Python Programming

Лекц 4

Багш Ж.Золжаргал
Х.Хулан

GOOD PROGRAMMING

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

EXAMPLE - PROJECTOR

- Прожектор 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: Прожекторыг 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 ашиглагдах боломжтой байдлаар төлөвлөх
  - Кодыг цэгцтэй байлгах
  - Кодыг уялдаатай байлгах
- Энэ лекцээр кодыг функцээр задлах
- Цаашид кодыг классаар задлах

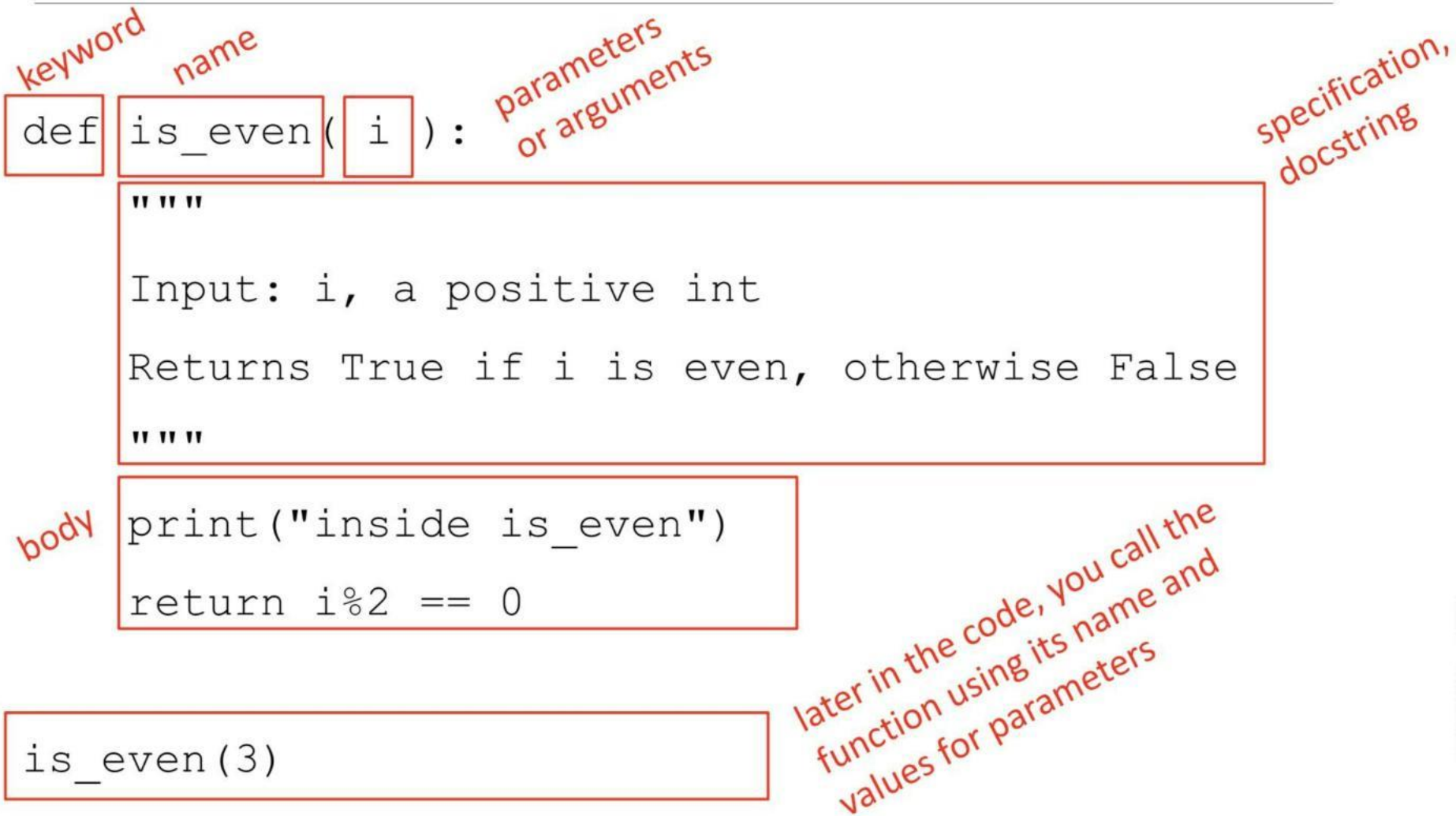
Supress details with ABSTRACTION

- 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 программ 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



