Effective Programming Practices for Economists

Basic Python

Principles for Good Functions

Janoś Gabler and Hans-Martin von Gaudecker

Contents

- Why functions are important!
- Guidelines for self-contained functions
 - Pass all variables you want to use inside
 - Do not modify mutable arguments

Why functions are important

- Help to re-use code and avoid duplication
- Help to structure code and reduce cognitive load
- Make individual code snippets testable
- Help to make your projects more reproducible
- Unlock the power of functional programming concepts
- Are also the basis for good object oriented code

Pass all variables you want to use inside

```
# bad example
>>> global_msg = "Hello {}!"
>>> def greet_with_global(name):
... print(global_msg.format(name))
>>> greet_with_global("Guido")
Hello Guido!
```

- Inside a function you have access to variables in the enclosing scope
- This is dangerous because the behaviour of the function now depends on global variables
- Do not use this in your code!

Pass all variables you want to use inside

- Inside a function you have access to variables in the enclosing scope
- This is dangerous because the behaviour of the function now depends on global variables
- Do not use this in your code!

Do not modify mutable arguments

```
>>> def append_4(some_list):
... some_list.append(4)
... return some_list
>>> a = [1, 2, 3]
>>> append_4(a)
[1, 2, 3, 4]
>>> a
[1, 2, 3, 4]
# better solution
>>> def append_4(some_list)
       out = some_list.copy()
... out.append(4)
    return out
```

- Arguments are passed by reference,
 i.e. without making a copy
- Make sure that functions do not modify mutable arguments!
 - Make copies
 - Avoid changing objects in the first place