



Python Programming

Лекц 3

Багш Х.Хулан
Ж.Золжаргал

STRINGS



- Үсэг, тусгай тэмдэгт, хоосон зай, тоо
- 'эсвэл' хашилтанд тэмдэгт мөрийг хавсаргадаг

```
hi = "hello there"
```

- Тэмдэгт мөрүүдийг залгах

```
name = "ana"
```

```
greet = hi + name
```

```
greeting = hi + " " + name
```

- Зарим үйлдэл хийх

```
silly = hi + " " + name * 3
```

STRINGS



- Тэмдэгтүүдийн дараалал
- Харьцуулж чадна `==`, `>`, `<` гэх мэт
- `len()` - функц нь тэмдэгт мөрийн уртыг олно

```
s = "abc"
```

```
len(s) → evaluates to 3
```

STRINGS



- Дөрвөлжин хаалт нь тэмдэгт мөрийн тодорхой индекс / байрлал дахь утгыг авахадашиглагддаг.

```
s = "abc"
```

index: 0 1 2 ← indexing always starts at 0

index: -3 -2 -1 ← last element always at index -1

s[0] → evaluates to "a"

s[1] → evaluates to "b"

s[2] → evaluates to "c"

s[3] → trying to index out of bounds, error

s[-1] → evaluates to "c"

s[-2] → evaluates to "b"

s[-3] → evaluates to "a"

STRINGS



- Тэмдэгт мөрүүдийг **тайрч** болно `[start : stop : step]`
- Хэрэв 2 тоо байвал, `[start : stop]`, `step = 1` анхдагчутга
- Тоонуудыг орхин зөвхөн 2 цэгийг үлдээж болно

```
s = "abcdefgh"
```

```
s[3:6] → evaluates to "def", same as s[3:6:1]
```

```
s[3:6:2] → evaluates to "df"
```

```
s[: :] → evaluates to "abcdefgh", same as s[0:len(s):1]
```

```
s[::-1] → evaluates to "hgfedcba", same as s[-1:- (len(s)+1) : -1]
```

```
s[4:1:-2] → evaluates to "ec"
```

STRINGS



- Strings бол “**immutable**” - утга нь өөрчлөгдөхгүй

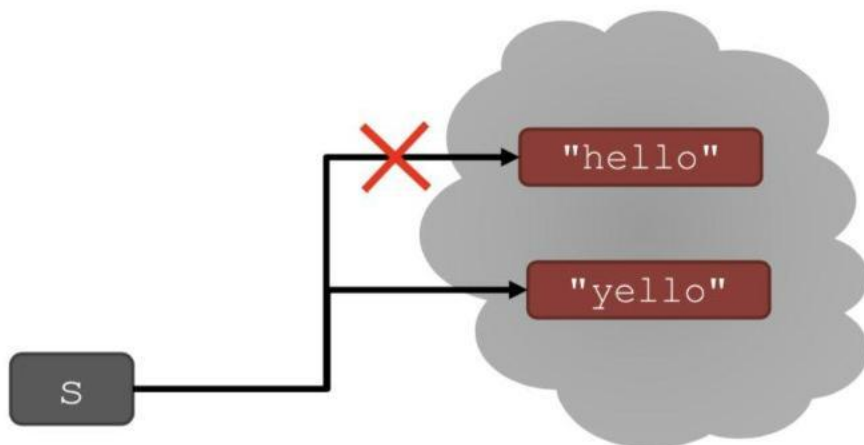
```
s = "hello"
```

```
s[0] = 'y'
```

```
s = 'y'+s[1:len(s)]
```

→ Алдаа өгнө

→ Зөвшөөрнө,
s-г шинэ объекттай
холбоно



for LOOPS



- For давдалт нь давталтын хувьсагчтай бөгөөд олонлогын утгуудаар гүйнэ

```
for var in range(4):  
    <expressions>
```

- Var нь 0, 1, 2, 3 утгуудаар давтана
- Expressions дотор var-н утга бүрт давталтанд юу хийхийг заана

```
for var in range(4, 6):  
    <expressions>
```

- Var нь 4, 5 утгуудаар давтана

- range бол тоогоор давтах арга, гэвч for далталт нь зөвхөн тоогоор давтдаггүй. Зөвхөн тоонууд биш
- **Дурын олонлог дээр давтаж болно**

STRINGS AND LOOPS



- Дараах хоёркодын хэсэг ижил зүйл хийдэг
- Доор байгаань илүү “pythonic”

```
s = "abcdefgh"
```

```
for index in range(len(s)):
```

```
    if s[index] == 'i' or s[index] == 'u':
```


```
        print("There is an i or u")
```

```
for char in s:
```

```
    if char == 'i' or char == 'u':
```

```
        print("There is an i or u")
```


CODE EXAMPLE: robot cheerleaders



```
an_letters = "aefhilmnorsxAEFHILMNORSX"
```

```
word = input("I will cheer for you! Enter a word: ")
times = int(input("Enthusiasm level (1-10): "))
```

```
i = 0
while i < len(word):
    char = word[i]
    if char in an_letters:
        print("Give me an " + char + "! " + char)
    else:
        print("Give me a  " + char + "! " + char)
    i += 1
print("What does that spell?")
for i in range(times):
    print(word, "!!!")
```

for char in word:




