Machine Learning Diploma

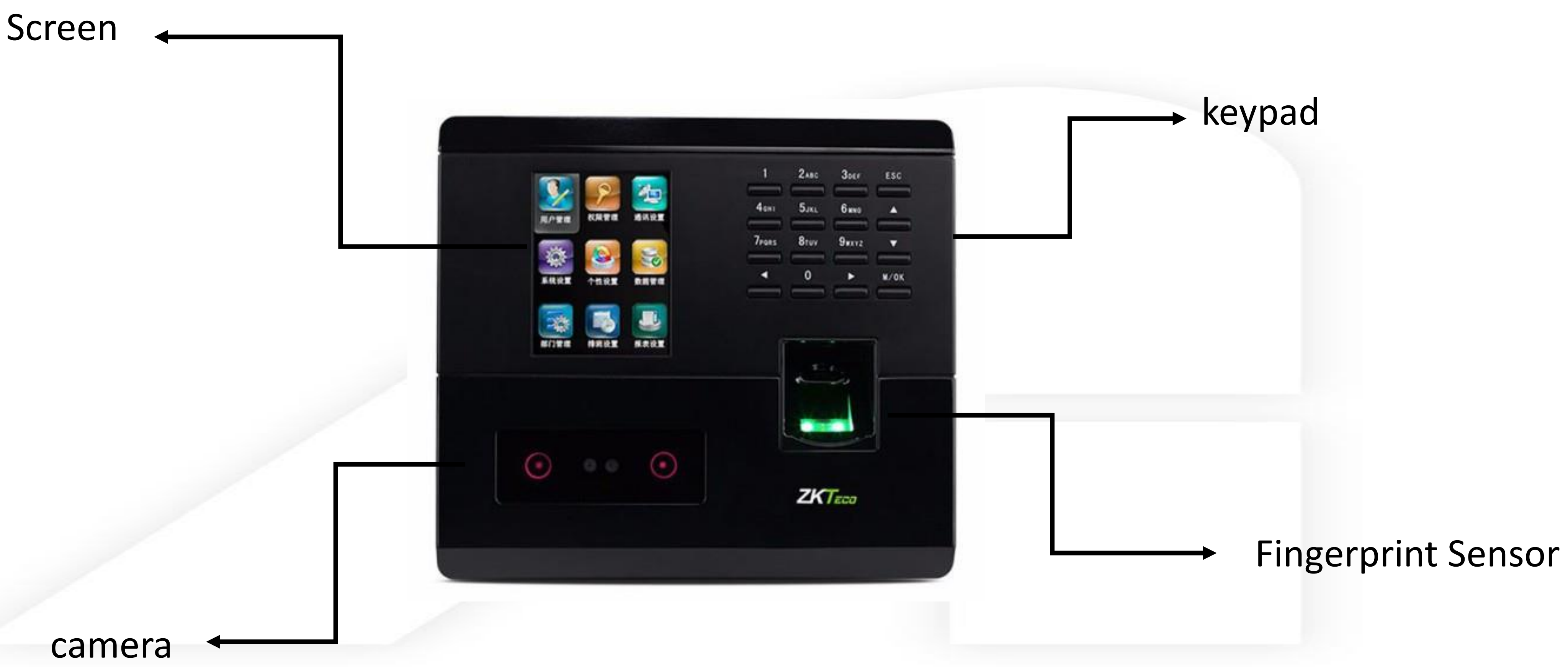
Session 3: Functions

Agenda:

1	introduction to function in Python Programming
2	Types of Functions
3	Built-In functions
4	User Defined Function
5	User Defined Function (def)
6	User Defined Function (lambda)

1- Introduction to Functions in Python Programming.

