CEng 240 – Spring 2021
Week 8

Sinan Kalkan

Functions [continued]

Disclaimer: Figures without reference are from either from "Introduction to programming concepts with case studies in Python" or "Programming with Python for Engineers", which are both co-authored by me.

Previously on CENG240!

Functions in Python

```
def function-name(parameter-1, ..., parameter-N):
        statement-1
        .
        .
        statement-M
```

- Syntax is important!
- Indentation is extremely important!

Previously on CENG240!

Nested Functions in Python

```python
def f(N):
    Number = N
    def g():
        C = 20
        return N * Number
    print("Number", N, "and its square: ", g())
```

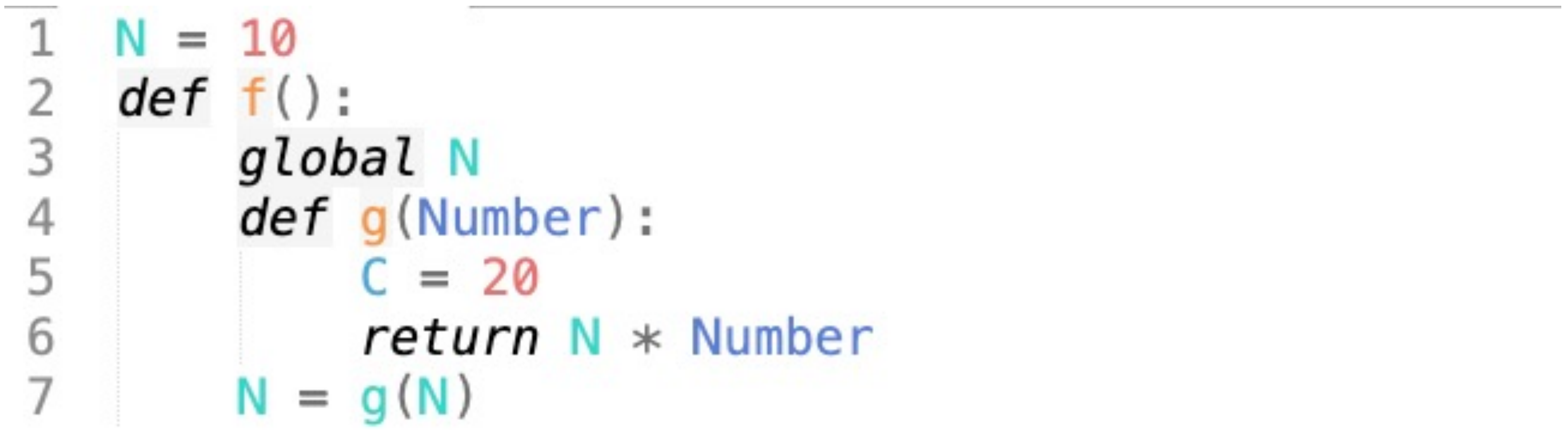
- Function `g()` can access all the local variables as well as the parameters of function `f()`.
- Function `f()` cannot access the local variables of function `g()`!
- Function `g()` cannot be used before it is defined! For example, the second line could not have been `Number = 10 * g(10)`.
- The indentation is extremely important to understand which statement belongs to which function! For example, the last line is part of function `f()` since they are at the same indentation!

METU Computer Engineering

Previously on CENG240!