EXERCISE



```
s1 = "mit u rock"
s2 = "i rule mit"
if len(s1) == len(s2):
    for char1 in s1:
        for char2 in s2:
            if char1 == char2:
                print("common letter")
                break
```

JOIN



```
# .join() with lists
numList = ['1', '2', '3', '4']
separator = ', '
print(separator.join(numList))

# .join() with tuples
numTuple = ('1', '2', '3', '4')
print(separator.join(numTuple))

s1 = 'abc'
s2 = '123'

# each element of s2 is separated by s1
# '1'+ 'abc'+ '2'+ 'abc'+ '3'
print('s1.join(s2):', s1.join(s2))

# each element of s1 is separated by s2
# 'a'+ '123'+ 'b'+ '123'+ 'b'
print('s2.join(s1):', s2.join(s1))
```

```
1, 2, 3, 4
1, 2, 3, 4
s1.join(s2): 1abc2abc3
s2.join(s1): a123b123c
```

STRIP



```
string = '  xoxo love xoxo  '

# Leading and trailing whitespaces are removed
print(string.strip())

# All <whitespace>,x,o,e characters in the left
# and right of string are removed
print(string.strip(' xoe'))

# Argument doesn't contain space
# No characters are removed.
print(string.strip('stx'))

string = 'android is awesome'
print(string.strip('an'))
```

```
xoxo love xoxo
lov
    xoxo love xoxo
droid is awesome
```

SPLIT



```
text= 'Love thy neighbor'
```

```
# splits at space
```

```
print(text.split())
```

```
grocery = 'Milk, Chicken, Bread'
```

```
# splits at ','
```

```
print(grocery.split(', '))
```

```
# Splitting at ':'
```

```
print(grocery.split(':'))
```

```
['Love', 'thy', 'neighbor']  
['Milk', 'Chicken', 'Bread']  
['Milk, Chicken, Bread']
```

PARTITION



```
string = "Python is fun"

# 'is' separator is found
print(string.partition('is '))

# 'not' separator is not found
print(string.partition('not '))

string = "Python is fun, isn't it"

# splits at first occurrence of 'is'
print(string.partition('is'))
```

```
('Python ', 'is ', 'fun')
('Python is fun', '', '')
('Python ', 'is', " fun, isn't it")
```

GOOD PROGRAMMING



- Илүү их код нь сайн гэсэн үг биш
- Сайн програмистуудыг функцын хэмжээгээр нь үнэлж болно
- Задрал болон хийсвэрлэлд хүрэх механизм
-

EXAMPLE - PROJECTOR



- Проектор бол хар хайрцаг
- Энэ хэрхэн ажилладагыг мэдэхгүй
- Интерфэйсийг мэднэ: input/output
- Бусад электрон төхөөрөмжтэй тэдгээр input-ээр холбогдоно
- Хар хайрцаг нь дүрсийг ямар нэгэн байдлаар оролтын эх үүсвэрээс хананд хөрвүүлж, томруулдаг
- **ХИЙСВЭРЛЭХ САНАА**: Проекторыг ашиглахын тулд үүнийг яаж ажилладагыг мэдэх шаарадлагагүй.

EXAMPLE - PROJECTOR



- Том дүрсүүдийг тусад нь преокторуудад даалгавар болгон задлах
- Преоктор бүр оролт авч тусдаа гаралтыг гаргадаг
- Преокторууд хамтран ажиллаж, илүү том дүрс гаргах болно
- **ЗАДЛАХ САНАА:** Эцсийн зорилгод хүрэхийн тулд ялгаатай төхөөрөмжүүд хамтран ажилладаг.

Create structure with DECOMPOSITION



- Преоктортой жишээнд, тусдаа төхөөрөмжүүд
- Програмчлалд, кодыг модулиудад хуваах
 - Бие даасан байдал
 - Кодыг задлахад ашигладаг
 - Дахин ашиаглагдах боломжтой байдлаар төлөвлөх
 - Кодыг цэгцтэй байлгах
 - Кодыг уялдаатай байлгах
- Энэ лекцээр кодыг функцээр задлах
- Цаашид кодыг классаар задлах

Supress details with ABSTRACTION



- Проекторын жишээнд, яаж хэрэглэх нь байхаас, яаж хийсэн нь байхгүй
- Програмчлалд, хэсэг кодыг хар хайрцаг шиг төсөөлнө
 - Дэлгэрэнгүйг харахгүй
 - Дэлгэрэнгүйг харах хэрэггүй
 - Дэлгэрэнгүйг харахыг хүсэхгүй
 - Кодын дэлгэрэнгүй хэсгийг нууна
- **Функцын тодорхойлолтоор** хийсвэрлэлийг гүйцэтгэнэ

FUNCTIONS



- Дахин хэрэглэгдэх хэсэг код, функц гэж дуудна
- Функц нь програмд дуудагдах хүртэлээ ажиллахгүй
- Функцын шинж чанар:
 - **Нэртэй**
 - **Параметртэй** (0 эсвэл олон)
 - **Тайлбартай** (заавал биш, гэвч шаардлагатай)
 - **Биетэй**
 - Ямар нэг зүйл **буцаана**

How to write and call FUNCTION

keyword

name

parameters
or arguments

specification,
docstring

def

is_even

(i) :

"""