AMIT

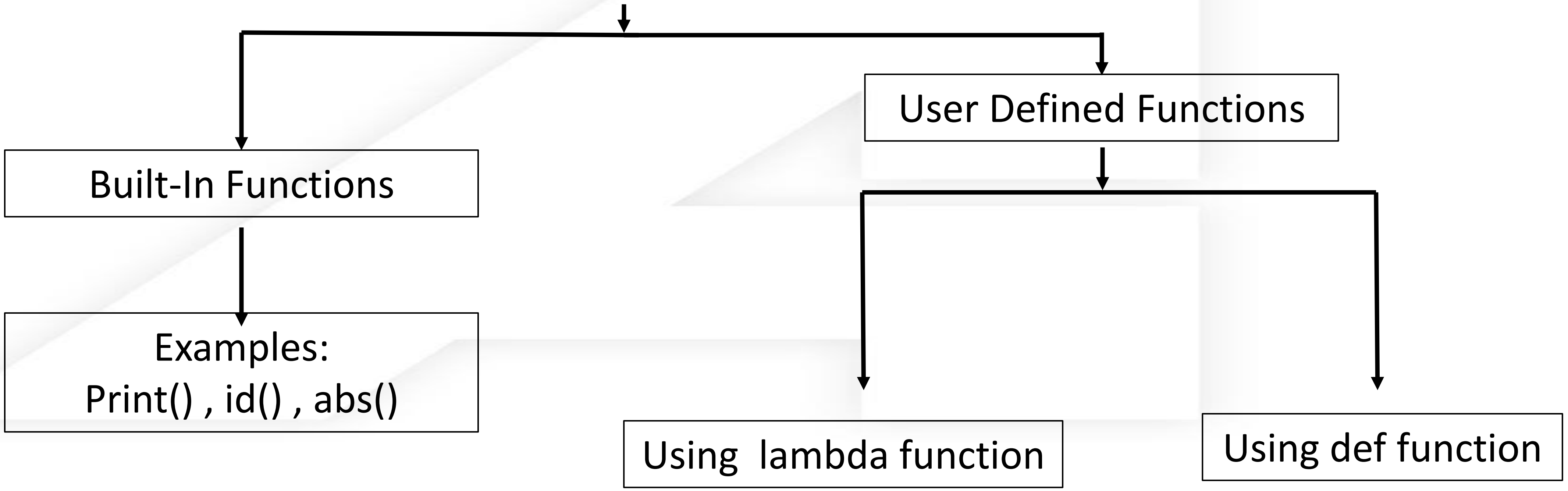

Screen
keypad
Fingerprint Sensor
camera
1
2ABC
3DEF
ESC
4GHI
5JKL
6MNO
7PQRS
8TUV
9WXYZ
0
M/OK
ZKTeco

2- Types of Functions

What is function

Functions in Python are blocks of reusable code that perform specific tasks.

Main Types Of functions

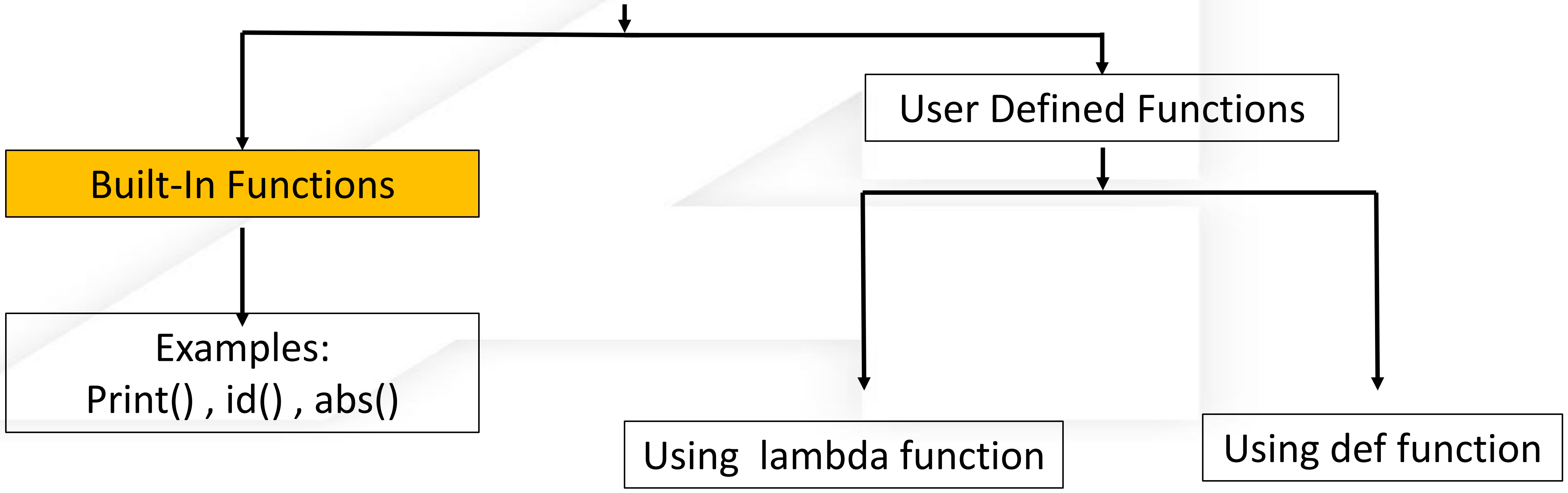


3- Built-In functions

What is function

Functions in Python are blocks of reusable code that perform specific tasks.

Main Types Of functions



Built-In Functions

A built-in function in Python is a predefined and integral function inherent to the language. These functions, covering various tasks like math, string manipulation, and list handling, are readily available for use without explicit declaration or external libraries, enhancing the language's versatility and efficiency.

Numeric Functions

abs():
Returns the absolute value of a number.

```
abs(-5)
```

```
5
```

round():
Rounds a number to the nearest integer.

```
round(3.14)
```

```
3
```

max() – min():
Returns the maximum or minimum of a sequence.

```
max(76,45,65)
```

```
76
```

String Functions

len():
Returns the length of a string.

```
len("Python")
```

```
6
```

str():
Converts a value to a string

```
str(4)
```

```
'4'
```

upper() – lower():
Converts a string to uppercase or lowercase.

```
"hello".upper()
```

```
'HELLO'
```

