សាកលវិទ្យាល័យបៀលប្រាយ
BUILD BRIGHT UNIVERSITY
Siem Reap Campus

មុខវិជ្ជា៖

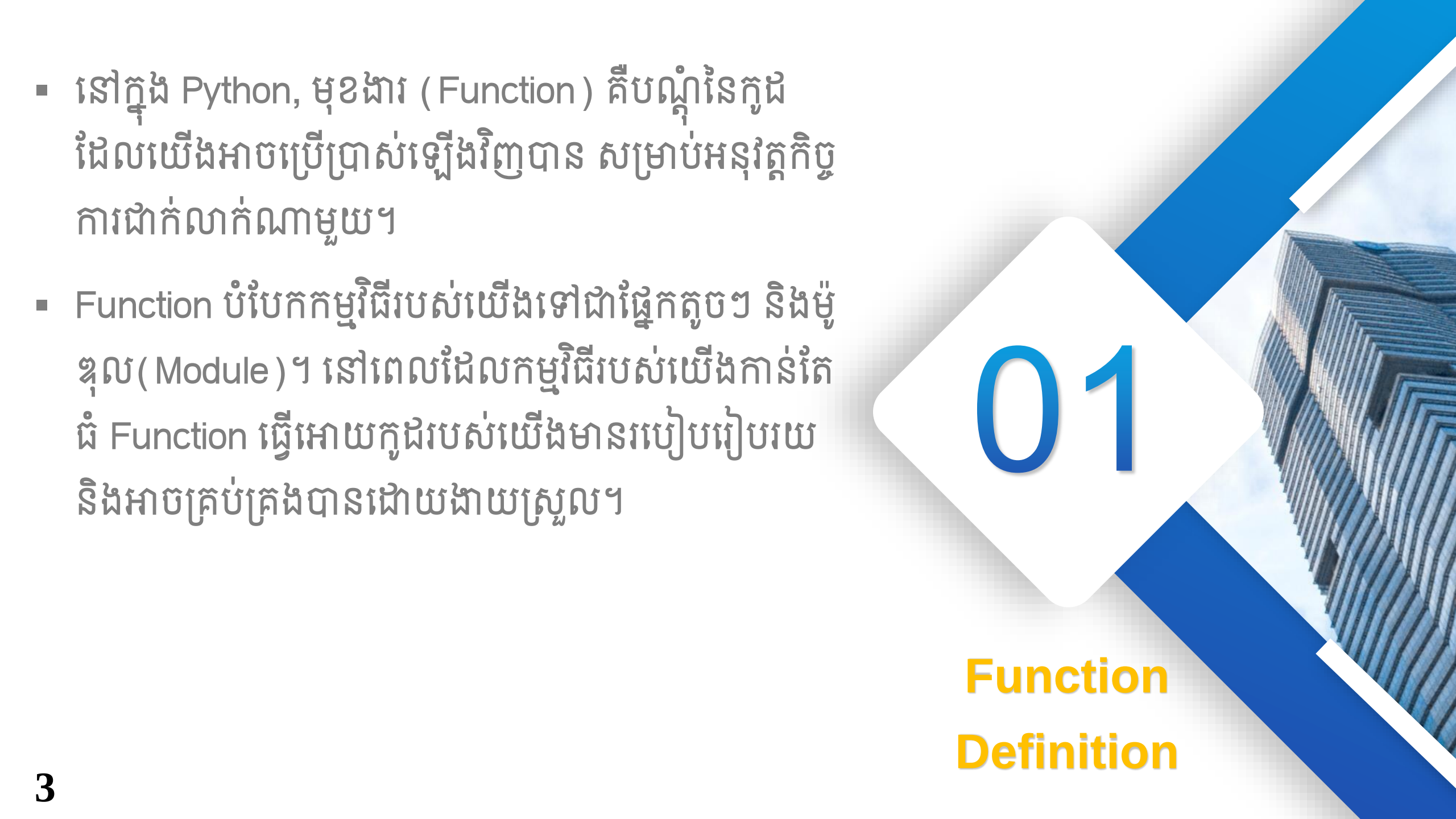
Python Programming

មេរៀនទី ៦៖

Functions

បង្រៀនដោយ៖ ខឿត សុផា

- នៅក្នុង Python, មុខងារ (Function) គឺបណ្តុំនៃកូដដែលយើងអាចប្រើប្រាស់ឡើងវិញបាន សម្រាប់អនុវត្តកិច្ចការជាក់លាក់ណាមួយ។
- Function បំបែកកម្មវិធីរបស់យើងទៅជាផ្នែកតូចៗ និងម៉ូឌុល(Module)។ នៅពេលដែលកម្មវិធីរបស់យើងកាន់តែធំ Function ធ្វើអោយកូដរបស់យើងមានរបៀបរៀបរយ និងអាចគ្រប់គ្រងបានដោយងាយស្រួល។