Global Variables in Python

- To access variables in the global workspace, you should use “global <varname>”



```
N = 10
def f():
    global N
    def g(Number):
        C = 20
        return N * Number
    N = g(N)
```

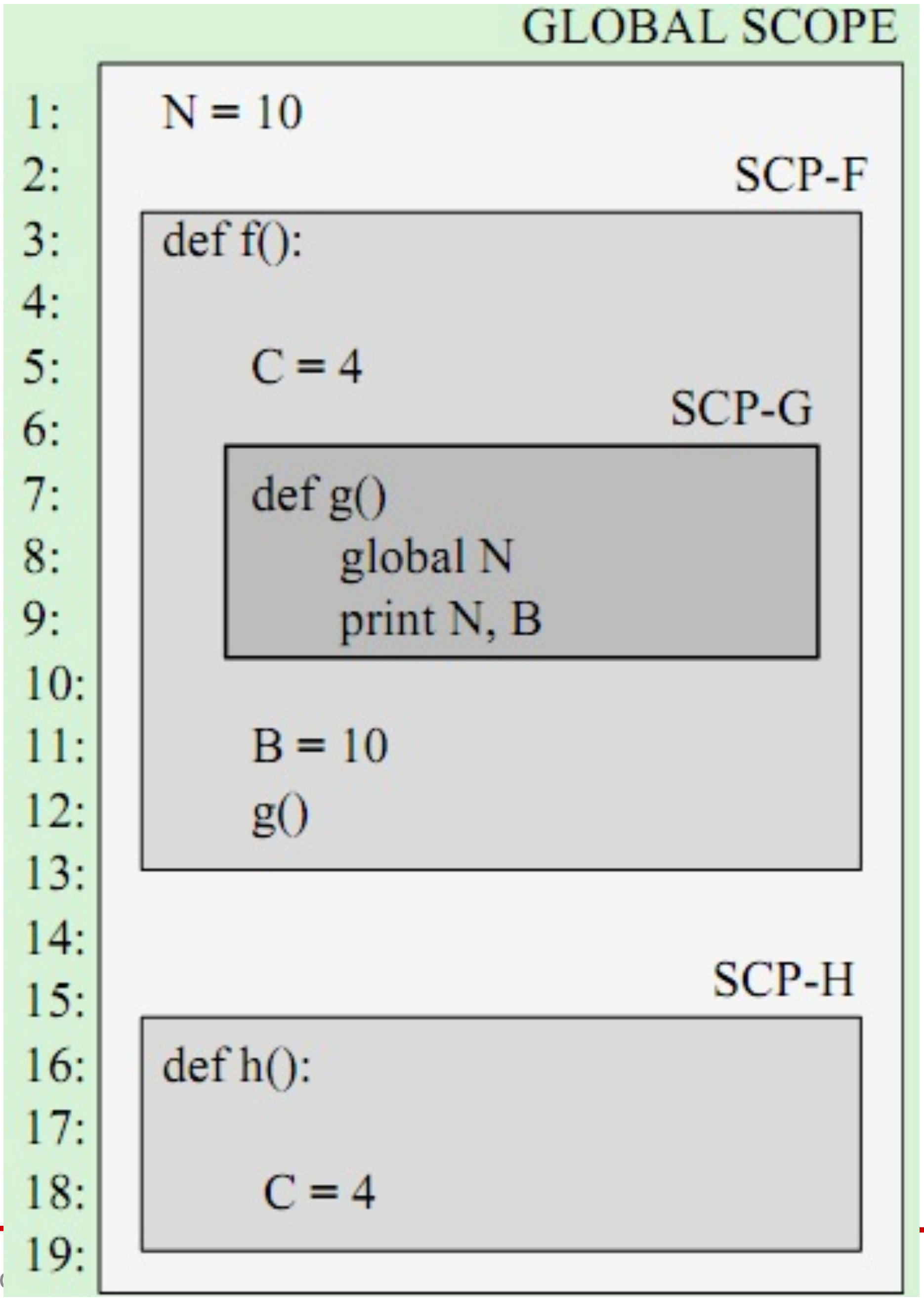
Scope in Python

Previously on CENG240!