Input/Output Functions

Input():
Reads a line from the user.

```
x = input("first_number = ")
print(x)
```

```
first_number = 5
5
```

print():
Prints a message to the console.

```
print(5)
```

```
5
```

Format():
Formats a string.

```
"My name is {}.".format("amit")
```

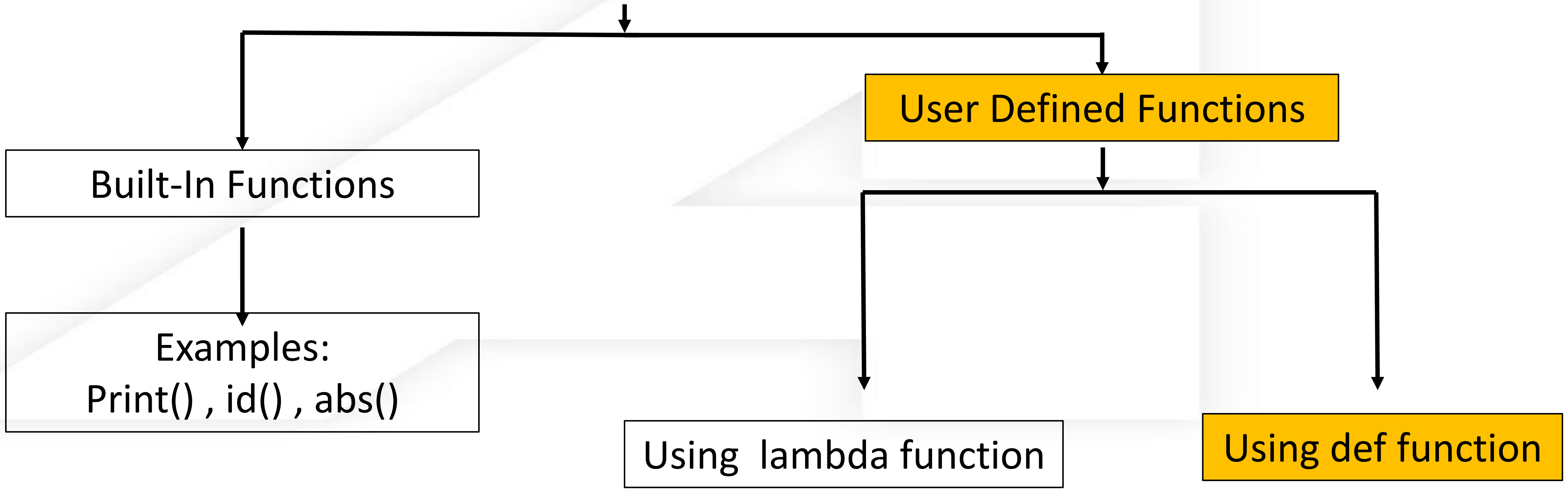
```
'My name is amit.'
```

4- User Defined Function

What is function

Functions in Python are blocks of reusable code that perform specific tasks.

Main Types Of functions

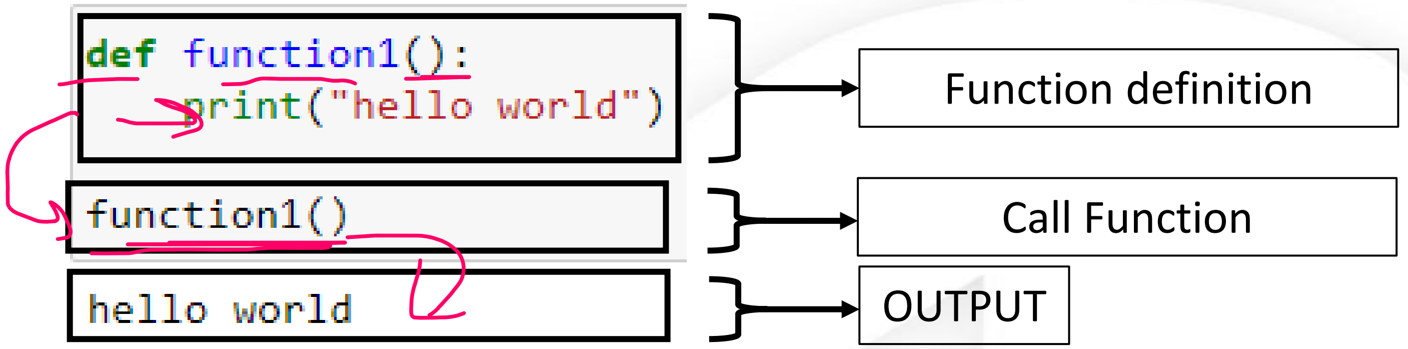


4- User Defined Function (def)

User Defined Function (def)

functions that you use to organize your code in the body of a policy. Once you define a function, you can call it in the same way as the built-in action and parser functions.

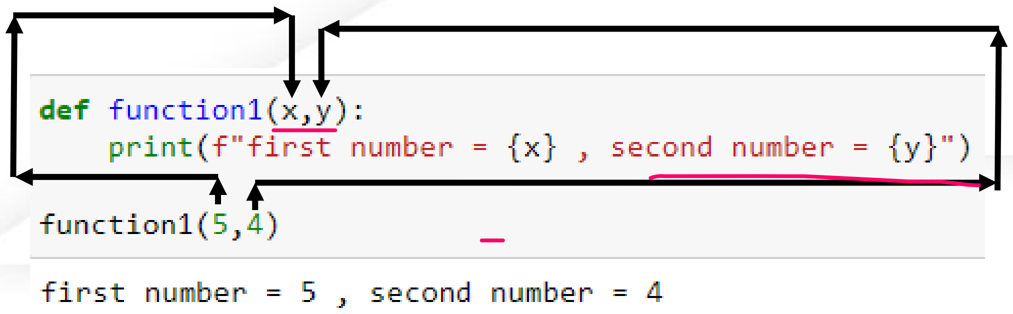
Using def Keyword:



```
def function1():
    print("hello world")
```

```
function1()
```