Input: i, a positive int

Returns True if i is even, otherwise False

"""

body

print("inside is_even")

return i%2 == 0

is_even(3)

later in the code, you call the
function using its name and
values for parameters

In the Function BODY



```
def is_even( i ):
    """
    Input: i, a positive int
    Returns True if i is even, otherwise False
    """
```

```
print("inside is_even")
```

```
return i%2 == 0
```

run some
commands

expression to
evaluate and return

Expression to
evaluate and return

keyword

Variable SCOPE



- Албан ёсны параметр нь функц дуудагдах үед бодит параметртэй холбогддог.
- Функц рүү ороход шинэ scope/frame/enviroment үүсдэг.
- Scope бол хувьсагчийн цар хүрээ

```
def f( x ) :  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

*formal
parameter*

*Function
definition*

```
x = 3  
z = f( x )
```

*actual
parameter*

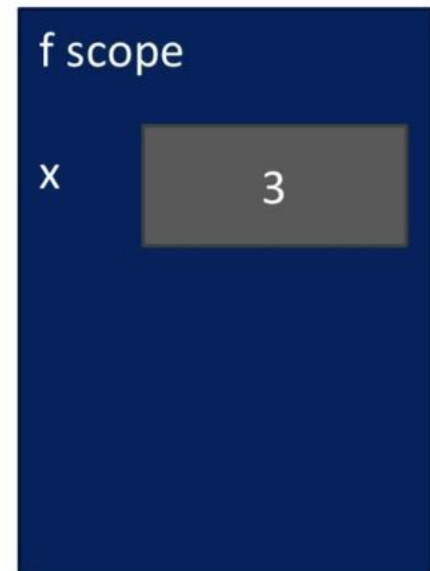
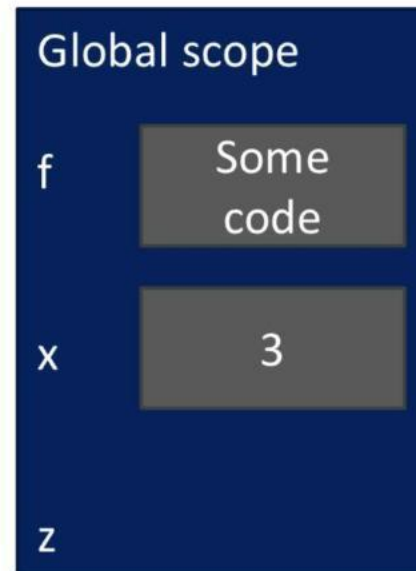
Main program code
* initializes a variable x
* makes a function call f(x)
* assigns return of function to variable z

Variable SCOPE



```
def f( x ) :  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

```
x = 3  
z = f( x )
```

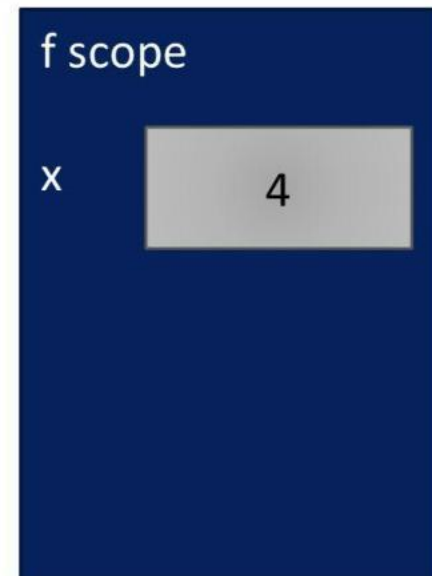


Variable SCOPE



```
def f( x ) :  
    x = x + 1  
    print('in f(x): x =', x)  
    return x
```

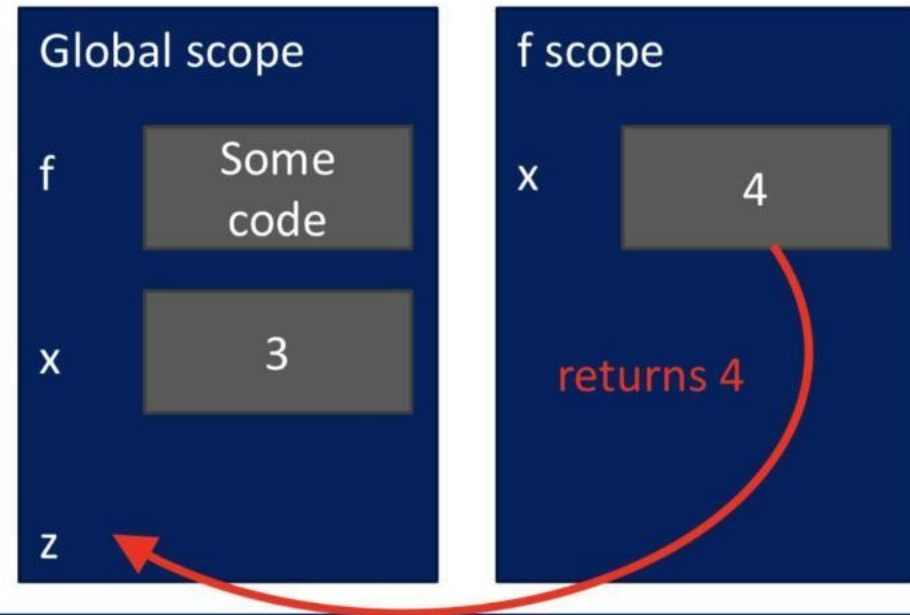
```
x = 3  
z = f( x )
```



Variable SCOPE



```
def f( x ):  
    x = x + 1  
    print('in f(x): x =', x)  
    return x  
  
x = 3  
z = f( x )
```