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

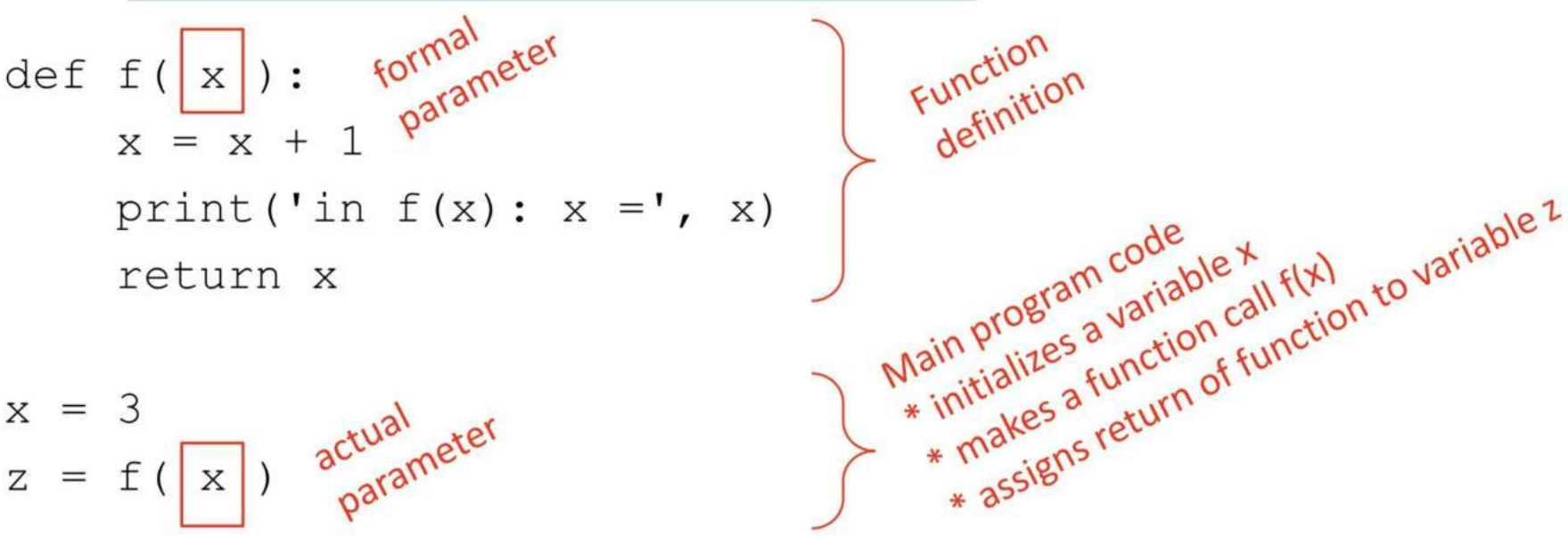
keyword

expression to evaluate and return

Expression to evaluate and return

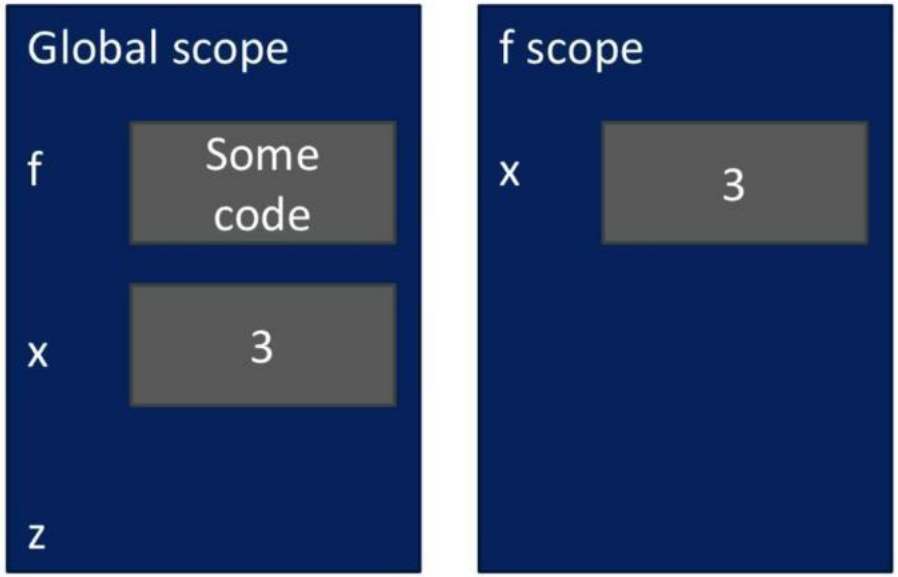
Variable SCOPE

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



Variable SCOPE

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