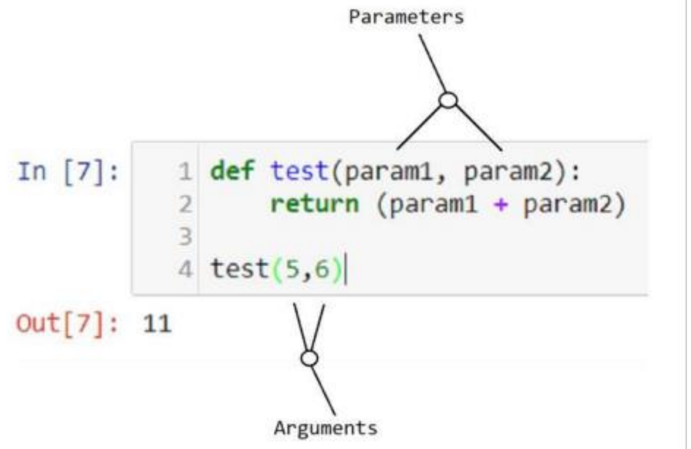
```
hello world
```



```
def function1(x,y):
    print(f"first number = {x} , second number = {y}")
```

```
function1(5,4)
```

```
first number = 5 , second number = 4
```



```
In [7]: def test(param1, param2):
            return (param1 + param2)

        test(5,6)
Out[7]: 11
```

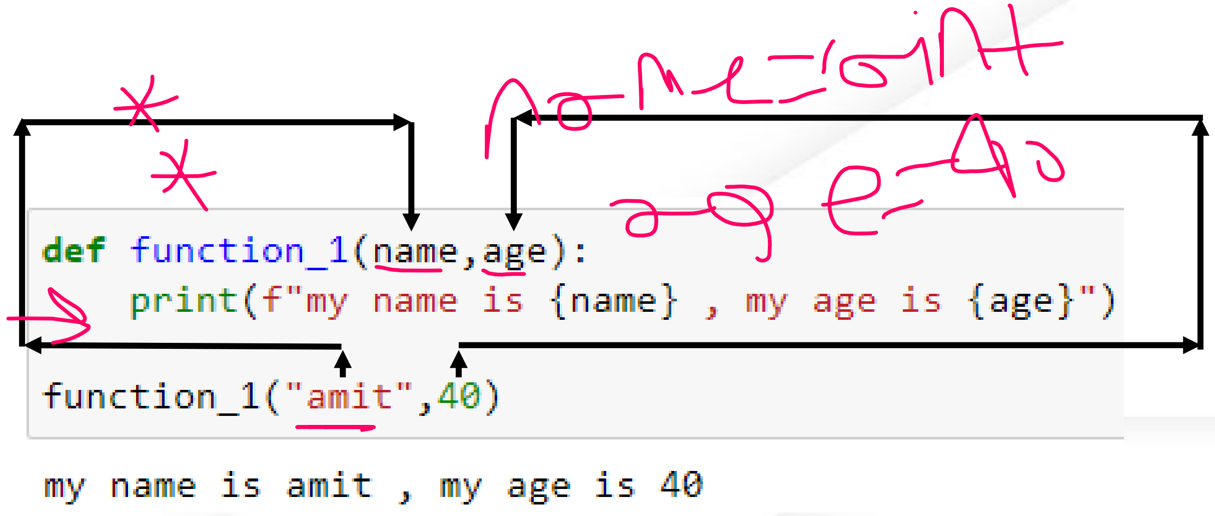
Types of Functions

Positional Argument Function	Keyword Argument Function	Default Argument Function
arguments are matched based on their position in the function call. The order and number of arguments matter.	Arguments are passed to the function using the parameter names, allowing you to specify which argument corresponds to which parameter.	Function parameters can have default values, which are used if the corresponding argument is not provided in the function call.

*Varadic Argument Function	**Kwargs Argument Function
The function can accept a variable number of arguments and store them as a tuple.	The function can accept a variable number of arguments and store them as a dictionary.