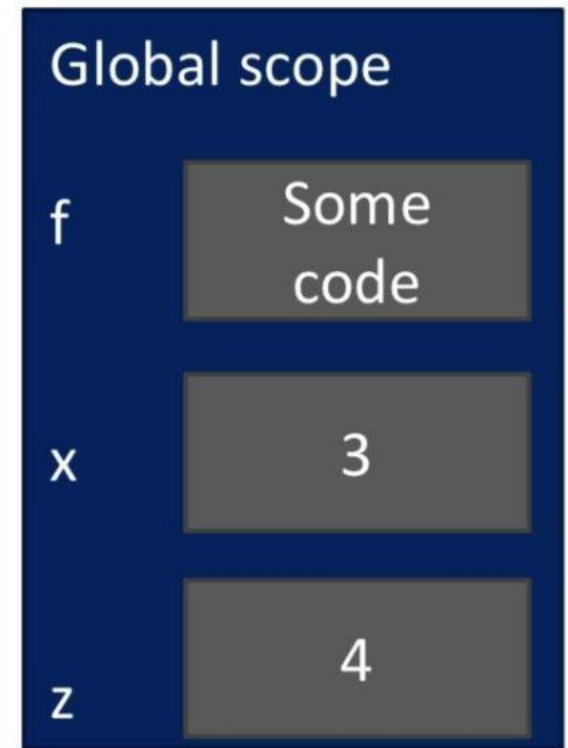


Variable SCOPE



```
def f( x ) :  
    x = x + 1  
    print('in f(x) : x =', x)  
    return x
```

```
x = 3  
z = f( x )
```



ONE WARNING IF NO return STATEMENT

```
def is_even( i ):
```

```
    """
```

```
    Input: i, a positive int
```

```
    Does not return anything
```

```
    """
```

```
    i%2 == 0
```

*without a return
statement*

-

- Хэрэг буцаах утга өгөхгүй бол паятон нь **None** утга буцаана. Утга байхгүйг илэрхийлнэ.

RETURN

VS

print



- Return нь функц доторх үйлдэл
- Нэг л удаа утга буцаана
- Функц доторх return-ны дахаах код биелэгдэхгүй
- Функц дуудсан газар зохих утгаа авна

- Print нь функцээс гаднах үйлдэл
- Олон удаа хэвлэж болно
- Функц доторх print-ийн дараах код биелэгдэнэ
- Консол руу гаралт хийнэ

FUNCTIONS AS ARGUMENTS



- Аргумент нь ямарч төрөлтэй, мөн функц болно

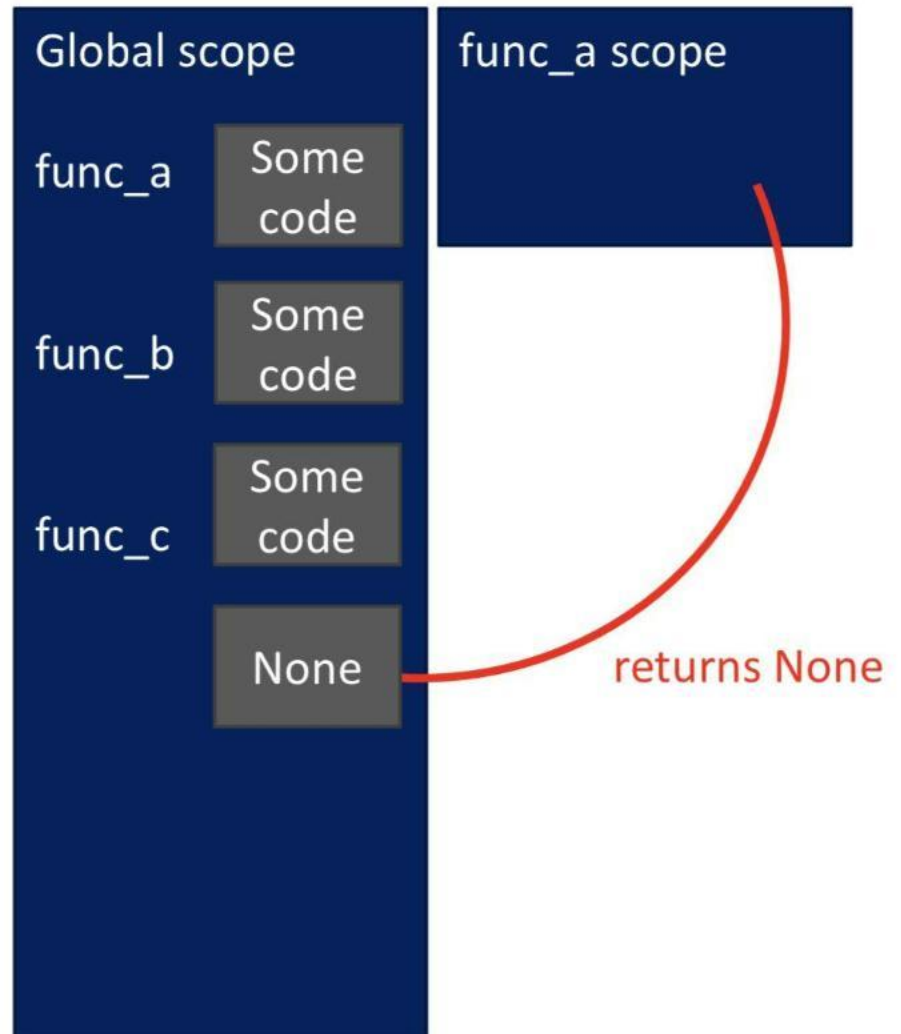
```
def func_a():  
    print 'inside func_a'  
def func_b(y):  
    print 'inside func_b'  
    return y  
def func_c(z):  
    print 'inside func_c'  
    return z()  
  
print func_a()  
print 5 + func_b(2)  
print func_c(func_a)
```