01

Function Definition

Function នៅក្នុង Python ចែកចេញជា ៤ ប្រភេទគឺ៖

1. Built-In Function
2. Predefined Function or User Defined Function
3. Lambda Function
4. Modules

02

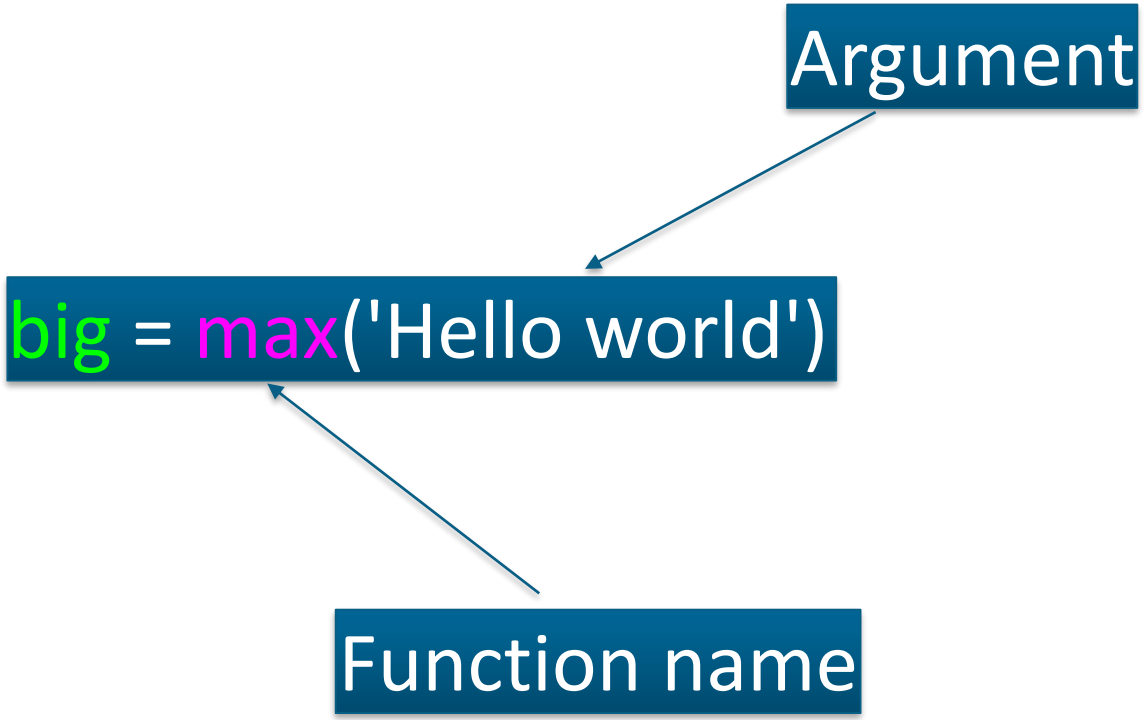
Built-In functions

Built-in functions គឺជាមុខងារ (Functions) ដែលមានស្រាប់នៅក្នុង Python ដែលត្រូវបានប្រើជាញឹកញាប់នៅពេលសរសេរកម្មវិធី។