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

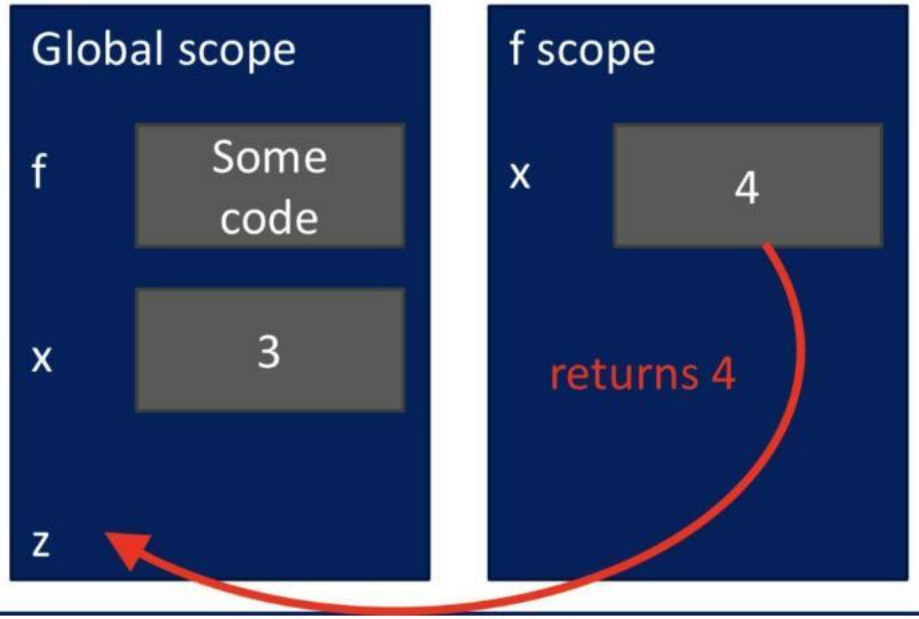


Variable SCOPE

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

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

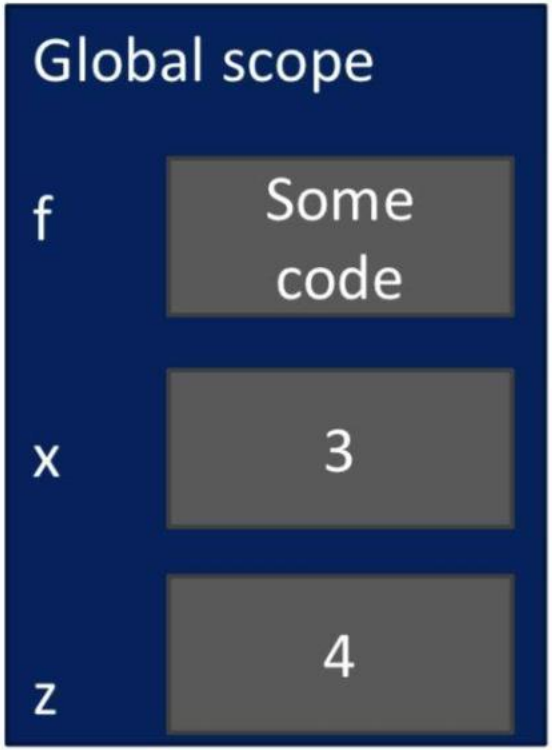
Variable SCOPE

```
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 дараах 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)
```