call func_a, takes no parameters
call func_b, takes one parameter
call func_c, takes one parameter, another function

FUNCTIONS AS ARGUMENTS

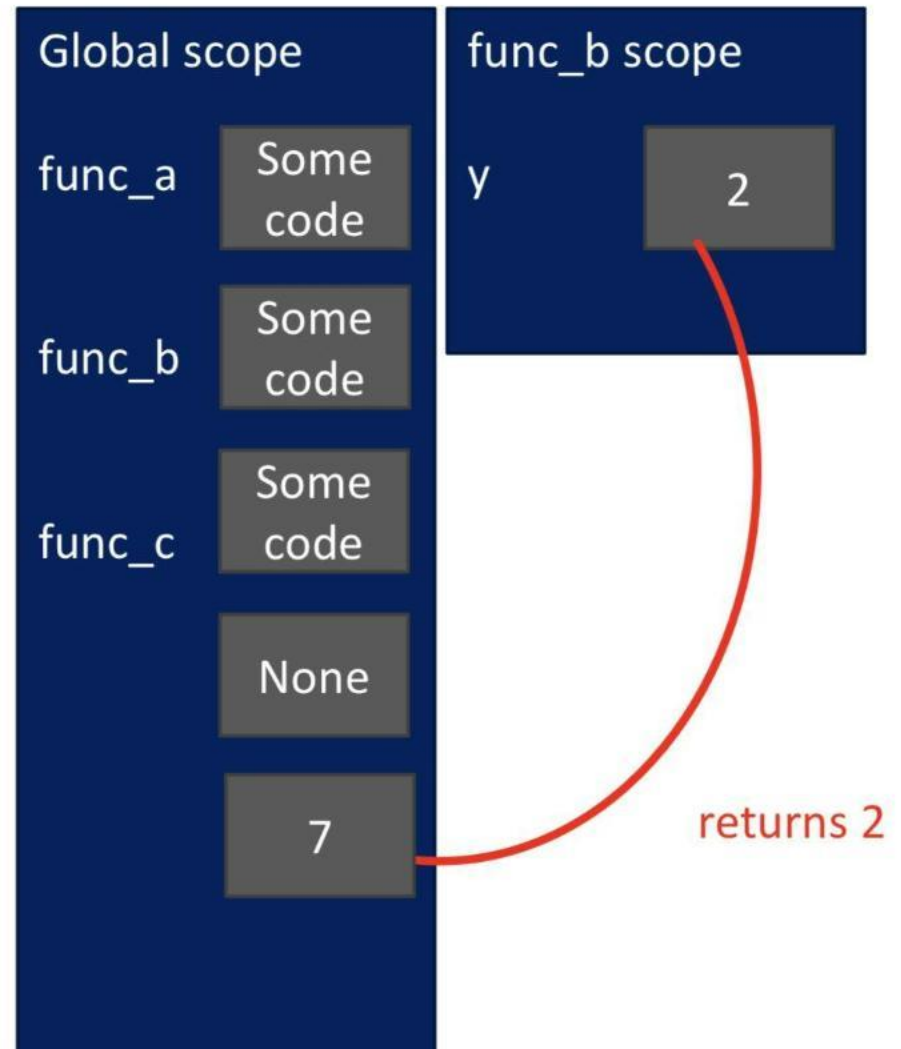


```
def func_a():  
    print 'inside func_a'  
def func_b(y):  
    print 'inside func_b'  
    return y  
def func_c(z):  
    print 'inside func_c'  
    return z()  
print func_a()  
print 5 + func_b(2)  
print func_c(func_a)
```