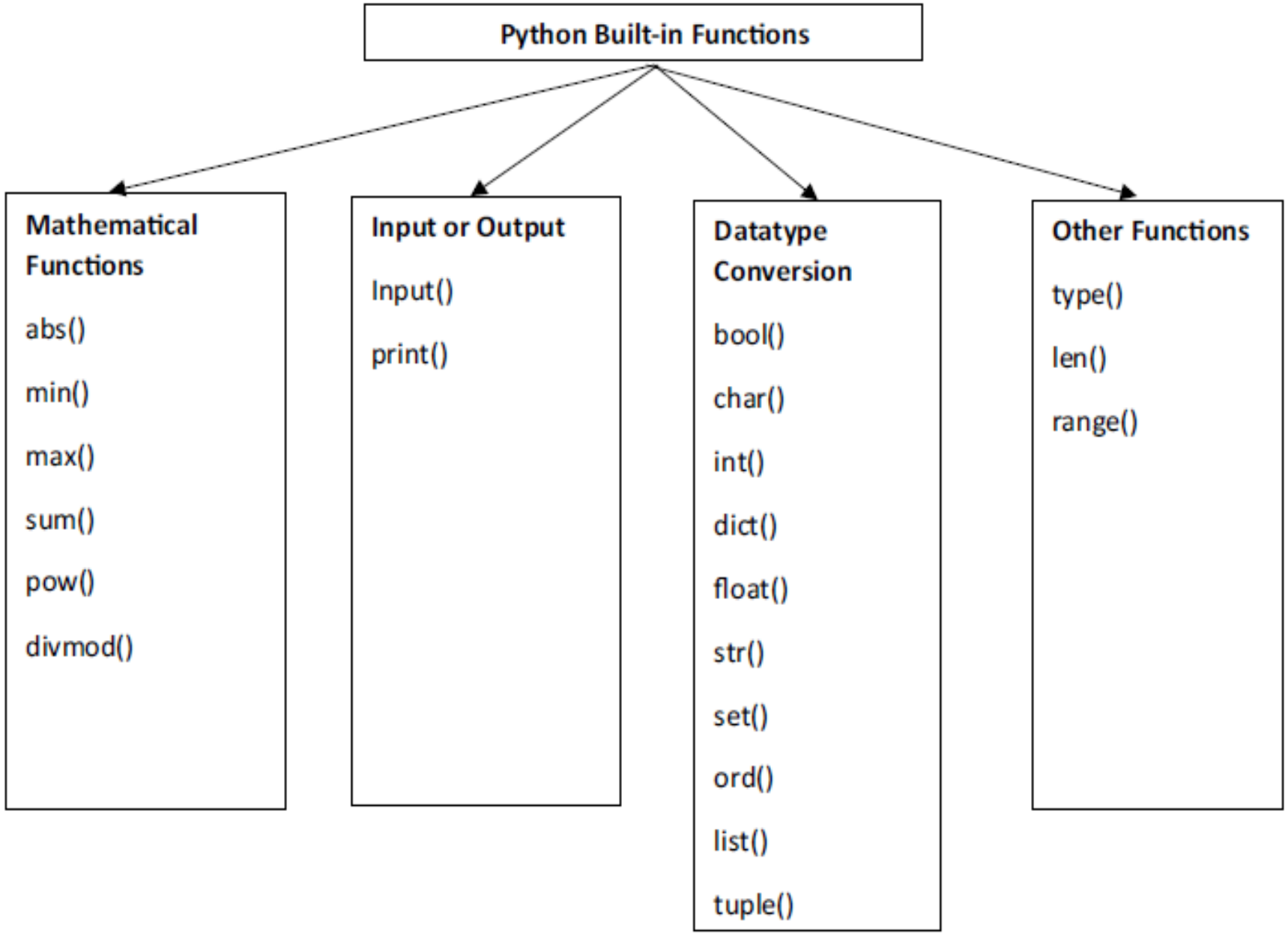
```
#Program to calculate square of a number

a = int(input("Enter a number: ")
b = a * a
print(" The square of ",a ,"is", b)
```

