Tip1 - line if

```
Do a command if this condition is true else do other command
```

Example

```
canFly = True
bird = "Dove" if canFly else "Penguin"
# bird = "Dove"
```

Tip2 - split and join

```
":".join(["1","Ali","grp"])    # colon is the separator
# '1:Ali:grp'
" ".join("ITI")                # space is the separator
# 'I T I'
"Sara Mohamed".split(" ")      # space is the delimiter
# ["Sara" , "Mohamed"]
"django:flask".split(":")      # colon is the delimiter
# ["django" , "flask"]
```

Tip3 - swapping

Traditional Way

```
x = 4
y = 5
temp = x
x = y
y = temp
```

Python Way

```
x,y = 4,5
x,y = y,x
```

Tip4 - enumerate

```
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
```

Tip5 - is and ==

```
True == 1                    # True

True is 1                    # False

list1 = [1,2,3]

list2 = [1,2,3]

list1 == list2               # True

list1 is list2               # False
```

Tip6 - all and 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.

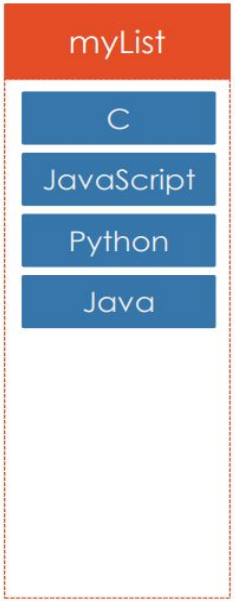
```
L = [0,5,9,7,8]
all(L)                    #False
any(L)                    #True
```

Day 2 Outline

1. List and Dictionary Methods
2. Python Scope
3. Practical Time

Lists- Methods

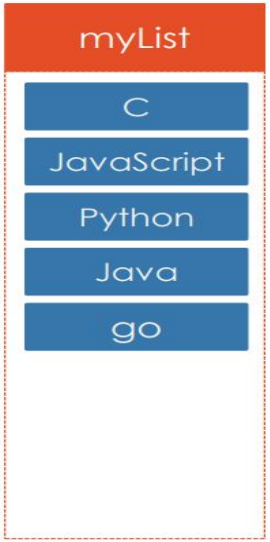
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```




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

Lists- Methods

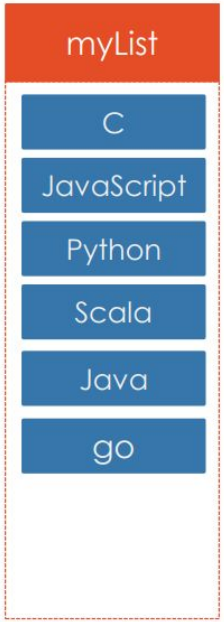
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```




```
myList.pop(4)

myList.append("go")
```

Lists- Methods

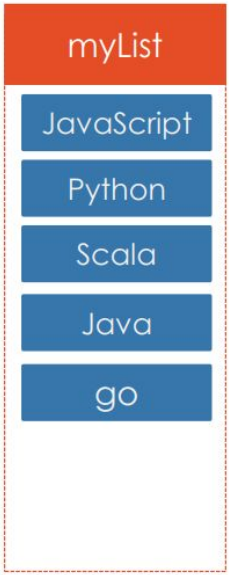
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```




```
myList.pop(4)

myList.append("go")

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

Lists- Methods

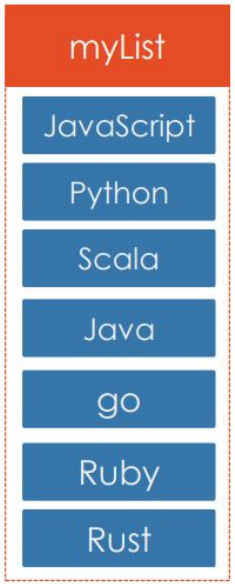
```
myList = ["C", "JavaScript", "Python", "Java", "php"];
```




```
myList.pop(4)

myList.append("go")

myList.insert(3, 'Scala')

myList.remove("C")
```

Lists- 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)
```

