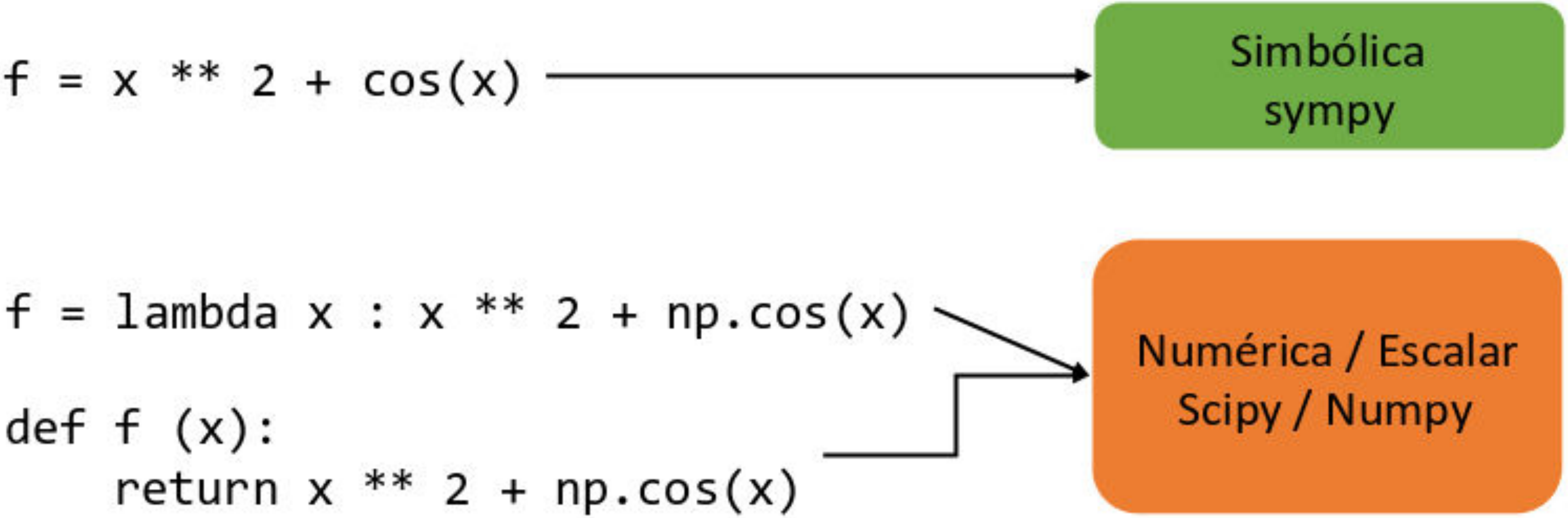


```
f = x ** 2 + cos(x)
```

Simbólica
sympy

```
f = lambda x : x ** 2 + np.cos(x)

def f (x):
    return x ** 2 + np.cos(x)
```

Numérica / Escalar
Scipy / Numpy

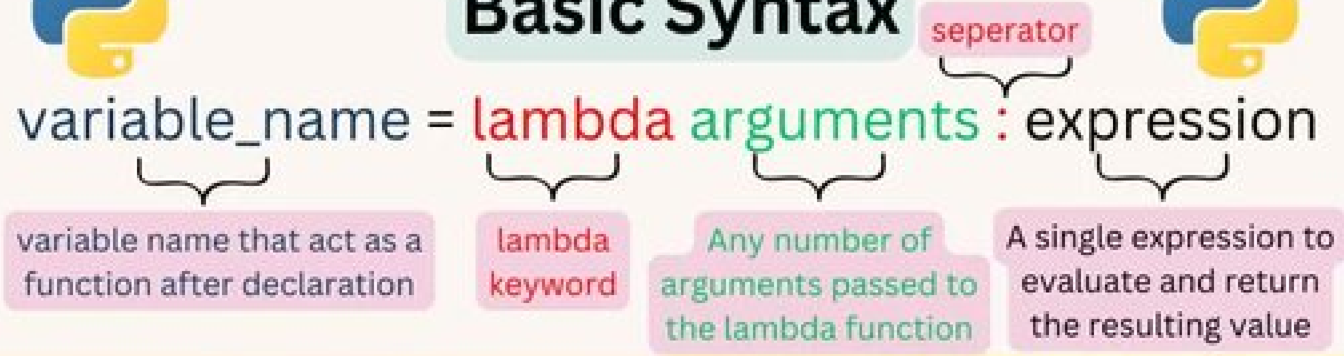
Transformar simbólica en numérica manualmente

`f = x ** 2 + cos(x)` → `f = lambda x : x ** 2 + np.cos(x)`

PYTHON LAMBDA FUNCTION

a small, anonymous function that can be defined in a single line of code

Basic Syntax


seperator
variable_name = lambda arguments : expression
variable name that act as a function after declaration
lambda keyword
Any number of arguments passed to the lambda function
A single expression to evaluate and return the resulting value

Example # 1

```
square = lambda x : x ** 2
result = square(5)
print(result)                #Output:  25
```

Example # 2

```
add = lambda x,y : x + y
result = add(3,4)
print(result)                #Output:  7
```

Lambda function can also be used as parameters for other functions

```
my_list = [1, 2, 3, 4, 5]
result = list(map(lambda x : x ** 2, my_list))
print(result)
#Output:  [1, 4, 9, 16, 25]
```

map() function applies given function to each element of the list

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = list(filter(lambda x : x % 2 == 0, my_list))
print(result)
#Output:  [2, 4, 6, 8, 10]
```

filter() function creates a list of elements for which function is True

```
from functools import reduce

my_list = [1, 2, 3, 4, 5]
result = reduce(lambda x,y : x * y, my_list)
print(result)
#Output:  120
```

reduce() function applies given function to the element of an iterable in a cumulative way, and returns a single value

in salmankhaliq22 @datascholar