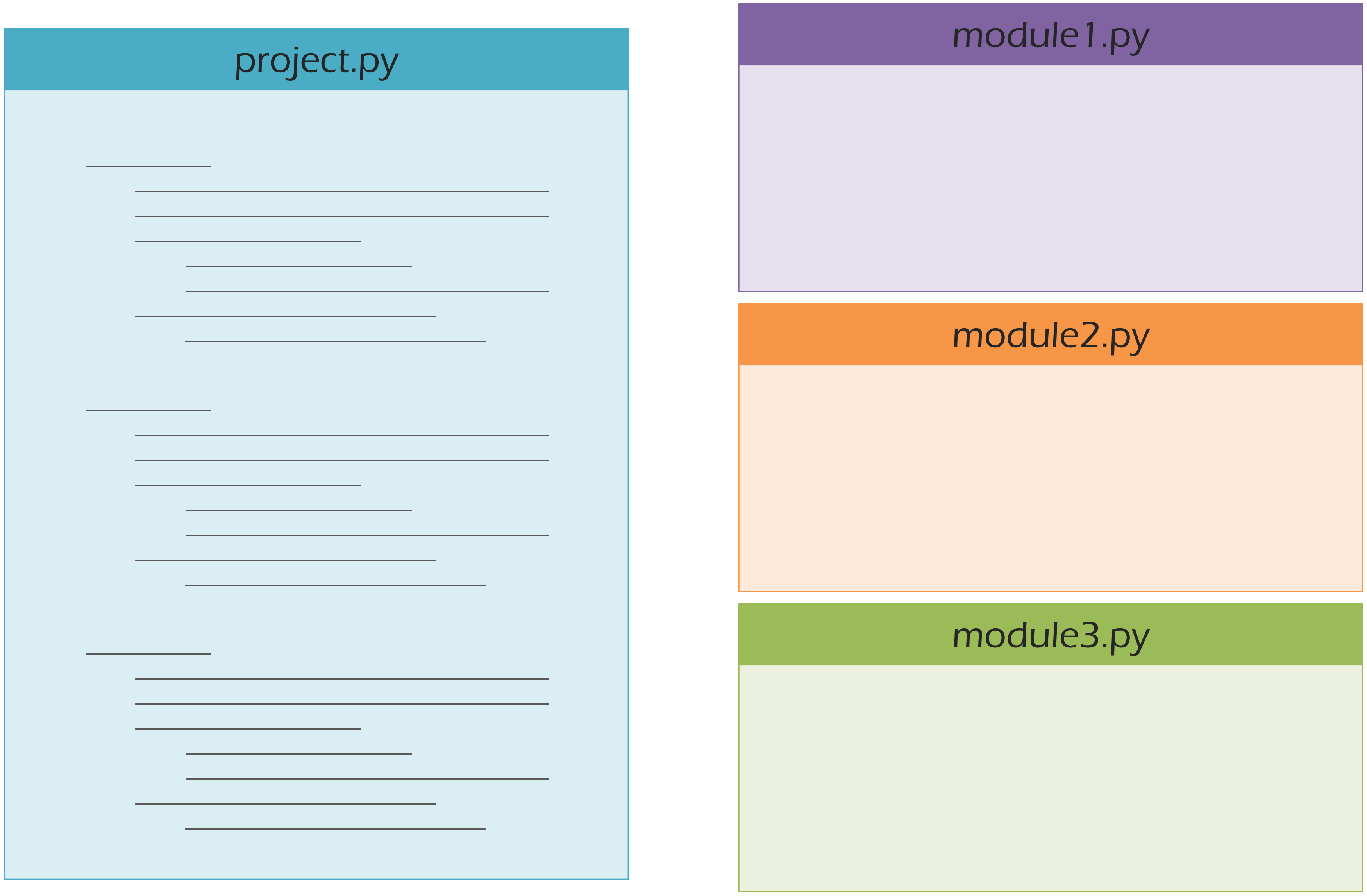
Calling It

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