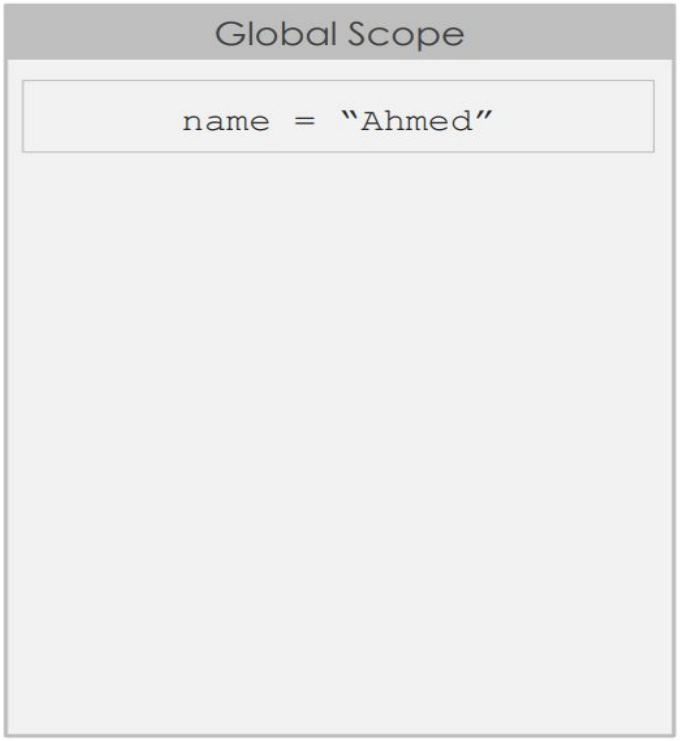
Dictionary - Methods

```
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"}
```

Scope

```
name = "Ahmed"
```

```
Output:
```



Scope

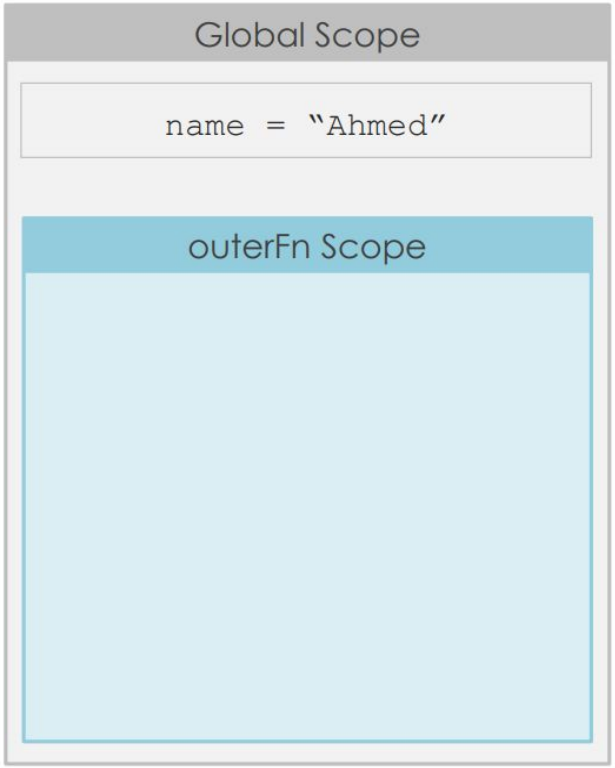
```
name = "Ahmed"
def outerFn():
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
```

Output:

Scope

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

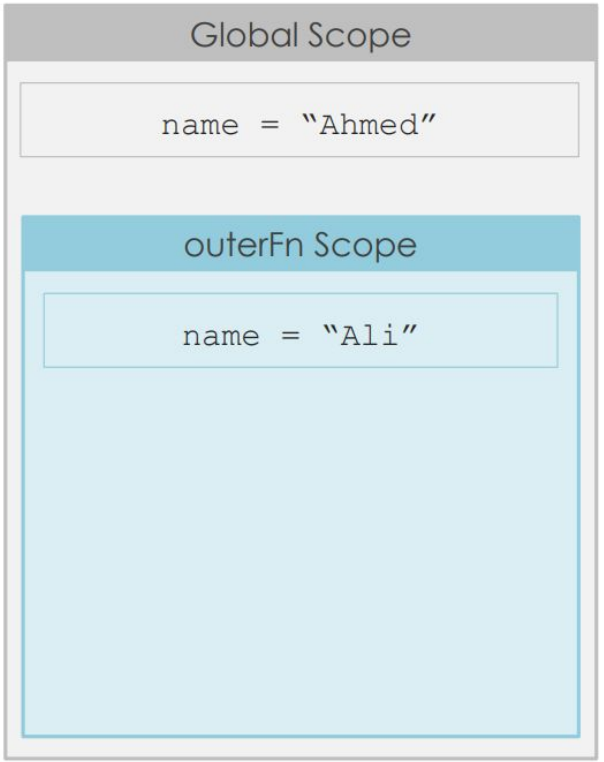
Output:



Scope

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

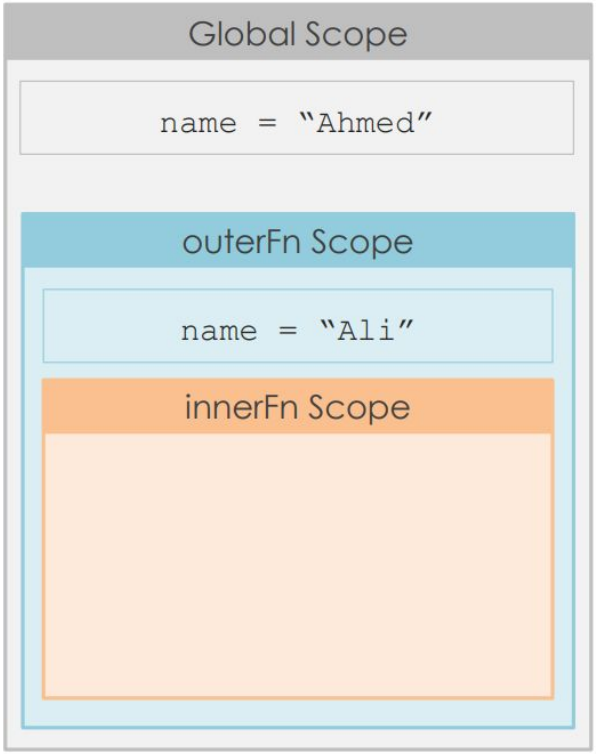
Output:



Scope

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

Output:

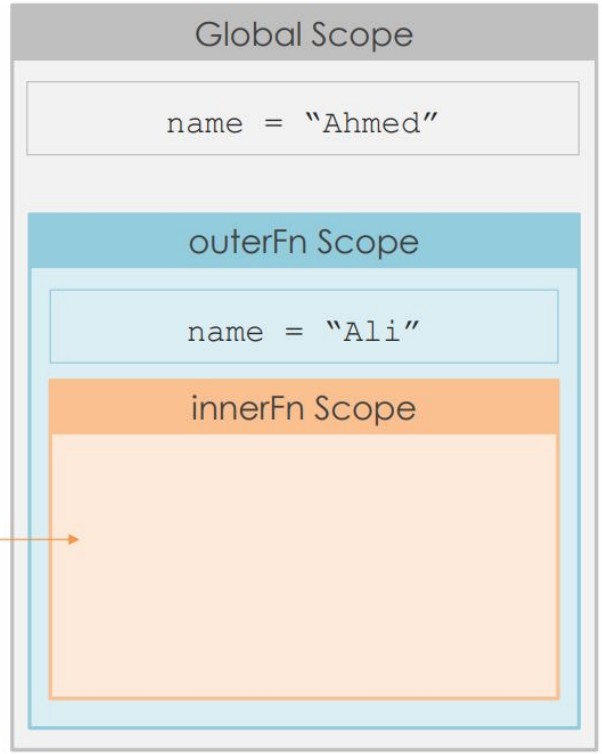


Scope

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

Output:

name
???

