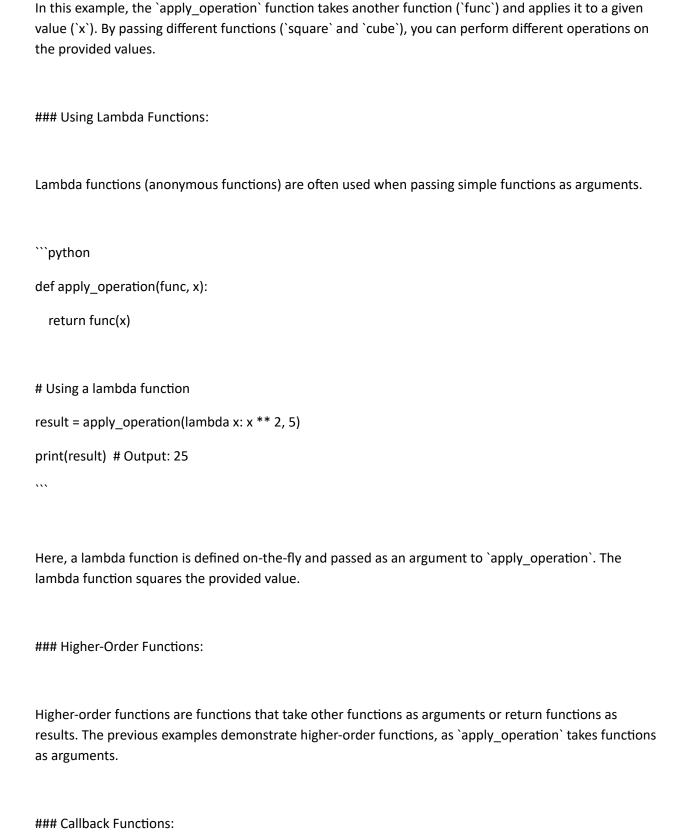
## **Passing Function to a function**

In Python, you can pass functions as arguments to other functions. This concept is a key aspect of functional programming and allows for the creation of more flexible and reusable code. Here's an explanation with an example:

```
### Passing Function as an Argument:
```python
def square(x):
  return x ** 2
def cube(x):
  return x ** 3
def apply_operation(func, x):
  return func(x)
# Passing functions as arguments
result_square = apply_operation(square, 4)
result_cube = apply_operation(cube, 3)
print(result_square) # Output: 16
print(result_cube) # Output: 27
```



Passing functions as arguments is common when working with callback functions. Callback functions are functions that are passed to another function to be executed at a later time or under certain conditions.

```
```python
def process_data(data, callback):
  processed_data = []
  for item in data:
    result = callback(item)
    processed_data.append(result)
  return processed_data
# Using a callback function
numbers = [1, 2, 3, 4, 5]
def square_callback(x):
  return x ** 2
def cube_callback(x):
  return x ** 3
squared_numbers = process_data(numbers, square_callback)
cubed_numbers = process_data(numbers, cube_callback)
print(squared_numbers) # Output: [1, 4, 9, 16, 25]
print(cubed_numbers) # Output: [1, 8, 27, 64, 125]
```

...

Here, the 'process\_data' function takes a list of numbers and a callback function, and it applies the callback function to each item in the list.

### Function Composition:

Function composition involves combining two or more functions to produce a new function. You can pass functions to another function to achieve composition.

```
"python

def compose(f, g):
    return lambda x: f(g(x))

# Composing two functions

double = lambda x: x * 2

square = lambda x: x ** 2

composed_function = compose(double, square)

result = composed_function(3)

print(result) # Output: 18 (double of the square of 3)
```

In this example, the `compose` function takes two functions (`f` and `g`) and returns a new function that is the composition of the two.

assing functions to other functions allows for more dynamic and modular code, promoting a functional rogramming paradigm in Python.