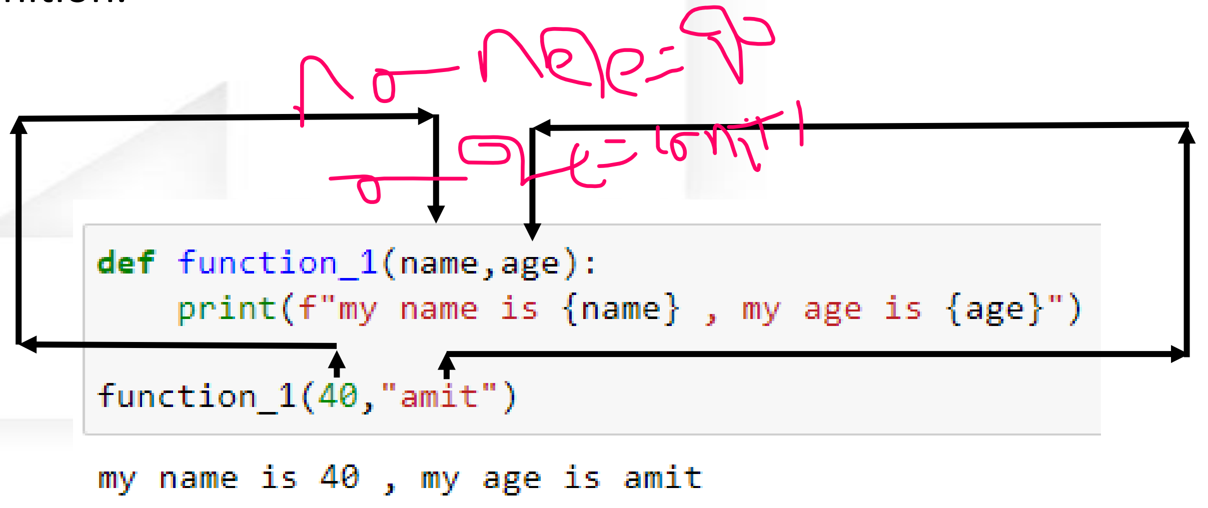
Positional Argument Function

During a function call, values passed through arguments should be in the order of parameters in the function definition.



```
def function_1(name,age):
    print(f"my name is {name} , my age is {age}")

function_1("amit",40)
```

```
my name is amit , my age is 40
```



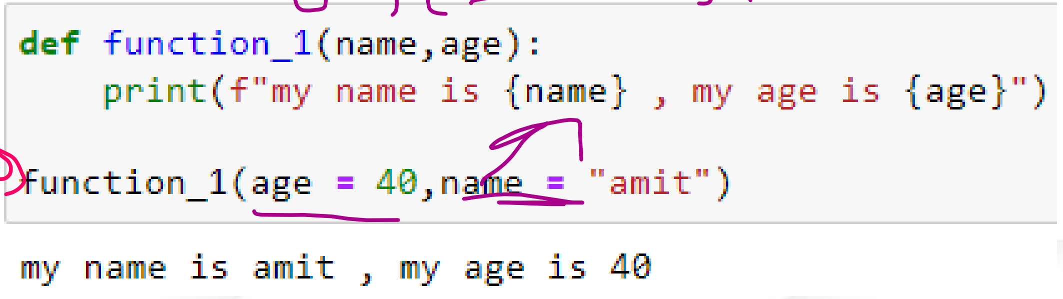
```
def function_1(name,age):
    print(f"my name is {name} , my age is {age}")

function_1(40,"amit")
```

```
my name is 40 , my age is amit
```

Keyword Argument Function

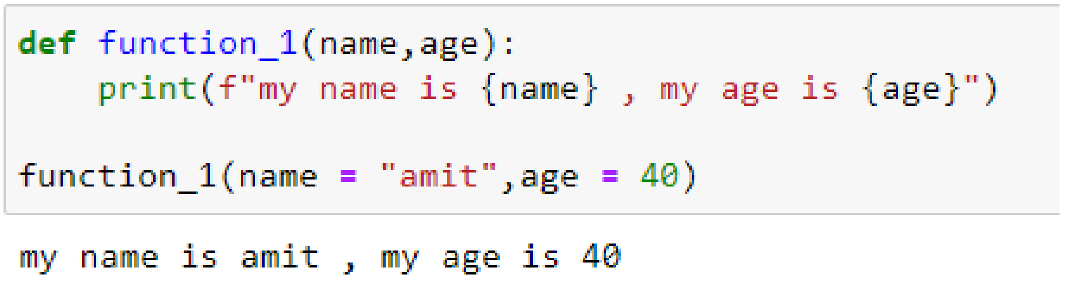
During a function call, values passed through arguments don't need to be in the order of parameters in the function definition.

age= 40 ,name=amit)



```
def function_1(name,age):
    print(f"my name is {name} , my age is {age}")

function_1(age = 40,name = "amit")
```

```
my name is amit , my age is 40
```



```
def function_1(name,age):
    print(f"my name is {name} , my age is {age}")

function_1(name = "amit",age = 40)
```

```
my name is amit , my age is 40
```

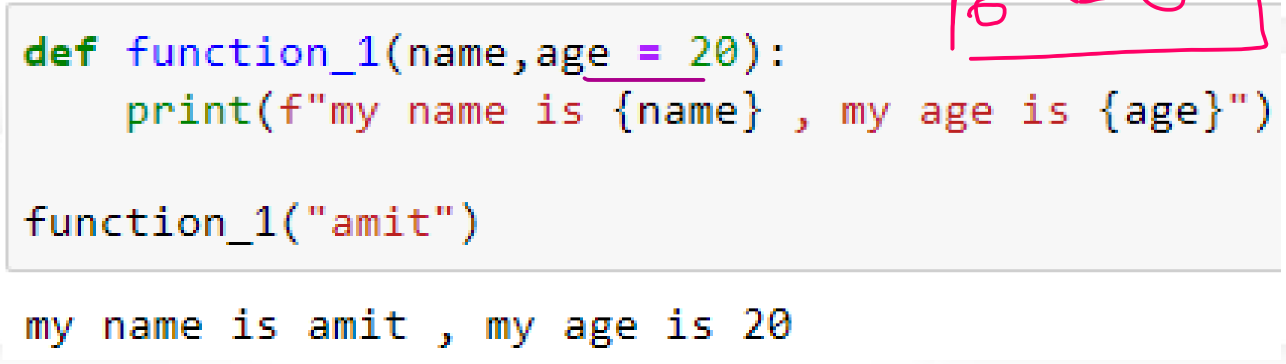
Default Argument Function

default parameter is defined with a fallback value as a default argument. Such parameters are optional during a function call. If no argument is provided, the default value is used, and if an argument is provided, it will overwrite the default value.