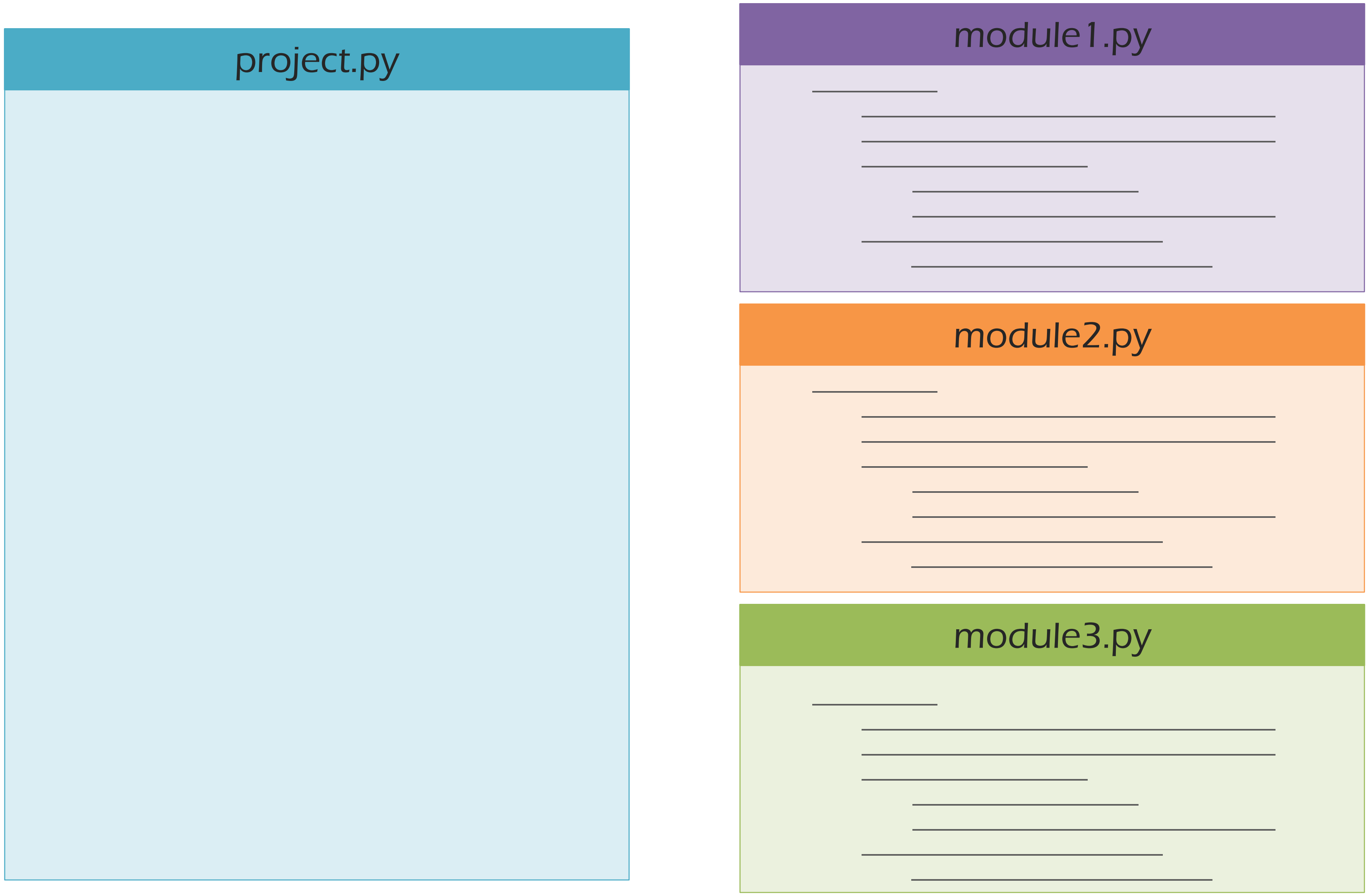
Modules

To make your code more modular

Intro