input( ), int( ) និង print( ) គឺជា built- in functions។

Max Function




```
>>> big = max('Hello world')
>>> print (big)
>>> w
>>> tiny = min('Hello world')
>>> print (tiny)
>>>
```


Python Built-in Functions
Mathematical Functions
abs()
min()
max()
sum()
pow()
divmod()
Input or Output
Input()
print()
Datatype Conversion
bool()
char()
int()
dict()
float()
str()
set()
ord()
list()
tuple()
Other Functions
type()
len()
range()

- នៅក្នុង Python ការកំណត់មុខងារ (Defining A Function) មានន័យថាបង្កើតប្លុកកូដដែលអាចប្រើឡើងវិញបាន ដែលអាចហៅបានដោយ ឈ្មោះជាក់លាក់មួយ ដើម្បីបំពេញកិច្ចការជាក់លាក់ណាមួយ។

Syntax

```
def function name([parameters]):
    Statement
    return[expression]
```

03

Defining A Function

Program

```
def test ():
   print (“this is inside the function”)
   print (“this is the second statement inside the function”)


test()
```

```
#Output
this is inside the function
this is the second statement inside the function
```

Program

```
def sum (i, j):
   print ("sum=", i+ j)
sum (10,20)
```

```
#Output
sum= 30
```

```
def sum (i, j, k):
   print ("sum=", i+ j+ k)
sum (10,20,30)
```

```
#Output
sum= 60
```

04

Function Arguments

- ប៉ារ៉ាម៉ែត្រ(Parameter)គឺជាអថេរដែលបានរាយក្នុងវង់ក្រចកក្នុងពេលកំណត់ឬបង្កើត Function ហើយវាជាកន្លែងសម្រាប់ដាក់តម្លៃដែលនឹងត្រូវបានបញ្ជូនទៅអនុគមន៍ នៅពេលវាត្រូវបានហៅ។
- អាគុយម៉ង់(Arguments) គឺជាតម្លៃពិតដែលត្រូវបានបញ្ជូនទៅអនុគមន៍នៅពេលវាត្រូវបានហៅ។
- **ចំណាំ៖** អ្នកសរសេរកម្មវិធីត្រូវតែផ្តល់អាគុយម៉ង់ ស្មើនឹងចំនួនប៉ារ៉ាម៉ែត្រដែលបានផ្តល់ឱ្យ បើមិនដូច្នោះទេកំហុសនឹងកើតឡើង។

For example

```
def test (i, j):
   print("i=", i, "j=",j)
def(10)
```

Output

```
File "<ipython-input-105-4ec7275e726b>", line 3
def(10)
  ^
SyntaxError: invalid syntax
```

អ្នកសរសេរកម្មវិធី អាចហៅមុខងារមួយដោយអនុវត្តប្រភេទនៃអាគុយម៉ង់ផ្លូវការដូចខាងក្រោម:

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable arguments

4.1 Required Arguments

អាគុយម៉ង់ដែលត្រូវការ គឺជាចំនួននៃអាគុយម៉ង់ដែលត្រូវការនៅពេលហៅមុខងារ(Function)មកប្រើ អោយត្រូវតាមចំនួននៃប៉ារ៉ាម៉ែត្រ នៅពេលបង្កើត Function។

```
def test(i,j):
    print("i=",i,"j=",j)

test(10,20)
test(20,10)
```

```
#Output
i= 10 j= 20
i= 20 j= 10
```