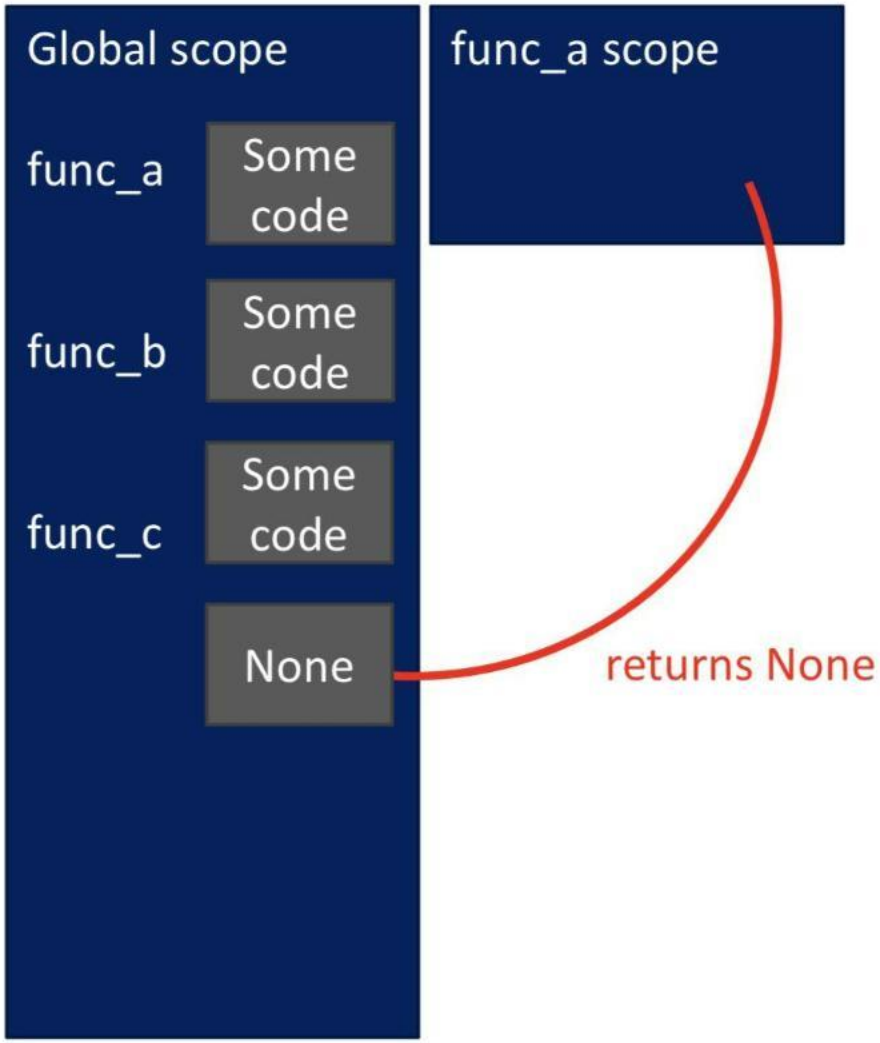
call `func_a`, takes no parameters

call `func_b`, takes one parameter

call `func_c`, takes one parameter, another function

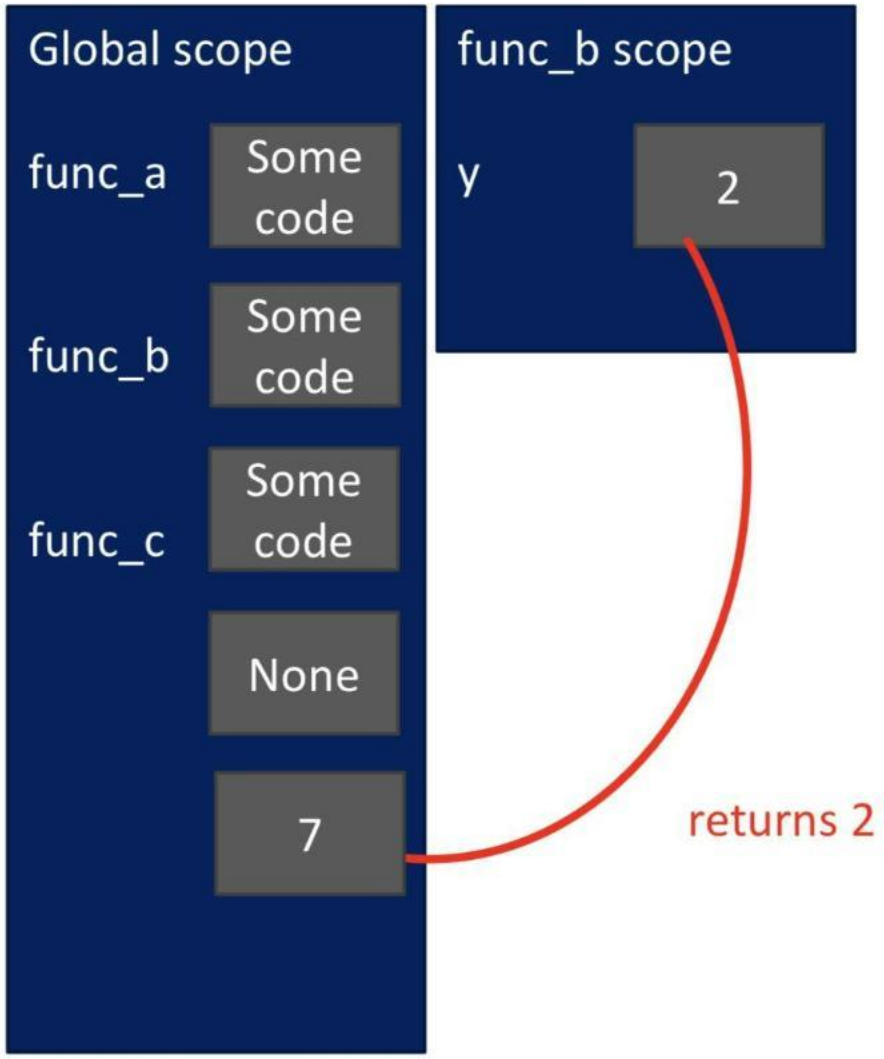
FUNCTIONS AS ARGUMENTS

```
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)
```