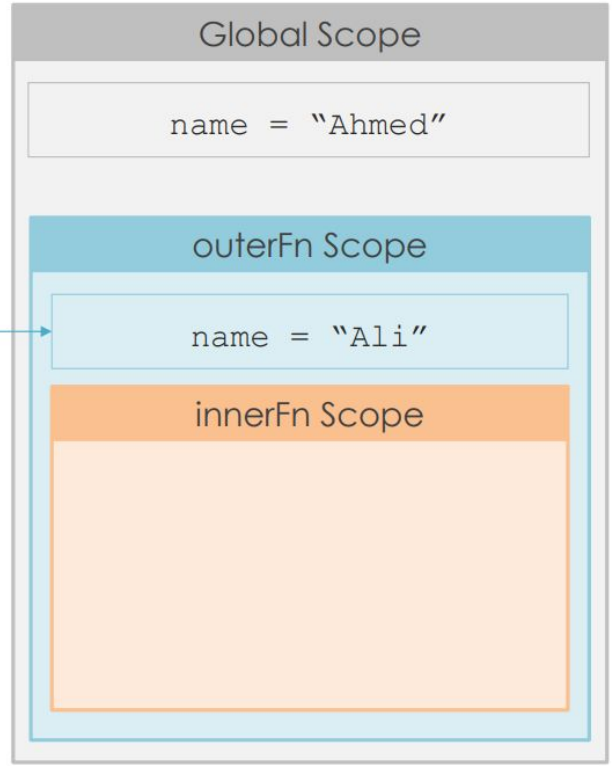


Scope

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

Output:

name
???



Scope

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

Output:

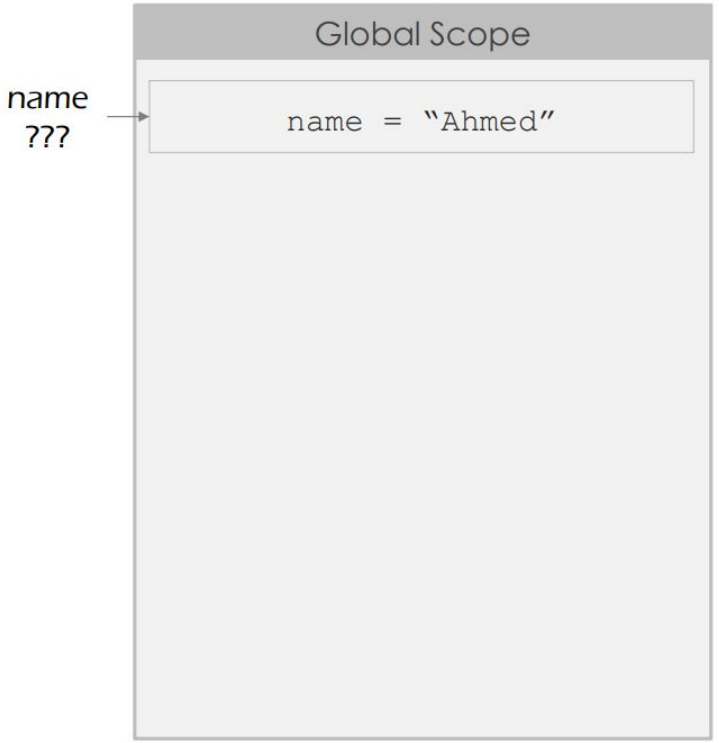
```
Ali
```

Scope

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

Output:

```
Ali
```



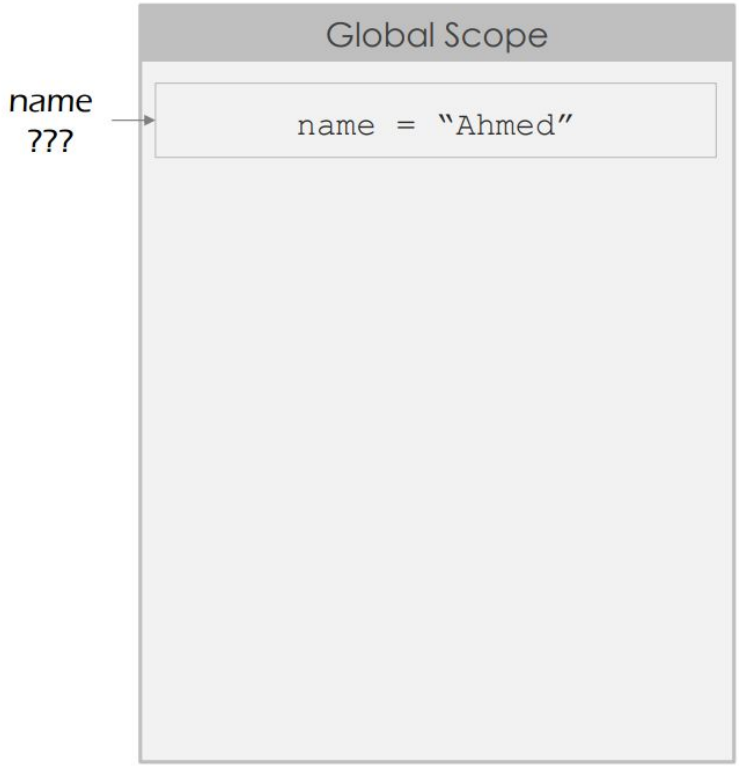
Scope

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

Output:

Ali

Ahmed



Global Scope

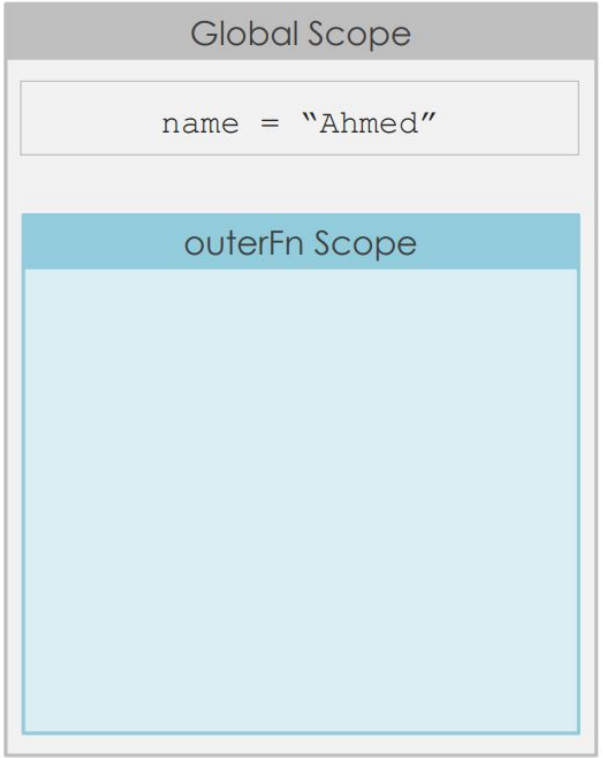
```
name = "Ahmed"
def outerFn():
        global name
        name = "Ali"
        def innerFn():
                print(name)
        innerFn()
outerFn()
```

Output:

Global Scope

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

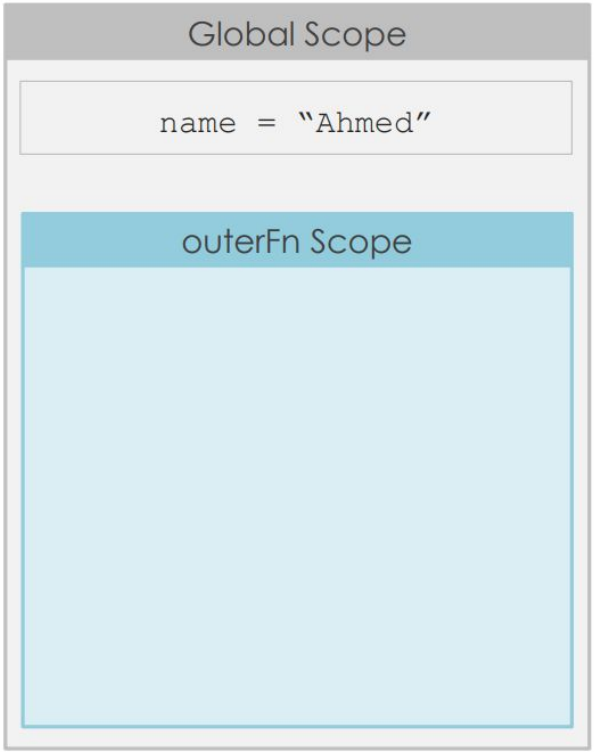
Output:



Global Scope

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

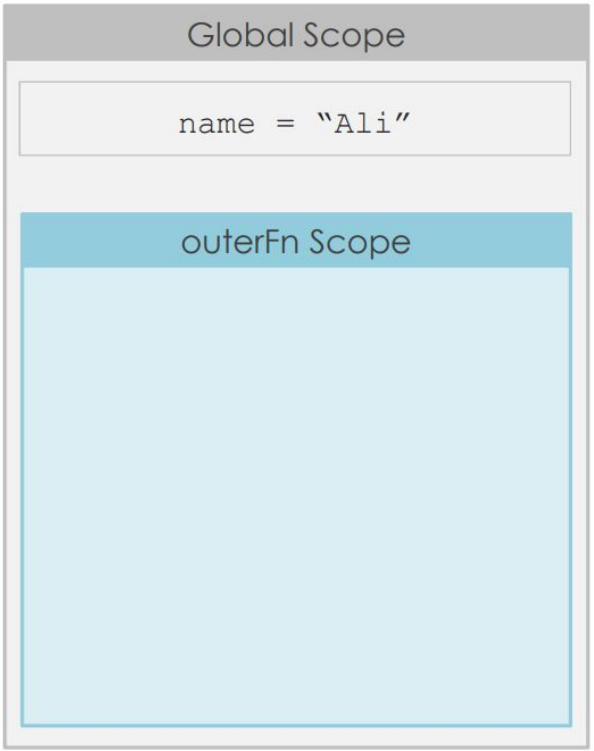
Output:



Global Scope

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

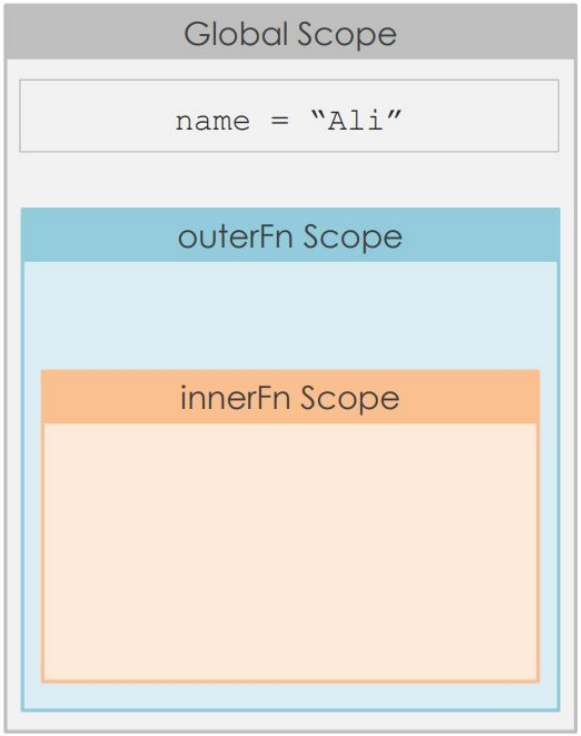
Output:



Global Scope

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

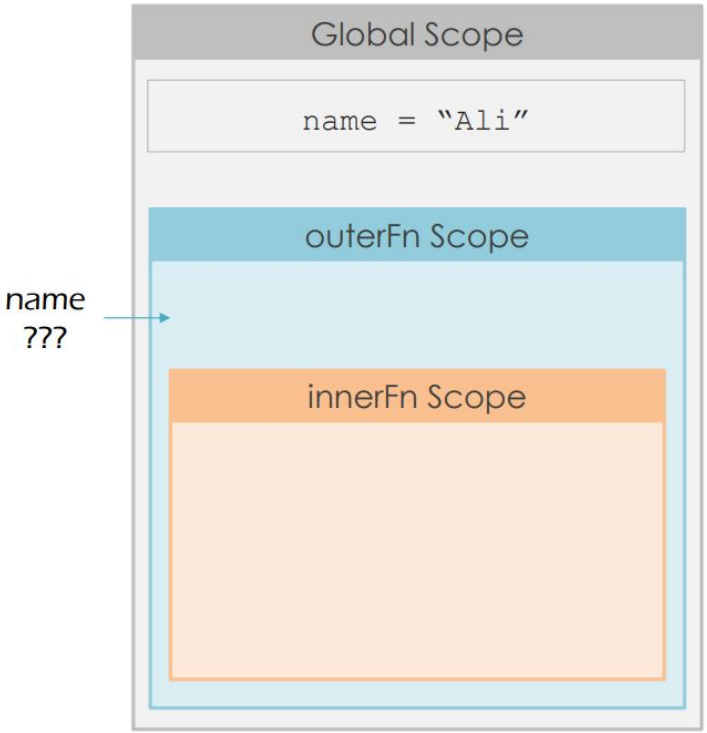
Output:



Global Scope

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

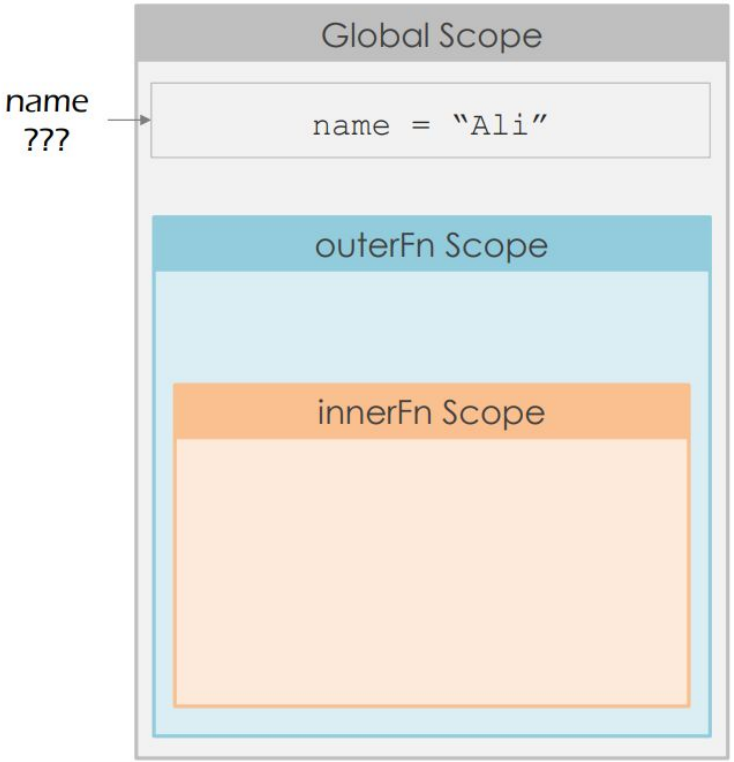
Output:



Global Scope

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

Output:

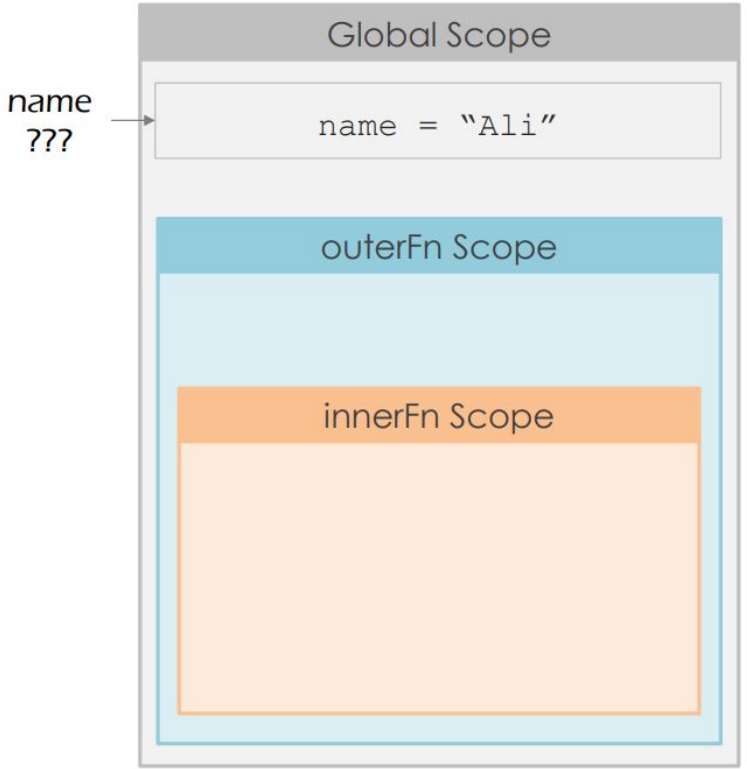


Global Scope

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

Output:

```
Ali
```

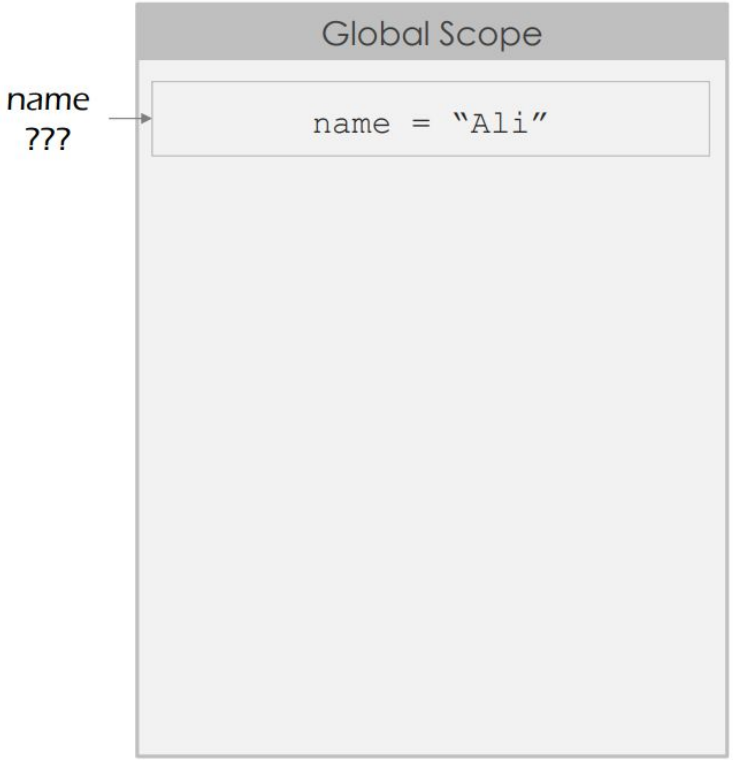


Global Scope

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

Output:

```
Ali
Ali
```



Practical time

1. What is the output of the following code?

```
def outer_fun(a, b):
    def inner_fun(c, d):
        return c + d
    return inner_fun(a, b)

res = outer_fun(5, 10)
print(res)
```

- ○ 15
- ○ Syntax Error
- ○ (5, 10)

2. What is the output of the following function call

```
def fun1(name, age=20):
    print(name, age)

fun1('Emma', 25)
```

- ○ Emma 25
- ○ Emma 20

3. What is the output of the following `display()` function call

```
def display(**kwargs):
    for i in kwargs:
        print(i)

display(emp="Kelly", salary=9000)
```

- TypeError
- Kelly
  9000
- ('emp', 'Kelly')
  ('salary', 9000)
- emp
  salary

5. Select which is true for Python function

- [ ] A Python function can return only a single value
- [ ] A function can take an unlimited number of arguments.
- [ ] A Python function can return multiple values
- [ ] Python function doesn't return anything unless and until you add a return statement

6. Choose the correct function declaration of `fun1()` so that we can execute the following function call successfully

```
fun1(25, 75, 55)
fun1(10, 20)
```

- `def fun1(**kwargs)`
- No, it is not possible in Python
- `def fun1(args*)`
- `def fun1(*data)`

7. Given the following function `fun1()` Please **select all the correct function calls**

```
def fun1(name, age):
    print(name, age)
```

- [ ]
    1. `fun1("Emma", age=23)`
    2. `fun1(age =23, name="Emma")`

- [ ] `fun1(name="Emma", 23)`

- [ ] `fun1(age =23, "Emma")`

8. What is the output of the following `display_person()` function call

```
def display_person(*args):
    for i in args:
        print(i)

display_person(name="Emma", age="25")
```

- TypeError
- Emma
  25
- name
  age

9. What is the output of the following function call

```
def outer_fun(a, b):
    def inner_fun(c, d):
        return c + d

    return inner_fun(a, b)
    return a

result = outer_fun(5, 10)
print(result)
```

- ○ 5
- ○ 15
- ○ (15, 5)
- ○ Syntax Error

10. What is the output of the `add()` function call

```
def add(a, b):
    return a+5, b+5

result = add(3, 2)
print(result)
```

- ○ 15
- ○ 8
- ○ (8, 7)
- ○ Syntax Error

12. What is the output of the following function call

```
def fun1(num):
    return num + 25

fun1(5)
print(num)
```

- 25
- 5
- NameError

Thank you