4.2 Keyword arguments

- ✓ Keyword argument ត្រូវបានភ្ជាប់ជាមួយនឹងការហៅមុខងារ(Function)។
- ✓ នៅពេលអ្នកសរសេរកម្មវិធីប្រើ Keyword argument ក្នុងការហៅមុខងារ មុខងារទទួលស្គាល់អាគុយម៉ង់ដោយឈ្មោះប៉ារ៉ាម៉ែត្រ។
- ✓ អ្នកសរសេរកម្មវិធីអាចរំលងឬដាក់វាខុសពីលំដាប់បាន ដោយសារអ្នកបកប្រែ python អាចប្រើប្រាស់ពាក្យគន្លឹះដែលត្រូវការដើម្បីផ្គូផ្គងតម្លៃជាមួយនឹងប៉ារ៉ាម៉ែត្រ។ ទីតាំងមិនសំខាន់ទេ ព្រោះតែតម្លៃអាគុយម៉ង់នីមួយៗដឹងពីទិសដៅរបស់វាដោយផ្អែកលើឈ្មោះ (ពាក្យគន្លឹះ) ដែលបានប្រើ។

4.2 Keyword arguments

```
def test(name,surname):
    print("name:",name,"surname:",surname)

test(name="Dara",surname="Sun")
test(surname="Sun",name="Dara")
```

```
#Output
name: Dara surname: Sun
name: Dara surname: Sun
```

4.3 Default Arguments

- ✓ អាគុយម៉ង់លំនាំដើម(Default Arguments) គឺជាអាគុយម៉ង់ដែលសន្មតថា នឹងយកតម្លៃលំនាំដើមមកប្រើ ប្រសិនបើតម្លៃមិនត្រូវបានផ្តល់ឱ្យនៅក្នុងការហៅមុខងារ (Function) សម្រាប់អាគុយម៉ង់នោះ។

```
def add(a=1,b=5,c=8):
    print("add=",(a+b+c))

add(10)
```

```
#Output
add= 23
```

4.3 Default Arguments

```
def add(a=1,b=5,c=8):
    print("add=",(a+b+c))

add(10,20,30)
```

```
#Output
add= 60
```

4.3 Default Arguments

```
def add(a,b=5,c):
    print("add=",(a+b+c))

add(a=10,c=20)
```

#Output

```
File "<ipython-input-118-957033311ea7>", line 1
def add(a,b=5,c):
              ^
SyntaxError: non-default argument follows default
  argument
```

✓ ចំណាំ៖ default arguments ត្រូវតែនៅបន្ទាប់ non-default arguments។

4.4 Variable Arguments

- ✓ Variable Arguments គឺអនុញ្ញាតឱ្យ function ទទួលយកចំនួននៃ arguments ណាមួយ។
- ✓ Variable Arguments គឺកំណត់ដោយប្រើ (*) នៅពេលប្រកាស function។

```
def sum_all(*args):
    return sum(args)

print(sum_all(1, 2, 3, 4))   # Output: 10
print(sum_all(10, 20))       # Output: 30
```

4.4 Variable Arguments

```
def myFun(*argv):
    for arg in argv:
        print(arg)

myFun('Hello', 'Welcome', 'to', 'BBU')
```

```
#Output
Hello
Welcome
to
BBU
```

4.4 Variable Arguments

```
def myFun(arg1, *argv):
    print("First argument :", arg1)
    for arg in argv:
        print("Next argument through *argv :", arg)

myFun('Hello', 'Welcome', 'to', 'BBU')
```