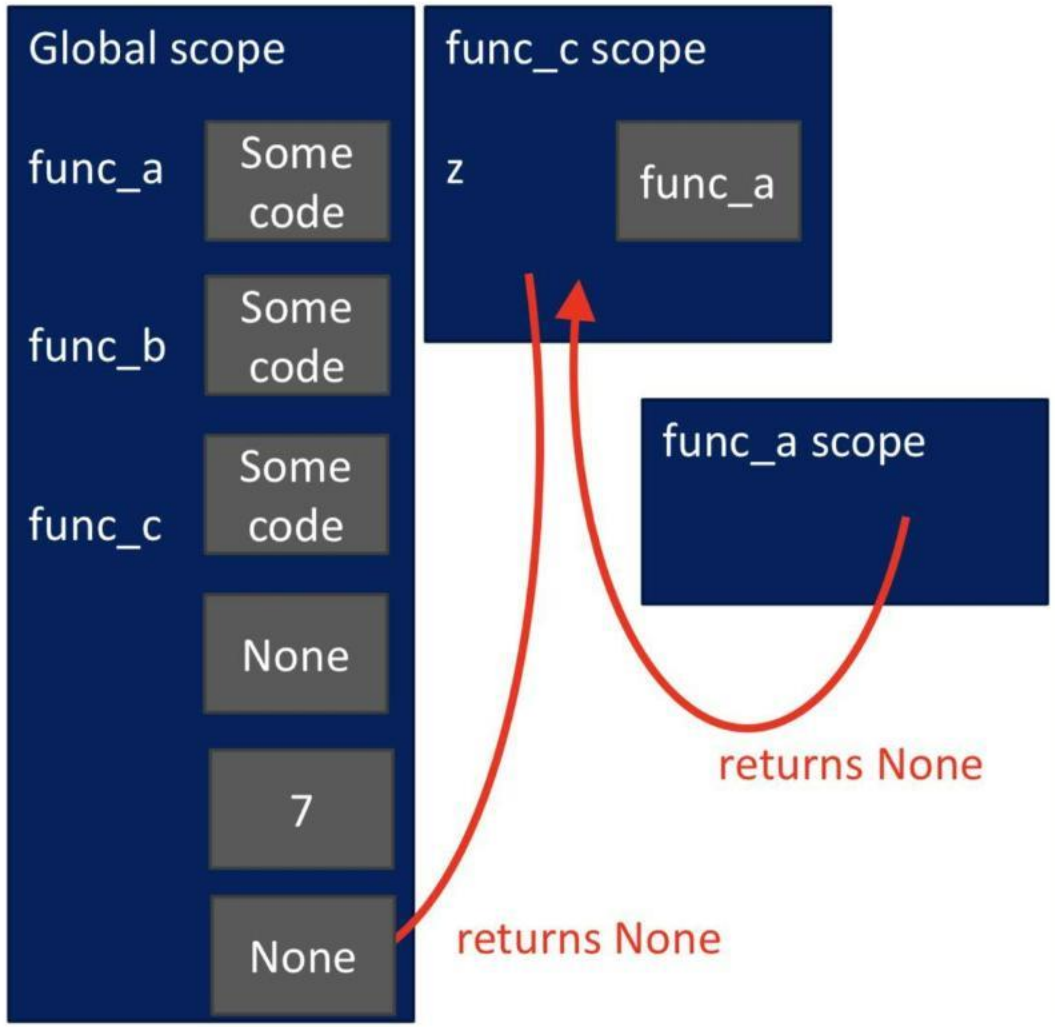
FUNCTIONS AS ARGUMENTS

```
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)
```



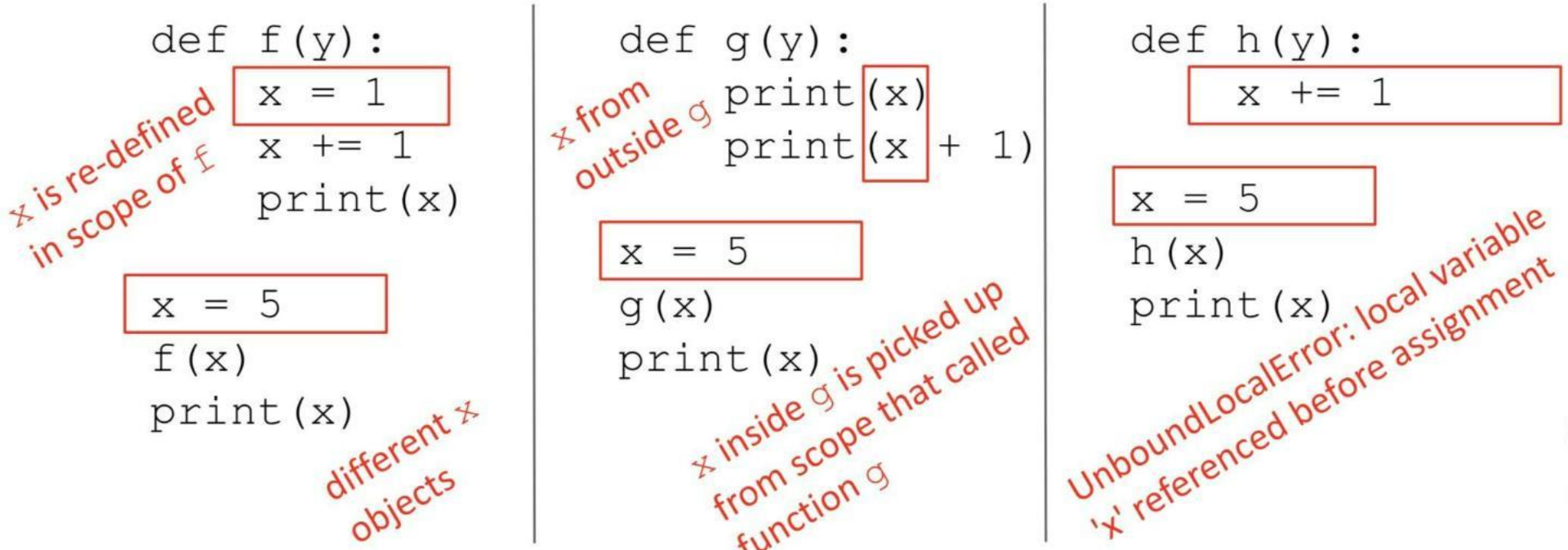
FUNCTIONS AS ARGUMENTS

```
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



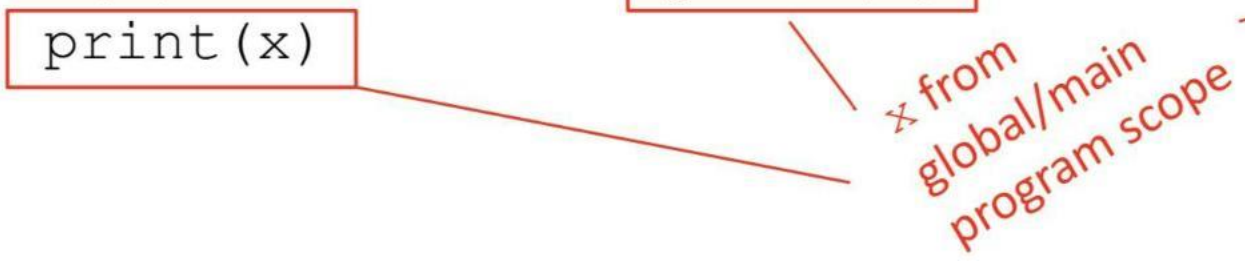
SCOPE EXAMPLE

- 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)
```