a = 5

$\boxed{a = 6}$

Print(a) = 6



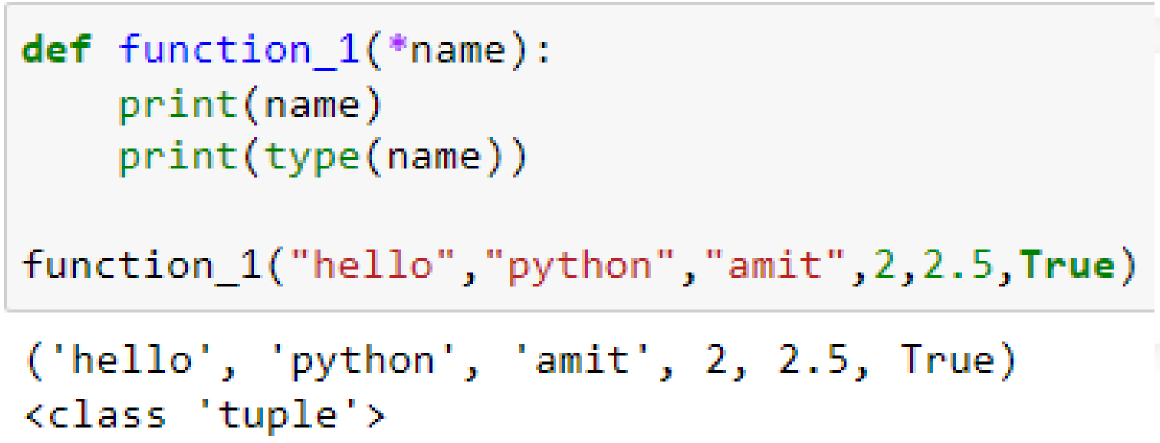
```
def function_1(name,age = 20):
    print(f"my name is {name} , my age is {age}")

function_1("amit")
```

```
my name is amit , my age is 20
```

Varadic Argument Function

When a function has a parameter preceded by an asterisk (*), it can accept a variable number of arguments. And you can pass zero, one, or more arguments to the *args parameter.

In Python, the parameters like *args are called variadic parameters and **store them as a tuple.**

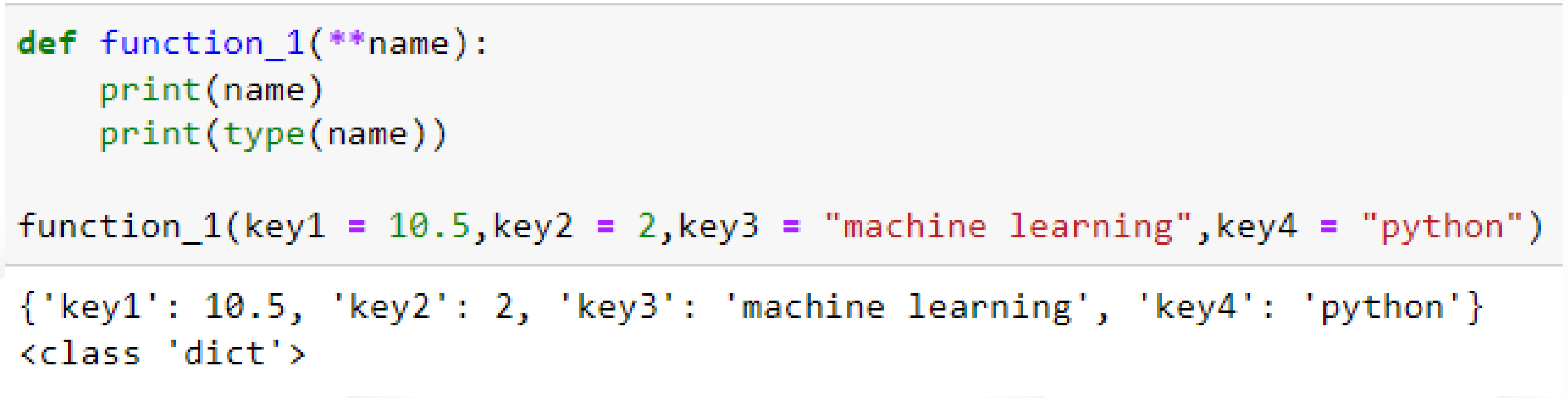


```
def function_1(*name):
    print(name)
    print(type(name))

function_1("hello","python","amit",2,2.5,True)
```

```
('hello', 'python', 'amit', 2, 2.5, True)
<class 'tuple'>
```

Kwargs Argument Function

special syntax that allows you to pass a keyworded, variable-length argument dictionary to a function. The function can accept a number of arguments and **store them as a dictionary.**