Intro



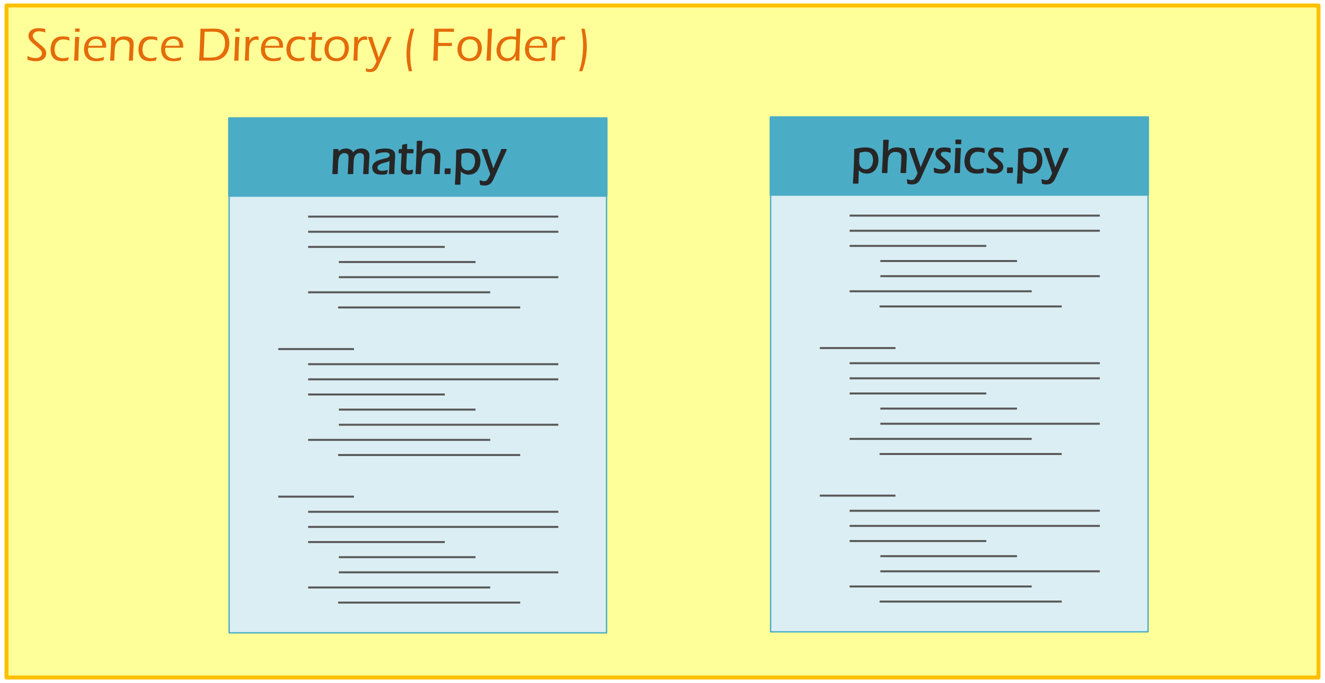
How To

```
from module_name import block_name
```