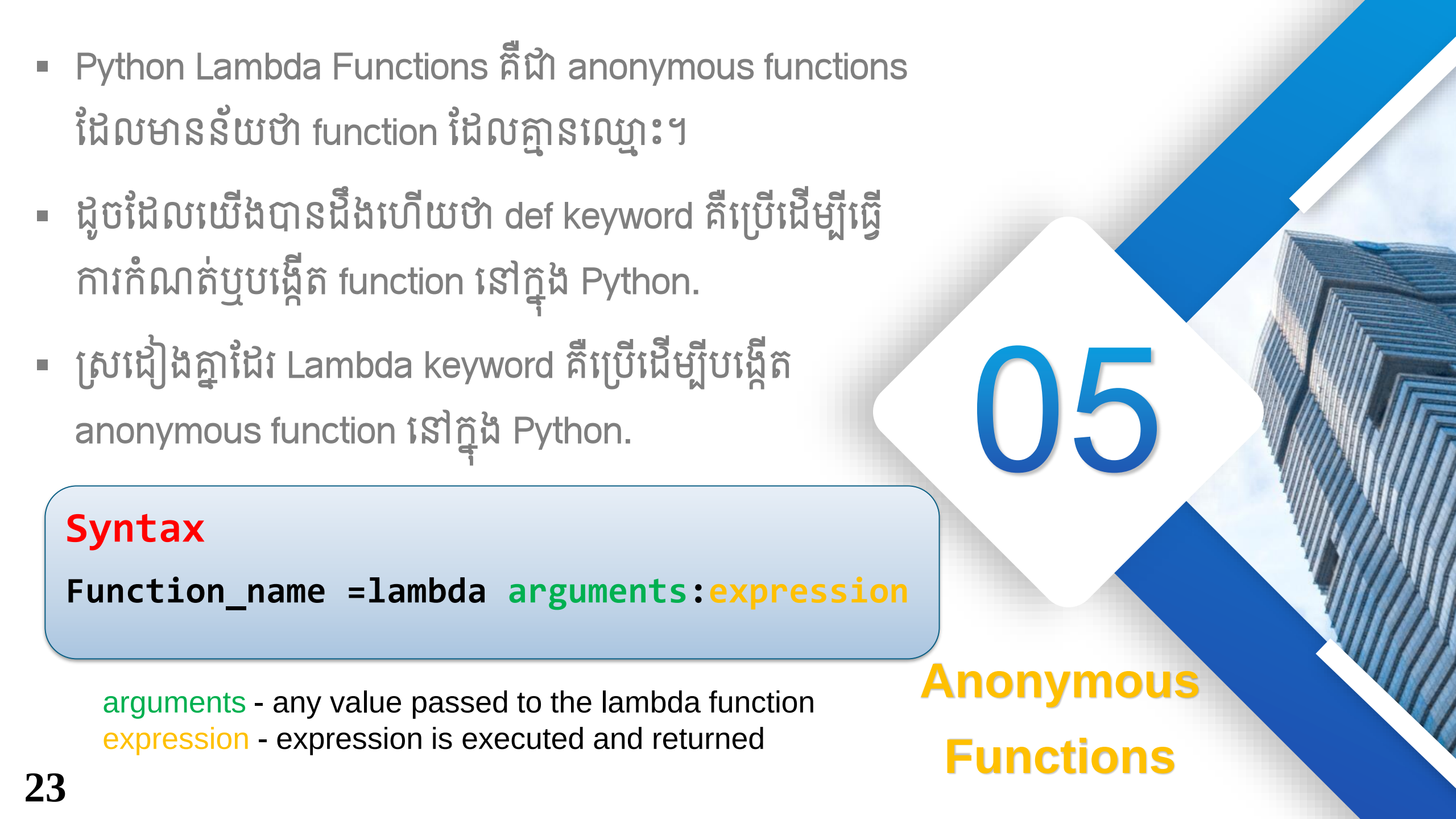
```
#Output
First argument : Hello
Next argument through *argv : Welcome
Next argument through *argv : to
Next argument through *argv : BBU
```

- Python Lambda Functions គឺជា anonymous functions ដែលមាននិយមថា function ដែលគ្មានឈ្មោះ។
- ដូចដែលយើងបានដឹងហើយថា def keyword គឺប្រើដើម្បីធ្វើការកំណត់ឬបង្កើត function នៅក្នុង Python.
- ស្រដៀងគ្នាដែរ Lambda keyword គឺប្រើដើម្បីបង្កើត anonymous function នៅក្នុង Python.