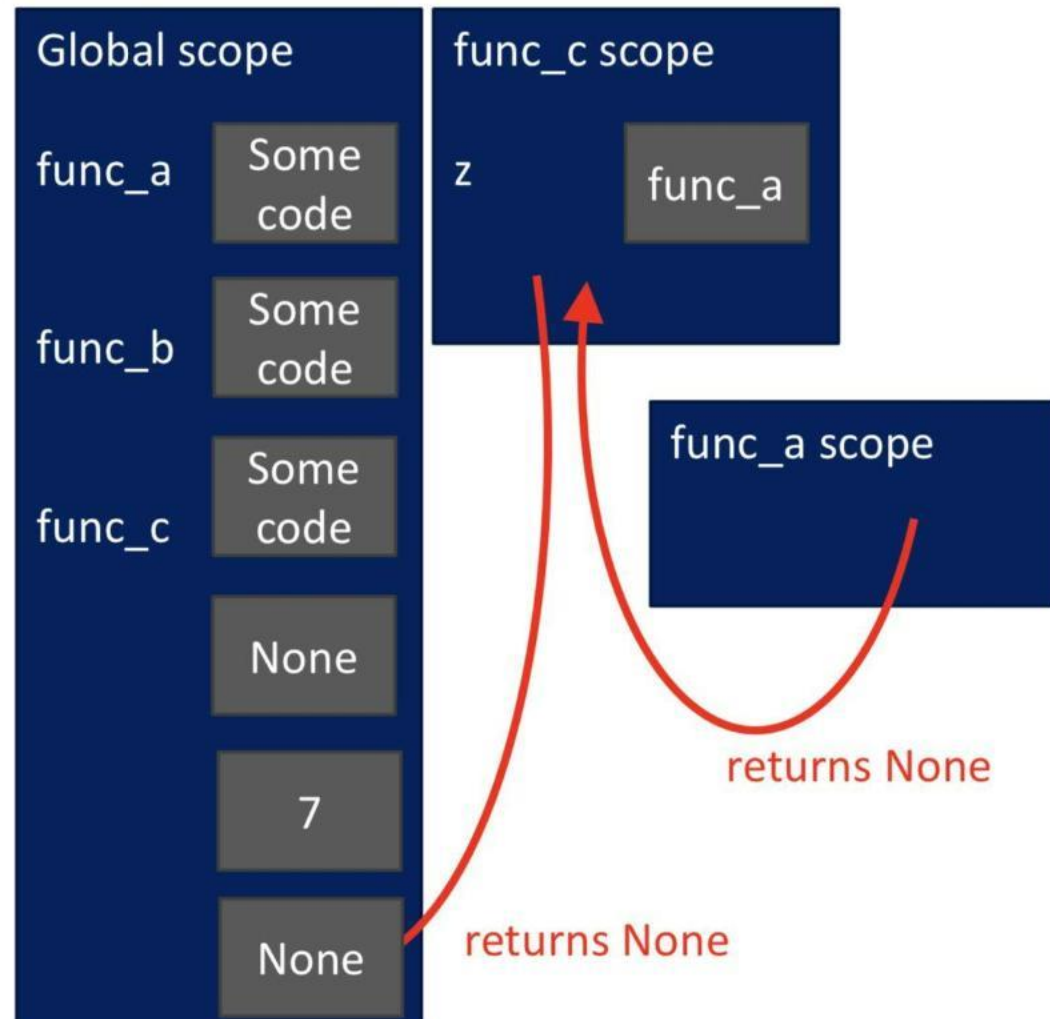
FUNCTIONS AS ARGUMENTS

```
def func_a():  
    print 'inside func_a'  
def func_b(y):  
    print 'inside func_b'  
    return y  
def func_c(z):  
    print 'inside func_c'  
    return z()  
print func_a()  
print 5 + func_b(2)  
print func_c(func_a)
```



FUNCTIONS AS ARGUMENTS

```
def func_a():  
    print 'inside func_a'  
  
def func_b(y):  
    print 'inside func_b'  
    return y  
  
def func_c(z):  
    print 'inside func_c'  
    return z()  
  
print func_a()  
print 5 + func_b(2)  
print func_c(func_a)
```



SCOPE EXAMPLE

- Гадна тодорхойлогдсон хувьсагч руу **хандана**
- Гадна тодорхойлсон глобал хувьсагчийг **ашиглана**,
өөрчлөхгүй

```
def f(y):  
    x = 1  
    x += 1  
    print(x)  
  
x = 5  
f(x)  
print(x)
```

*x is re-defined
in scope of f*

*different x
objects*

```
def g(y):  
    print(x)  
    print(x + 1)  
  
x = 5  
g(x)  
print(x)
```

*x from
outside g*

*x inside g is picked up
from scope that called
function g*

```
def h(y):  
    x += 1  
  
x = 5  
h(x)  
print(x)
```

*UnboundLocalError: local variable
'x' referenced before assignment*

SCOPE EXAMPLE

- Гадна тодорхойлогдсон хувьсагч руу **хандана**
- Гадна тодорхойлсон глобал хувьсагчийг **ашиглана**,
өөрчлөхгүй

```
def f(y):  
    x = 1  
    x += 1  
    print(x)
```

```
x = 5  
f(x)  
print(x)
```

```
def g(y):  
    print(x)
```

```
x = 5  
g(x)  
print(x)
```

```
def h(y):  
    x += 1
```

```
x = 5  
h(x)  
print(x)
```