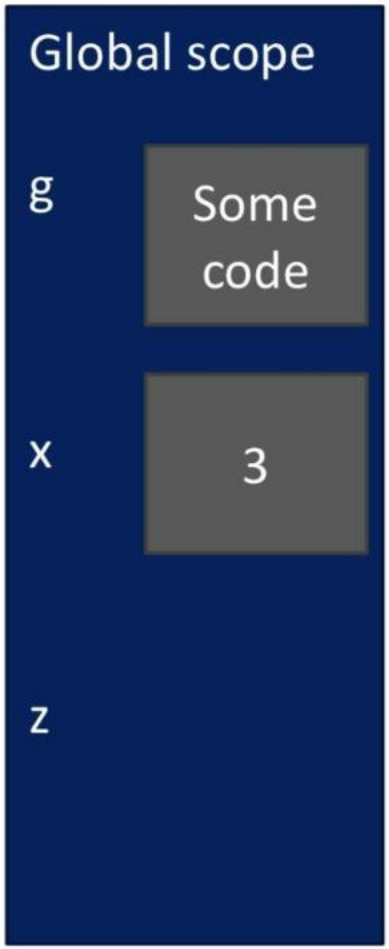
```
def h(y):
    x += 1


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



SCOPE DETAILS

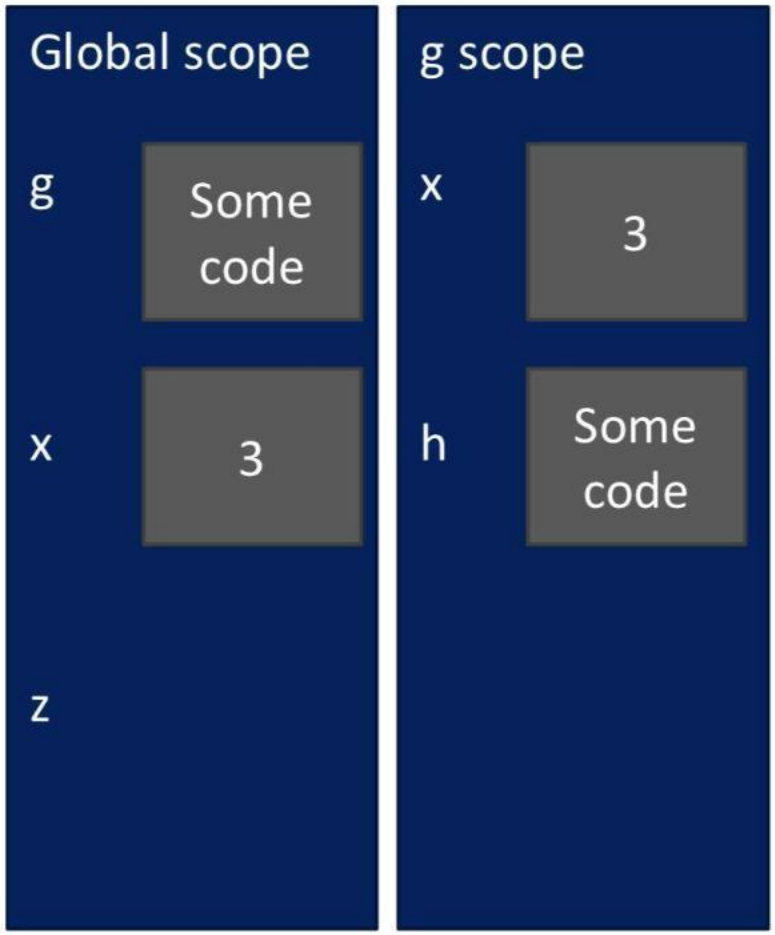
```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