i.e. from math import tan

Packages

```
from pkge_name.module_name import block_name
```

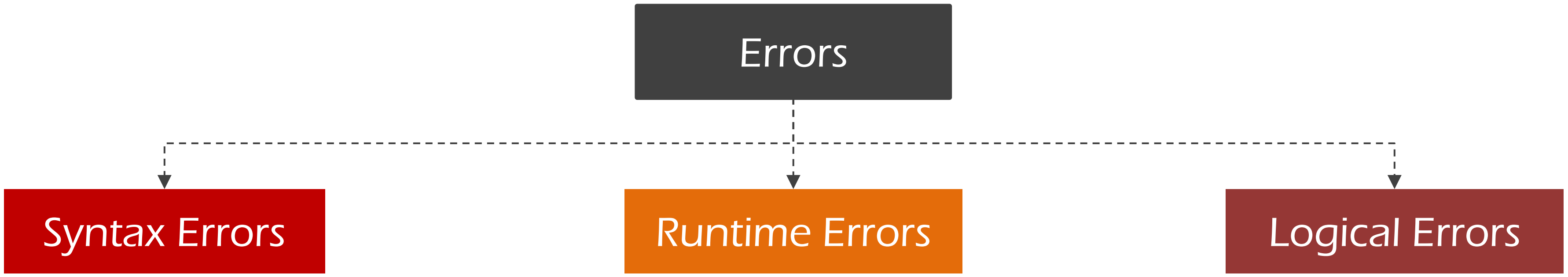


i.e. from science.math import tan

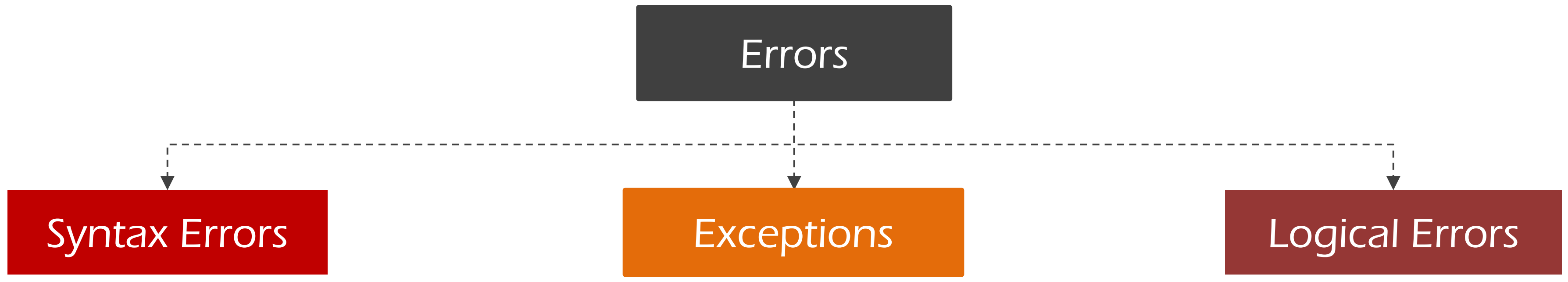
Errors & Exceptions

Gotta catch ‘em all

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:                  ---->  Put the code that you want to handle its exceptions
        doTry()
except:               ---->  Handle the exception if it raised in the try clause
        doExcept()
else:                 ---->  Run when code in try clause run without raising
                             exceptions
        doElse()
finally:              ---->  Put the code that you want to run always if there is
                             an exception or not.
        doFinally()
```

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

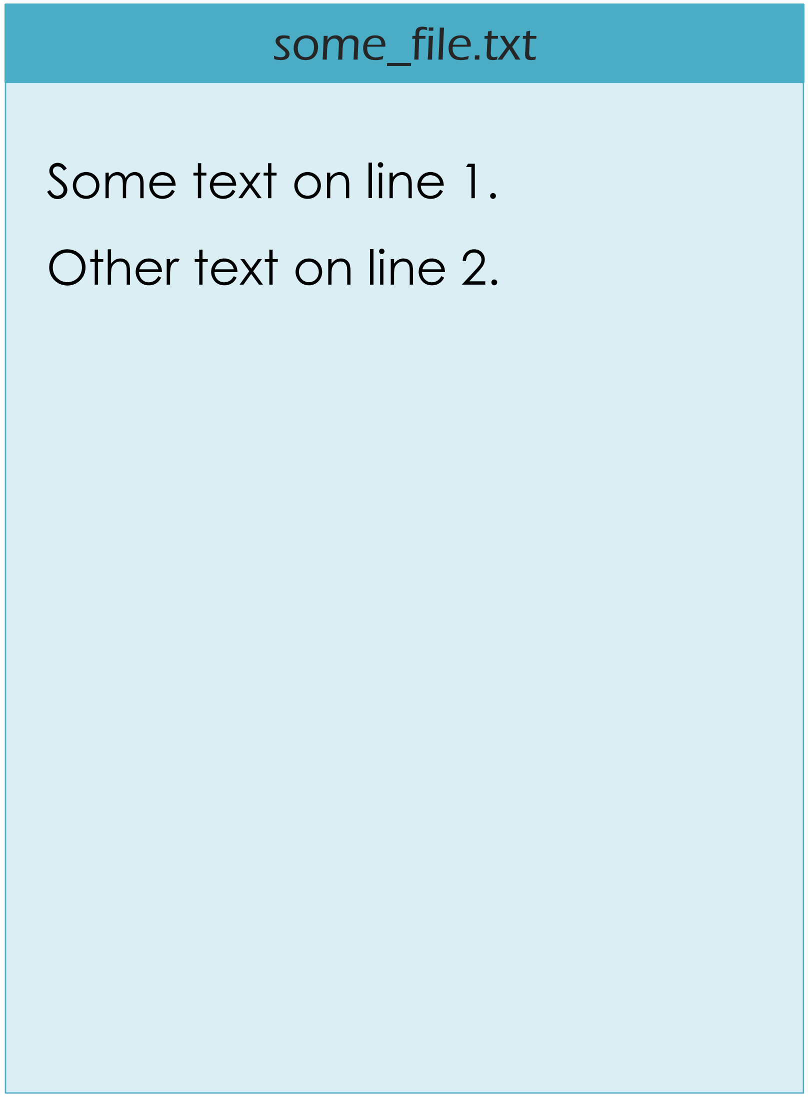
```
fl = open("some_file.txt", 'r')
fl.read()
#output: Some text on line 1.
         Other text on line 2.
fl.read(4)
#output: Some
fl.readline()
#output:  text on line 1.
fl = open("some_file.txt", 'r')
for line in fl:
      print(line)
#output: Some text on line 1.
         Other text on line 2.
```

some_file.txt