*x from
global/main
program scope*

SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x
```

Some code

```
x = 3  
z = g(x)
```

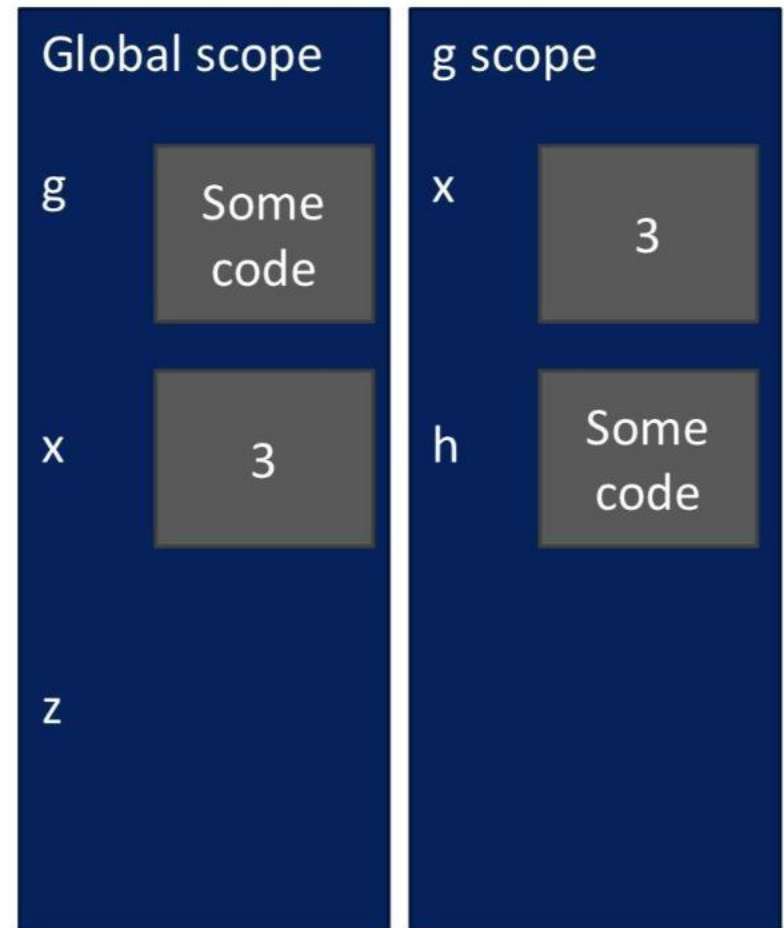


SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x
```

```
x = 3  
z = g(x)
```



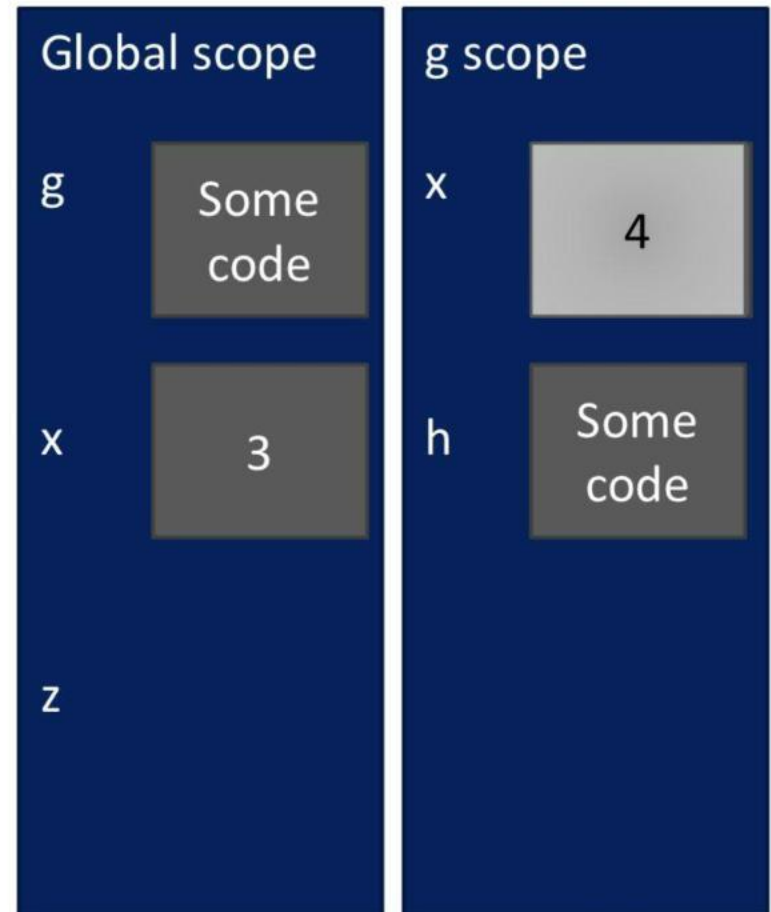
SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x
```