- Since you can nest functions in Python, understanding scope is important

- LEGB rule:

Local < Enclosing < Global < Built-in



```
N = 10

def f():

    C = 4

    def g()
        global N
        print N, B

    B = 10
    g()



def h():

    C = 4
```

Parameter passing in functions in Python

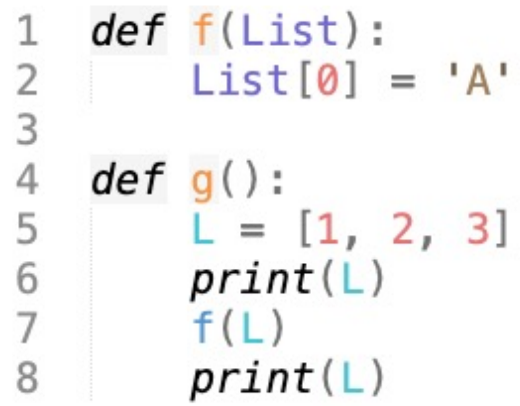
Previously on CENG240!

```python
def f(N):
    N = N + 20

def g():
    A = 10
    print(A)
    f(A)
    print(A)
```

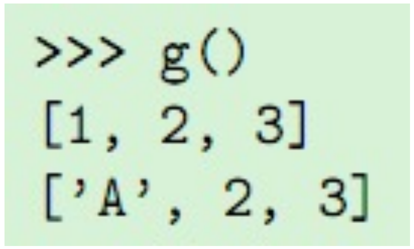
```
>>> g()
10
10
```

Previously on CENG240!

Parameter passing in functions in Python



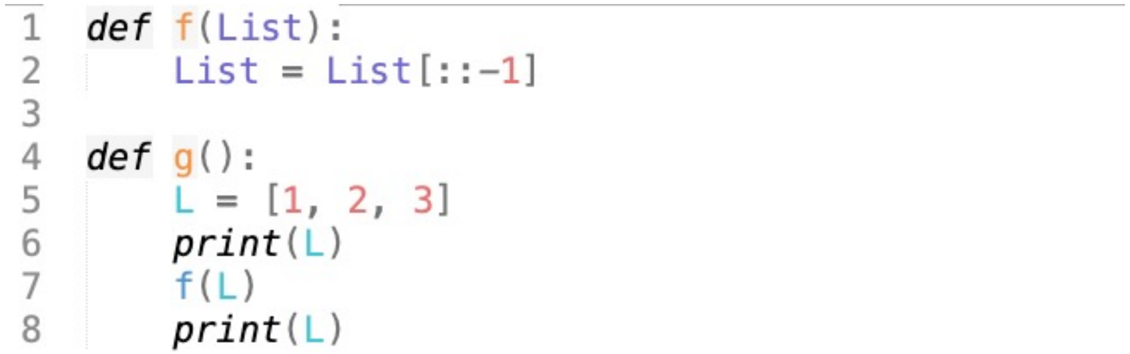
```python
def f(List):
    List[0] = 'A'

def g():
    L = [1, 2, 3]
    print(L)
    f(L)
    print(L)
```



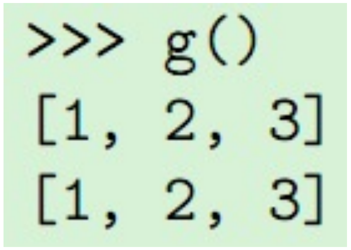
```
>>> g()
[1, 2, 3]
['A', 2, 3]
```

Parameter passing in functions in Python

Previously on CENG240!



```python
def f(List):
    List = List[::-1]

def g():
    L = [1, 2, 3]
    print(L)
    f(L)
    print(L)
```



```
>>> g()
[1, 2, 3]
[1, 2, 3]
```

Previously on CENG240!

METU Computer Engineering

Default Parameters in Python

```python
def reverse_num(Number = 123):
    """reverse_num: Reverse the digits in a number"""
    str_num = str(Number)
    return int(str_num[::-1])
```

- We can now call this function with reverse_num() in which case Number is assumed to be 123.
- If we supply a value for Number, that value is used instead.