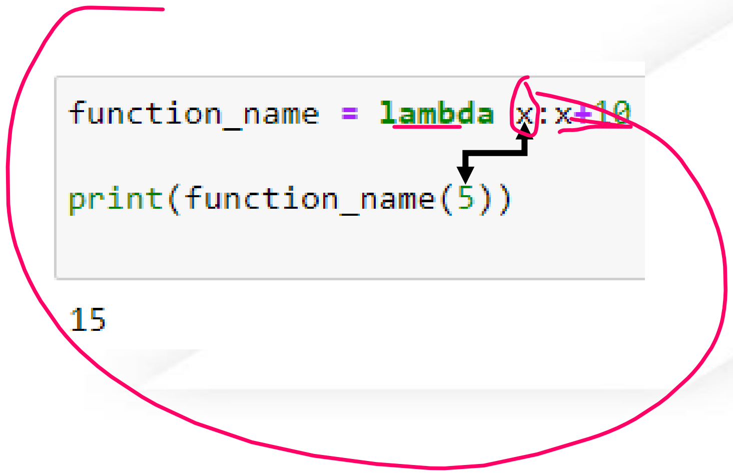
```
def function_1(**name):
    print(name)
    print(type(name))

function_1(key1 = 10.5,key2 = 2,key3 = "machine learning",key4 = "python")
```

```
{'key1': 10.5, 'key2': 2, 'key3': 'machine learning', 'key4': 'python'}
<class 'dict'>
```

6- User Defined Function (lambda)

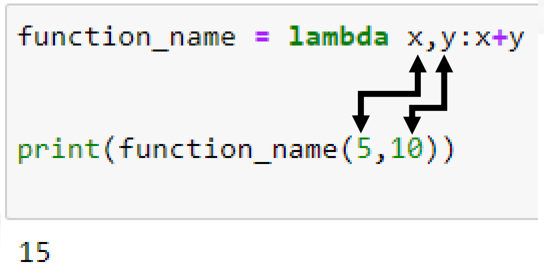
Lambda function

A lambda function is a small anonymous function. It can take any number of arguments, but can only have one expression



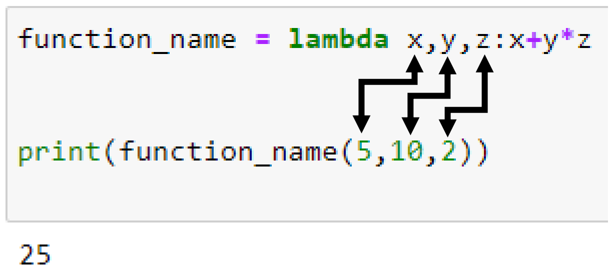
```
function_name = lambda x:x+10

print(function_name(5))
```

```
15
```



```
function_name = lambda x,y:x+y

print(function_name(5,10))
```

```
15
```



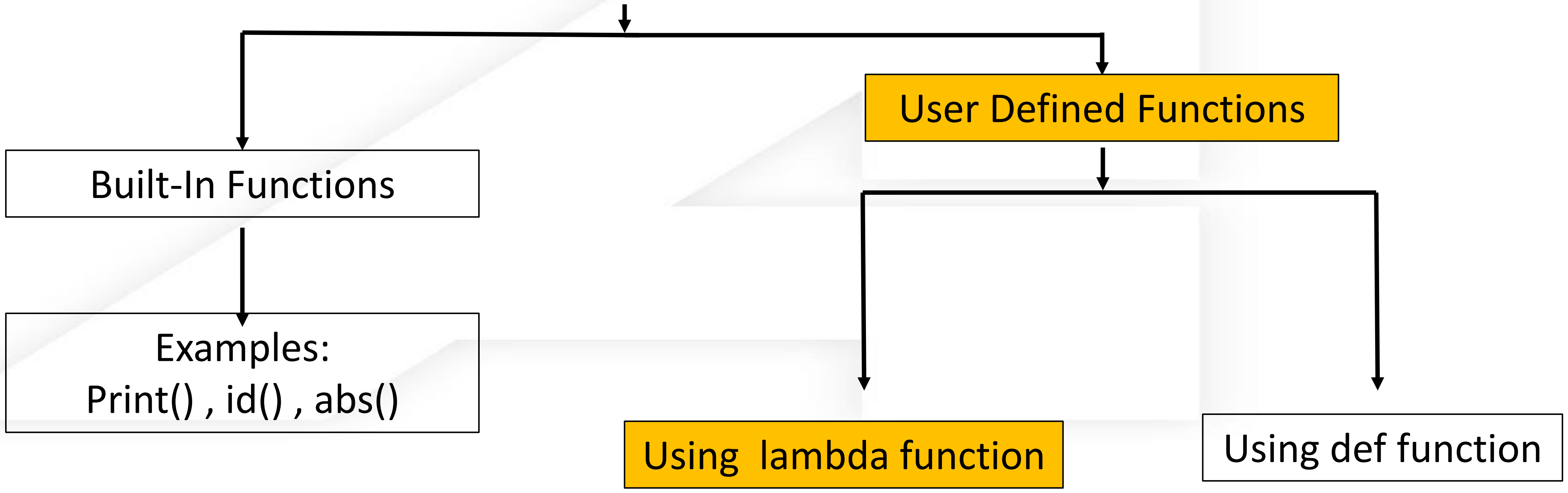
```
function_name = lambda x,y,z:x+y*z

print(function_name(5,10,2))
```

```
25
```