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

Some code



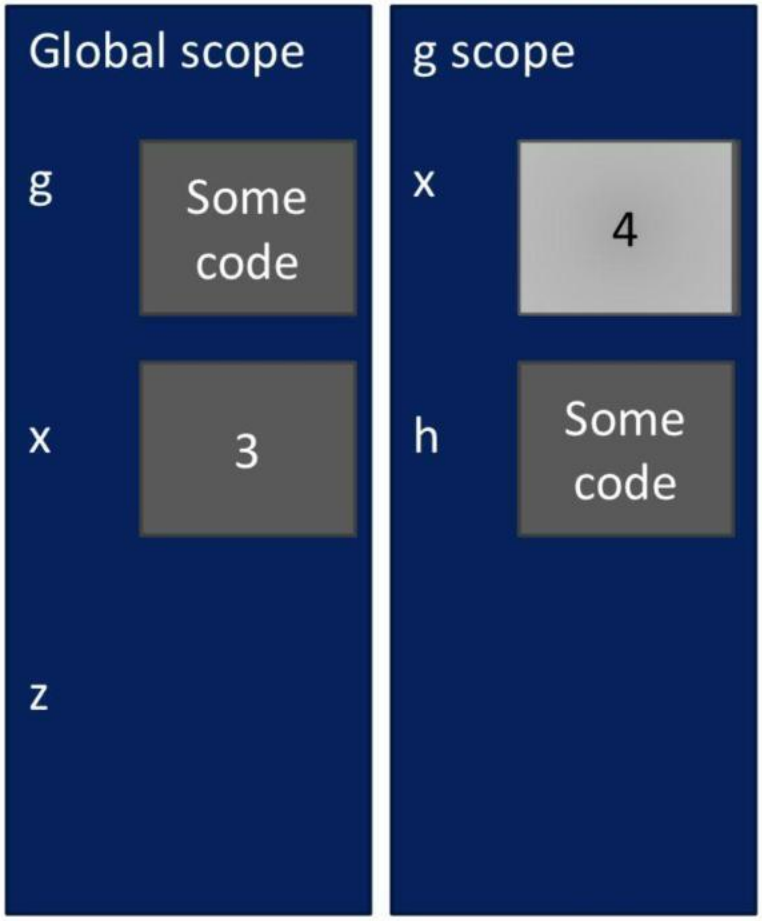
SCOPE DETAILS

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

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



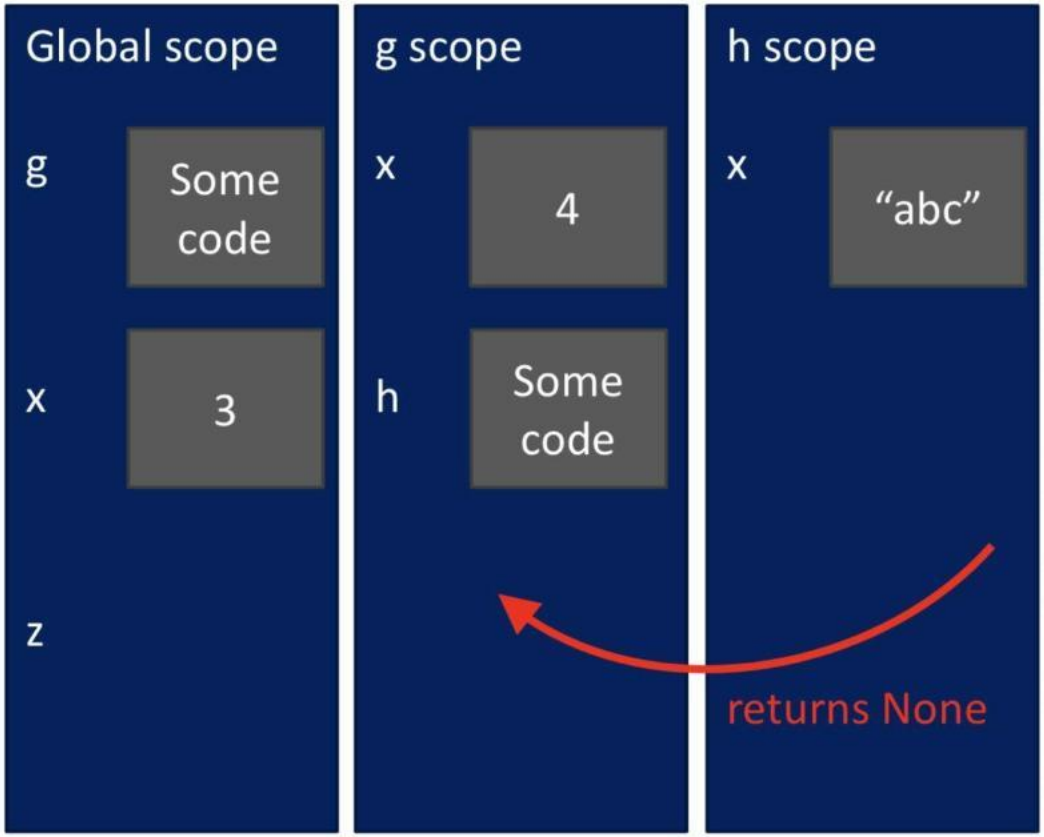
SCOPE DETAILS

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