Syntax

```
Function_name =lambda arguments:expression
```

arguments - any value passed to the lambda function
expression - expression is executed and returned



05

Anonymous Functions

Program

```
greet = lambda : print('Hello World’)

# call the lambda
greet()
```

```
#Output
Hello World
```

```
# lambda that accepts one argument
greet_user = lambda name : print(‘Hey ', name)

# lambda call
greet_user(‘Student')
```

```
#Output
Hey Student
```

Program

```
#lambda statement with one parameter.

j = lambda i : i + 5

print(j(3))
```

```
#Output

8
```

```
#lambda statement with two parameter.

m = lambda i,j : i+j

print(m(3,4))
```

```
#Output

7
```

Program

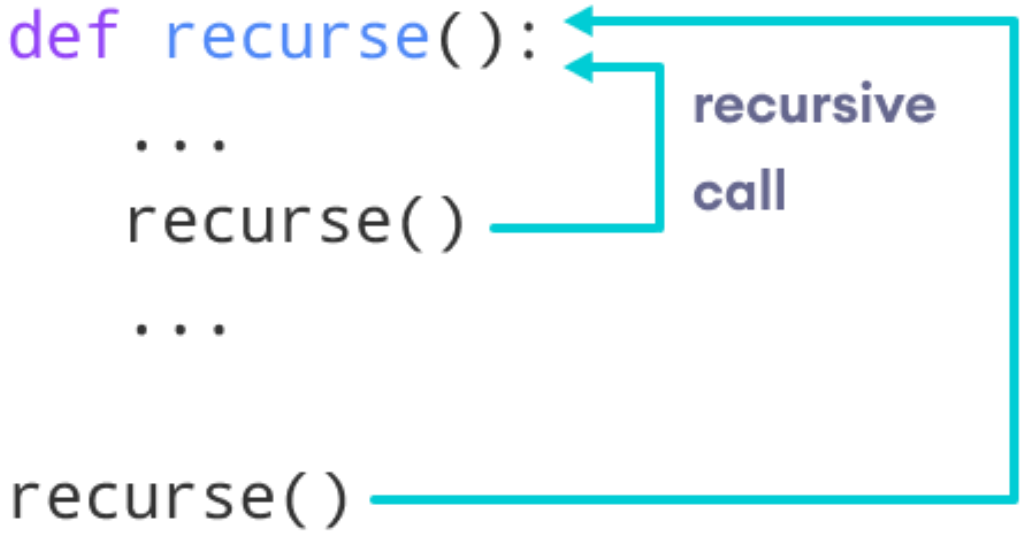
```
#lambda and the filter()

s = [1,2,3,4,5,6,7]

m = list(filter(lambda i: i%2==0, s))

print(m)
```

```
#Output
[2, 4, 6]
```

06

Python Recursion

- នៅក្នុង Python យើងធ្លាប់ដឹងហើយថា Function មួយអាចហៅ Functions មួយផ្សេងទៀត។ ផងដែរនោះ Function ក៏អាចហៅខ្លួនឯងបានដែរ ដែល Function ប្រភេទនេះហៅថា Recursive Functions។



Program

```
def rec(n):
   if n == 1:
      return n
   else:
      return n*rec(n-1)

n = 5
if n < 0:
   print("no factorial for negative numbers")
elif n == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of", n, "is", rec(n))
```

#Output

```
The factorial of 5 is 120
```

Program

```
# Python program to display the Fibonacci sequence
def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms = 10

# check if the number of terms is valid
if nterms <= 0:
    print("Plese enter a positive integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))
```

```
#Output
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
34
```

THANKS