This Week

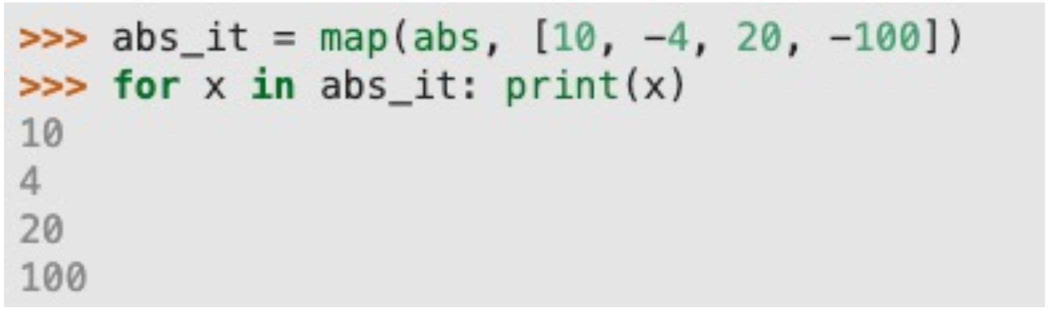
- Previous week was:
  - Why define functions?
  - Defining functions
  - Parameter passing
  - Default parameters
  - Scope of variables
  - Examples
- This week:
  - Recursion, higher-order functions
  - Examples

Administrative Notes

- Lab 5
- Midterm: 1 June, Tuesday, 17:40

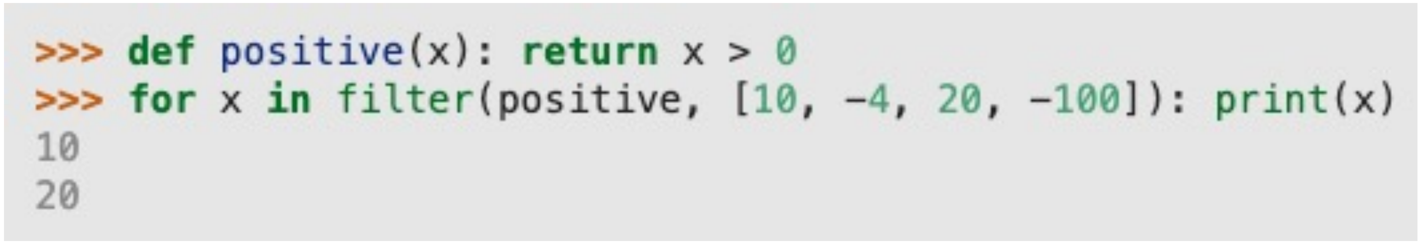
Higher-order functions

- map(function, Iterator)



```
>>> abs_it = map(abs, [10, -4, 20, -100])
>>> for x in abs_it: print(x)
10
4
20
100
```

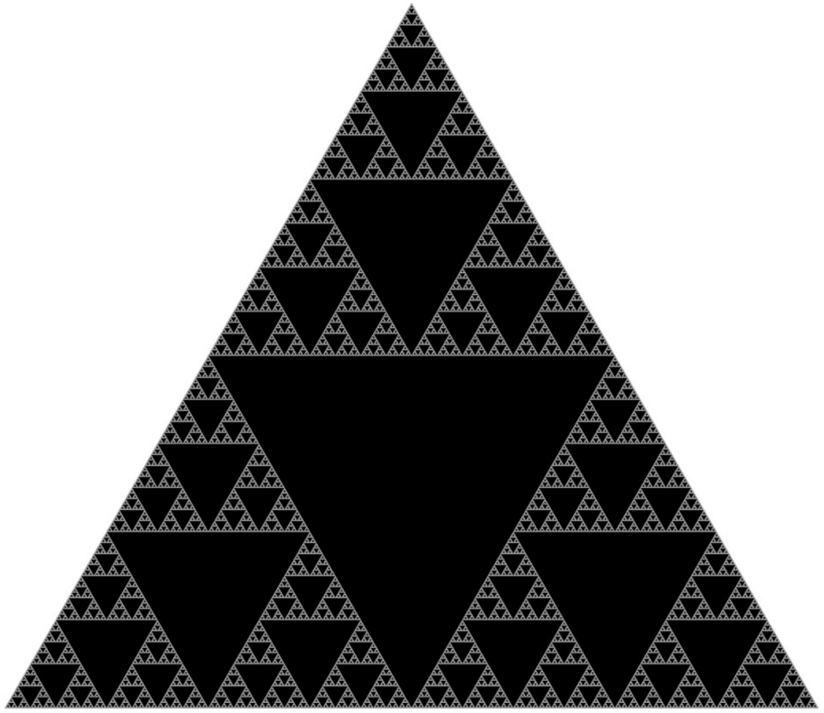
- filter(predicate, Iterator)



```
>>> def positive(x): return x > 0
>>> for x in filter(positive, [10, -4, 20, -100]): print(x)
10
20
```

