PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Callback

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Functions - Callback

Callback Functions

- A callback function is a function that is **passed to another function as an argument**.

Example:

```
def process_data(data, callback):
    result = [d * 2 for d in data]
    callback(result)      # Call the callback function with the result

def print_result(result): Call Back Function
    print("Processed data:", result)

data = [1, 2, 3]
process_data(data, print_result)
```

Need for Callback in Functions

- **Used in event-driven programming**, where a function is called in response to a specific event or action, such as a button press or the completion of a network request.
- **Also used in functional programming**, where a function is passed as an argument to another function to be used as a "hook" for performing specific operations.
- **Helps to separate functions' functionality** and make code more reusable and modular.

Function: Callback

Example 1 (using built-in function):

```
s=["Hello", "Welcome", "to", "python", "world"]
print(sorted(s))   #The list is sorted based on the ASCII values only.
print(sorted(s, key=str.upper))  #Sort the list based on only the uppercase form of each letter
```

Output

```
['Hello', 'Welcome', 'python', 'to', 'world’]
['Hello', 'python', 'to', 'Welcome', 'world’]
```

Explanation

We are calling the str.upper function inside the sorted function. So, str.upper function is the callback function.

Function: Callback

Example 2 (using user-defined function):

```
def multiply(x):
    return num_list[0]*num_list[1]

def compute(func,x):
    return func(x)

num_list=[2,3]
product=compute(multiply,num_list)
print("Multiplication=",product)
```

Output

```
Multiplication= 6
```

Explanation

compute(multiply,num_list) – the caller function with 2 arguments,
1) a function,multiply and 2) a list,num_list

Here, multiply is the callback function.

Functions - Callback

Example 3 : Multiple Callback functions

```
def function(func_list, x, y):
    print("Inside function")
    for func in func_list:
        func(x,y)

def add(x,y):
    z = x+y
    print('Sum =',z)

def divide(x,y):
    z = x/y
    print('Quotient =',z)

cb_list=[add, divide]
function(cb_list, 10, 5)
```

Output

```
Inside function
Sum = 15
Quotient = 2.0
```

Advantages of Callback Functions

- Calling function (outer function) can **call the callback function as many times as required** to complete the specified task.
- Calling function **can pass appropriate parameters according to the task** to the called functions. This allows **information hiding.**
- Improves **code modularity and reusability**
- Allows you to **dynamically change the working of a function** without changing its core implementation

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CDSAML & CCBD, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu
Prof. Sowmya Shree