```
x = 3
```

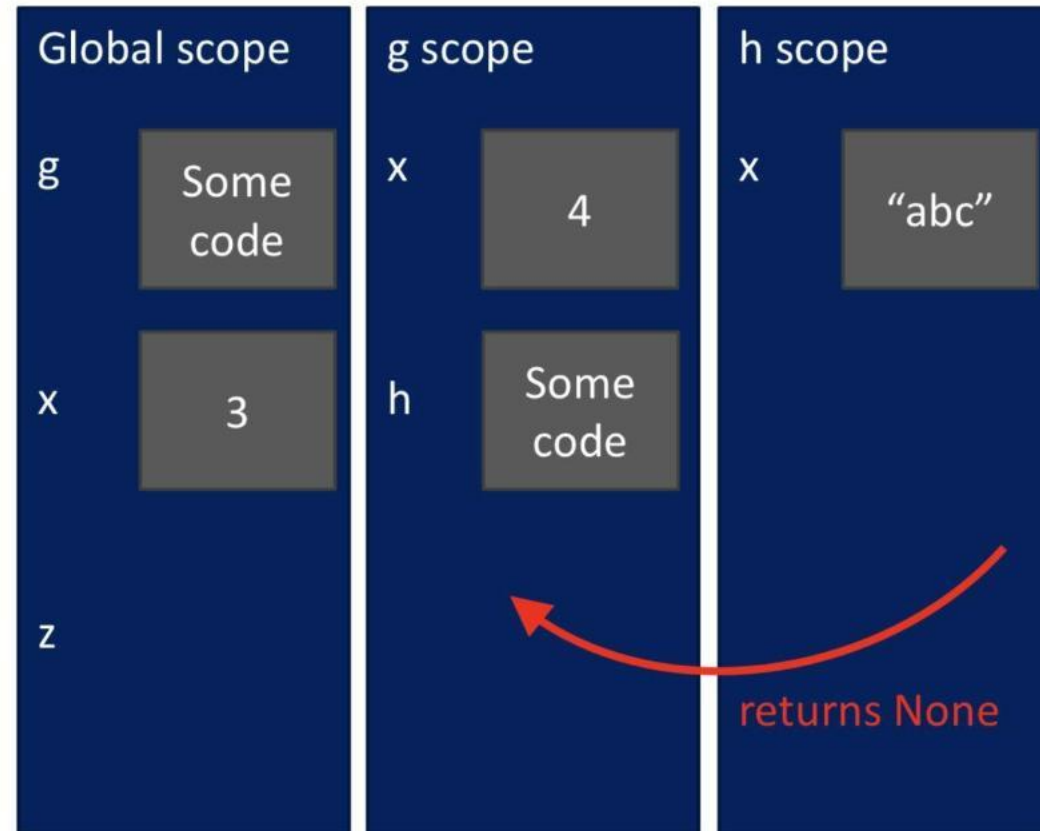
```
z = g(x)
```



SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x  
  
x = 3  
z = g(x)
```

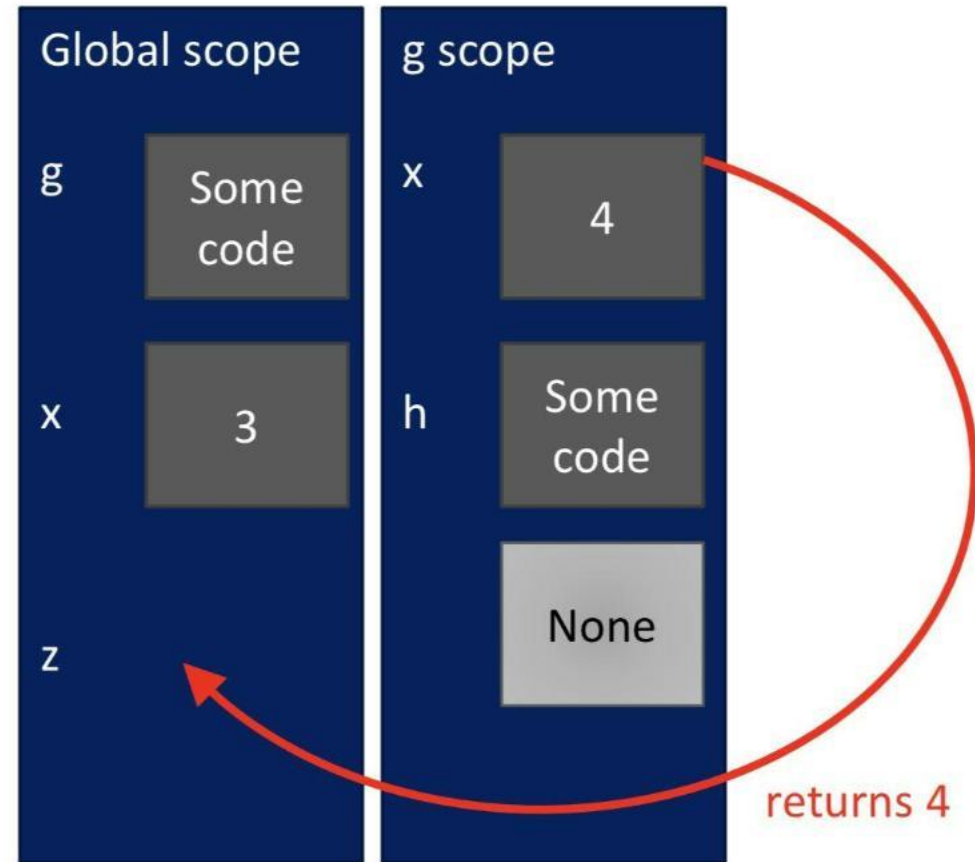


SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x
```

```
x = 3  
z = g(x)
```



SCOPE DETAILS



```
def g(x):  
    def h():  
        x = 'abc'  
    x = x + 1  
    print('g: x =', x)  
    h()  
    return x
```

```
x = 3
```

```
z = g(x)
```



DECOMPOSITION & ABSTRACTION



- Хамтдаа хүчтэй
- Кодыг олон удаа хэрэглэгч боловч нэг л удаа дебаг хийгдэнэ.
- `*args` болон `**kwargs`