Recursion: An action wizard

- What is recursion?



Recursion: An example

- Definition of factorial:

Except for zero, the factorial of a natural number is defined to be the number obtained by multiplication of all the natural numbers lesser or equal to it. The factorial of zero is defined, and is set to be 1.

- More formally:

$$\begin{aligned} N! &= 1 \times 2 \times \cdots \times (N-1) \times N \qquad N \in \mathbf{N},\ \ N > 0 \\ 0! &= 1 \end{aligned}$$

Recursion: an example (cont'd)

$$\begin{aligned} N! &= 1 \times 2 \times \cdots \times (N-1) \times N \qquad N \in \mathbf{N},\ \ N > 0 \\ 0! &= 1 \end{aligned}$$

- A careful look at the formal definition:

$$N! = \underbrace{1 \times 2 \times \cdots \times (N-1)}_{(N-1)!} \times N \qquad N \in \mathbf{N},\ \ N > 0$$

$$\begin{aligned} N! &= (N-1)! \times N \qquad N \in \mathbf{N},\ \ N > 0 \\ 0! &= 1 \end{aligned}$$

Factorial uses its own definition!

Recurrence & Recursion

$$\begin{aligned} N! &= (N-1)! \times N \qquad N \in \mathbf{N},\ \ N > 0 \\ 0! &= 1 \end{aligned}$$

- This is called recurrence relation/rule.
- Algorithms which make use of recurrence relations are called recursive.

Recursion: an example (cont'd)

- Let us look at the pseudo-code:

$$\begin{aligned} N! &= (N-1)! \times N \qquad N \in \mathbf{N},\ \ N > 0 \\ 0! &= 1 \end{aligned}$$

```
def factorial(N):
    if N == 0: return 1
   else: return N * factorial(N-1)
```

Recursion: another example

- Recursive definition of even or odd (a mutually recursive example):

  - 0 is even.
  - 1 is odd.
  - A number is even if one less the number is odd.
  - A number is odd if one less the number is even.

Another example for recursion

- Reversing a list / string
  - [a, b, c] → [c, b, a]
- **reverse([a, b, c]) = reverse([b, c]) + [a]**.

Function Examples

What is printed by the following code piece?

```
N = 1
M = 2

def f():
        global M
        N = 3
        M = 4
        print(N, M)

f()
print(N,M)
```

Function Examples

- Finding the median
  - https://pp4e-workbook.github.io/chapters/functions/finding_median.html
- Bankruptcy Countdown:
  - https://pp4e-workbook.github.io/chapters/functions/bankruptcy_countdown.html
- Perfect Number:
  - https://pp4e-workbook.github.io/chapters/functions/perfect_number.html

Final Words: Important Concepts

- Benefits of defining functions
- How to define functions
- Default parameters
- Scopes of variables

THAT'S ALL FOLKS!

STAY HEALTHY