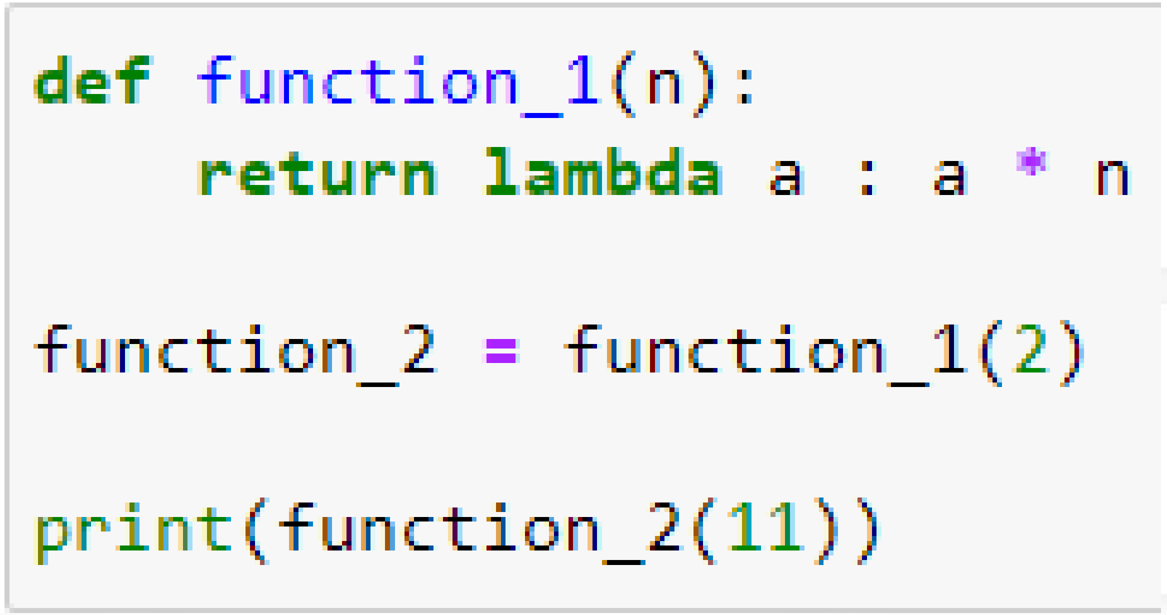
What is function

Functions in Python are blocks of reusable code that perform specific tasks.

Main Types Of functions

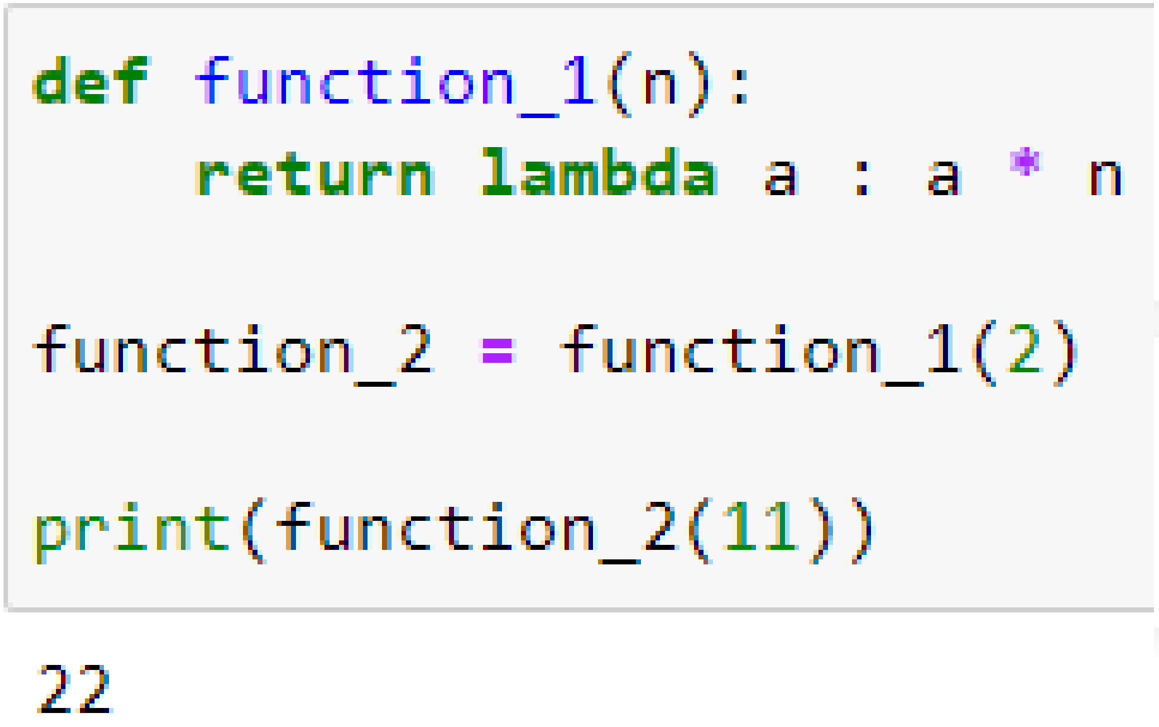


Try To Solve



```
def function_1(n):
    return lambda a : a * n

function_2 = function_1(2)

print(function_2(11))
```

Try To Solve



```
def function_1(n):
    return lambda a : a * n

function_2 = function_1(2)

print(function_2(11))
```

```
22
```

Thank You