```
Some text on line 1.
Other text on line 2.
```

Write on Files

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


some_file.txt

Some text on line 1.

Other text on line 2.

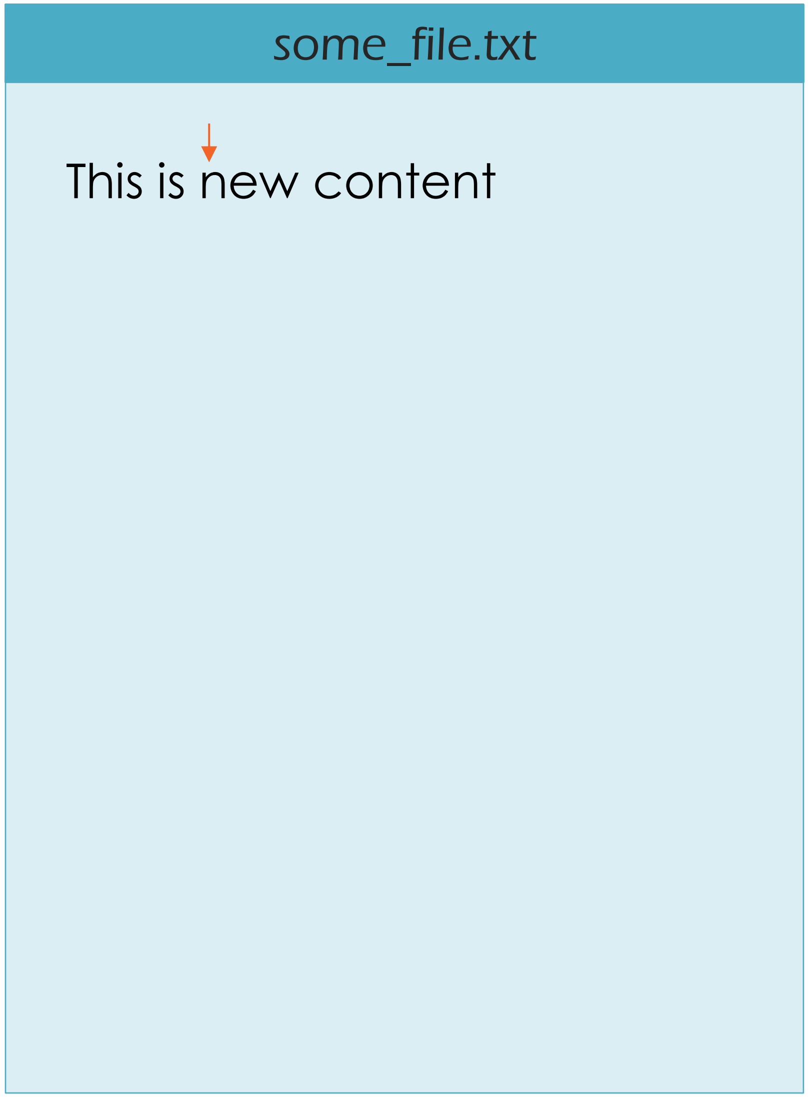
Write on Files

```
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 is appended
```

Python Standard Library

OS

os module provides functions for interacting with the operating system