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



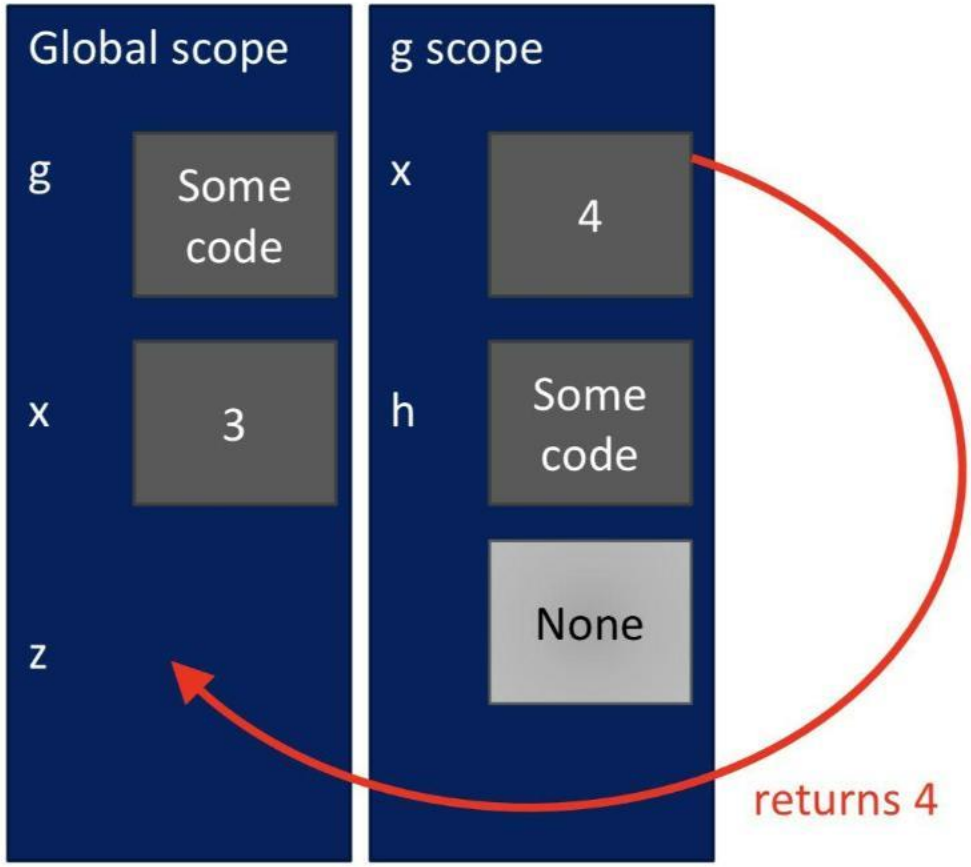
SCOPE DETAILS

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

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



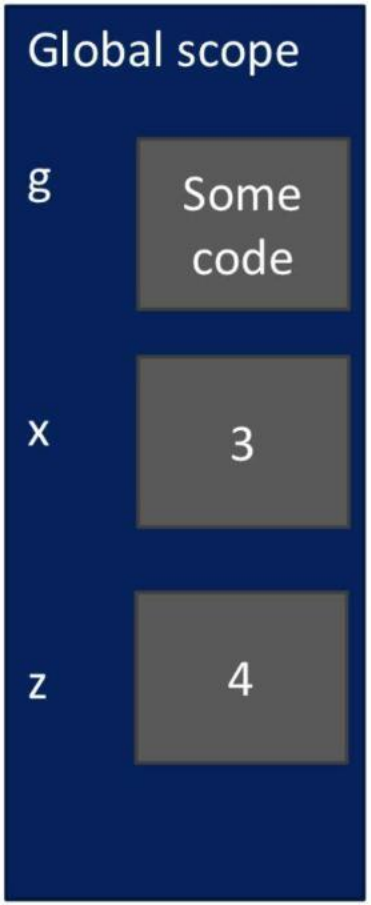
SCOPE DETAILS

```
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 хэрэглэх боловч нэг л удаа дебаг хийгдэнэ.
- *args болон **kwargs