```
import os

os.getcwd()                              #  /usr/bin/python33

os.system("rmdir dir2")                  #  it will remove dir2

os.chdir("/home/ahmedmoawad")            #  change the dir. to /home/…

os.getlogin()                            #  "Ahmed Moawad"
```

math

math module provides access to the mathematical functions by the C standard

```
import math

math.ceil(3.2)                       #  4

math.floor(3.6)                      #  3

math.sqrt(9)                         #  3

math.pi                              #  3.14
```

re

re provides regular expression matching operations

```
import re

re.match(pattern,string)

#match string with pattern from its starting

re.fullmatch(pattern,string)

#match full string with the pattern

re.search(pattern,string)

#scan the string finding the part that match the pattern
```

External Libraries

pip tool

pip

pip is a package management system used to install and manage software packages written in Python

```
pip install "some library"
```

i.e. pip install libcloud

Tips and Tricks

Sequence Unpacking

```
l = [1,13,3,7]

a,b,c,d = l

# a=1,b=13,c=3,d=7

a,*b,c = l

# a=1,b=[13,3],c=7
```

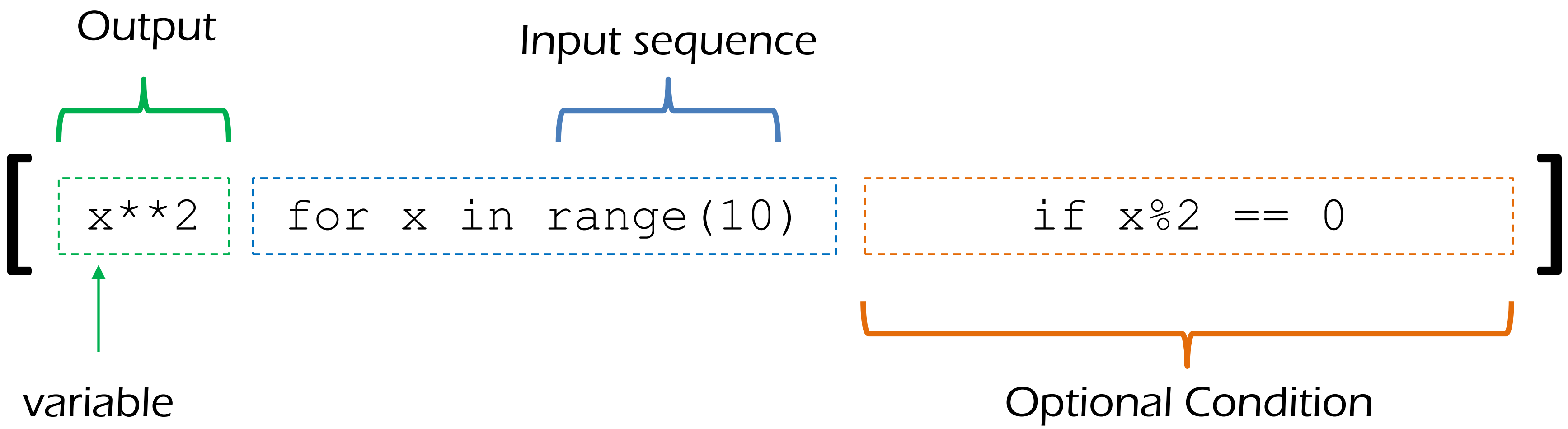
with statement

with statement is used for handling the entry (set-up) and exit (tear-down) tasks for its input

```
with open(“file.txt”, ‘r’) as fp:
    fp.read()
```

List Comprehension

It is an easy method to construct a list



```
L = [ x**2 for x in range(10) if x%2 == 0 ]
#output: [0, 4, 16, 36, 64]
```

enumerate Function

```
languages = [“JavaScript”, “Python”, “Java”]
for i , l in enumerate(languages):
      print(“Element Value: ” , l, end=“, ”)
      print(“Element Index: ” , i)
```

```
Output:
Element Value: JavaScript, Element index: 0
Element Value: Python, Element index: 1
Element Value: Java, Element index: 2
```

all & any

all check if all items in an iterable are truthy value.
any check if one item at least in an iterable is truthy value.

```
L = [0,5,9,7,8]
all(L)                      #False
any(L)                      #True
```

Thank You